Prime Implicates and Prime Implicants in Modal Logic: Extended and Revised Version

Meghyn Bienvenu
IRIT-UPS, 118 route de Narbonne,
31062 Toulouse Cedex, France

IRIT Research Report IRIT/RR–2007-17–FR

December 2007

Latest modification: January 8, 2008

Abstract

The purpose of this paper is to examine how the propositional notions of prime implicates and prime implicants might be appropriately extended to the modal logic $\mathcal{K}$. We begin the paper by considering a number of potential definitions of clauses and terms for $\mathcal{K}$. The different definitions are evaluated with respect to a set of syntactic, semantic, and complexity-theoretic properties characteristic of the propositional definition. We then compare the definitions with respect to the properties of the notions of prime implicates and prime implicants that they induce. While there is no definition which perfectly generalizes the propositional notion, we show that there does exist one definition which satisfies many of the desirable properties of the propositional case. In the second half of the paper, we consider the computational properties of this definition. To this end, we provide sound and complete algorithms for generating and recognizing prime implicates, and we prove the prime implicate recognition task to be PSPACE-complete. While the paper focusses on the logic $\mathcal{K}$, all of our results hold equally well for multi-modal $\mathcal{K}$ and for concept expressions in the description logic $\mathcal{ALC}$.

1 Introduction

Prime implicates and prime implicants are important notions in artificial intelligence. They have given rise to a significant body of work in automated reasoning and have been applied to a number of different sub-areas in AI, among them distributed reasoning [1], belief revision (cf. [3], [27]), non-monotonic reasoning (cf. [29]), relevance (cf. [23], [22], [21]), knowledge compilation (cf. [6], [9]), and abduction and diagnosis (cf. [10], [13]). Traditionally, these concepts have been studied in the context of propositional logic, but they have also been considered for many-valued [30] and first-order logic ([24], [25]). Surprisingly, however, no definition of prime implicate or prime implicant has ever been proposed for a modal or description logic, nor has it been shown that no reasonable definition can be provided. Given the increasing interest in modal and

description logics as knowledge representation languages, one naturally wonders whether these notions can be suitably generalized to these more expressive logics.

This question is not only of theoretical but also of practical interest as there are a variety of settings in which prime implicates/implicants could prove useful. One obvious domain of application is abductive reasoning and diagnosis in modal and description logics, an as yet largely unaddressed problem whose importance has been argued for in [14]. In this context, prime implicants play the role of abductive explanations: the minimal explanations of an observation o with respect to a background theory t are just the prime implicants of $t \rightarrow o$.

Prime implicates might also prove an important tool in rendering modal and description logic knowledge bases more accessible to knowledge engineers and end users alike. For instance, prime implicates might aid knowledge engineers in evaluating the consequences of adding new pieces of information ("find all prime implicates of KB + *new* that are not prime implicates of KB"). They could also be used as the basis for rich query languages that allow users to pose general questions about the contents of a KB ("find all of the prime implicates concerning professors").

The aim of this paper is to investigate the notions of prime implicates and prime implicants for the modal logic $\mathcal{K}$. We have chosen to begin our investigation with $\mathcal{K}$ simply because it is the most basic modal logic. All of our results can be straightforwardly extended to multi-modal $\mathcal{K}$ and, via the well-known correspondence [31], to concept expressions in the description logic $\mathcal{ALC}$. Other modal and description logics as well as interesting restricted forms of prime implicates will be examined in future work.

Our paper is organized as follows. After some preliminaries, we consider how to generalize the notions of clauses and terms to $\mathcal{K}$. As there is no obvious definition, we enumerate a list of syntactic, semantic, and complexity-theoretic properties of the propositional definitions, which we then use to compare the different candidate definitions. We next consider the different definitions in light of the notions of prime implicate and prime implicant they induce. Once again, we list some basic properties from the propositional case that we would like to satisfy, and we see how the different definitions measure up. In the second half of the paper, we investigate the computational properties of the most satisfactory definition of prime implicates. We consider the problems of prime implicate generation and recognition, and we provide sound and complete algorithms for both tasks. We also study the complexity of the prime implicate recognition problem, showing it to be PSPACE-complete and thus of the same complexity as satisfiability and deduction in $\mathcal{K}$. We conclude the paper with a discussion of interesting avenues of future research. In order to enhance readability, proofs have been omitted from the body of the text. Full proofs for all results can be found in Appendix B.

2 Preliminary Definitions and Notation

In this section, we recall some standard notions from propositional logic, modal logic, and computational complexity.

2.1 Propositional Logic

Formulae in propositional logic are built from a set of propositional variables and the logical connectives $\neg$ (negation), $\wedge$ (conjunction), and $\vee$ (disjunction). A *literal* is either a propositional variable or the negation of a propositional variable. A *clause* is defined to be a disjunction of

literals, and a *term* is a conjunction of literals. A formula is said to be in *negation normal form* (NNF) if negation only appears directly before propositional variables. A formula is in *conjunctive normal form* (CNF) if it is a conjunction of clauses, and it is in *disjunctive normal form* (DNF) if it is a disjunction of terms. Any formula can be put into NNF in linear time, but the transformation to CNF or DNF can be exponential in both time and space.

The semantics of propositional logic is defined with respect to *valuations*, which assign a truth value to each of the propositional variables of the language. Satisfaction of a formula ϕ by a valuation v is defined as follows:

- if ϕ is a propositional variable a, then $v \models \phi$ if and only if $v(a) = true$
- if $\phi = \neg\psi$, then $v \models \phi$ if and only if it is not the case that $v \models \phi$
- if $\phi = \psi_1 \wedge \psi_2$, then $v \models \phi$ if and only if both $v \models \psi_1$ and $v \models \phi_2$
- if $\phi = \psi_1 \vee \psi_2$, then $v \models \phi$ if and only if either $v \models \psi_1$ or $v \models \phi_2$

A formula ϕ is said to be a *logical consequence* of a formula ψ just in the case that every valuation satisfying ϕ also satisfies ψ. We say that ϕ is *logically stronger* than ψ whenever ϕ is a logical consequence of ψ but ψ is not a logical consequence of ϕ.

A clause is a *prime implicate* of a formula if it is a logical consequence of the formula and there are no logically stronger clauses implied by the formula. A term is a *prime implicant* of a formula if it implies the formula and there are no logically weaker terms which imply the formula. For a comprehensive overview of prime implicates and prime implicants, the reader is directed to [26].

2.2 Modal Logic $\mathcal{K}$

We briefly recall the basics of the modal logic $\mathcal{K}$[1]. Formulae in $\mathcal{K}$ are built up from a set of propositional variables $\mathcal{V}$, the standard logical connectives, and the modal operators $\Box$ and $\Diamond$. We will use $var(\phi)$ to refer to the set of propositional variables appearing in a formula ϕ. The modal depth of a formula ϕ, written $\delta(\phi)$, is defined as the maximal number of nested modal operators appearing in ϕ, e.g. $\delta(\Diamond(a \wedge \Box a) \vee a) = 2$. We define the length of a formula ϕ, written $|\phi|$, to be the number of occurrences of propositonal variables and logical connectives in ϕ. Negation normal form (NNF) is defined just as in propositional logic. Every formula in $\mathcal{K}$ can be transformed in linear time into an equivalent formula in NNF of the same modal depth via a straightforward application of the equivalences $\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi$, $\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$, $\neg\neg\phi \equiv \phi$, $\neg\Box\phi \equiv \Diamond\neg\phi$, and $\neg\Diamond\phi \equiv \Box\neg\phi$ (see Appendix A for details). For example, applying this transformation to the formula $\neg\Box(a \wedge \Diamond(\neg b \vee c))$ results in the formula $\Diamond(\neg a \vee \Box(b \wedge \neg c))$ which is in NNF.

A model for $\mathcal{K}$ is a tuple $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, v \rangle$, where $\mathcal{W}$ is a non-empty set of possible worlds, $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{W}$ is a binary relation over worlds, and $v : \mathcal{W} \times \mathcal{V} \rightarrow \{true, false\}$ is a valuation of the propositional variables at each world. Satisfaction of a formula ϕ in a model $\mathcal{M}$ at the world w (written $\mathcal{M}, w \models \phi$) is defined as follows:

- $\mathcal{M}, w \models a$ if and only if $v(w, a) = true$

[1] Refer to [4] or [7] for good introductions to modal logic.

- $\mathcal{M}, w \models \neg\phi$ if and only if $\mathcal{M}, w \not\models \phi$
- $\mathcal{M}, w \models \phi \wedge \psi$ if and only if $\mathcal{M}, w \models \phi$ and $\mathcal{M}, w \models \psi$
- $\mathcal{M}, w \models \phi \vee \psi$ if and only if $\mathcal{M}, w \models \phi$ or $\mathcal{M}, w \models \psi$
- $\mathcal{M}, w \models \Box\phi$ if and only if $\mathcal{M}, w' \models \phi$ for all w' such that $w\mathcal{R}w'$
- $\mathcal{M}, w \models \Diamond\phi$ if and only if $\mathcal{M}, w' \models \phi$ for some w' such that $w\mathcal{R}w'$

A formula ϕ is said to be a *tautology*, written $\models \phi$, if $\mathcal{M}, w \models \phi$ for every model M and world w. A formula ϕ is *satisfiable* if there is some model $\mathcal{M}$ and some world w such that $\mathcal{M}, w \models \phi$. If there is no $\mathcal{M}$ and w for which $\mathcal{M}, w \models \phi$, then ϕ is called *unsatisfiable*, and we write $\phi \models \bot$.

In modal logic, the notion of logical consequence can be defined in one of two ways:

- a formula ψ is a *global consequence* of ϕ if whenever $\mathcal{M}, w \models \phi$ for every world w of a model $\mathcal{M}$, then $\mathcal{M}, w \models \psi$ for every world w of $\mathcal{M}$
- a formula ψ is a *local consequence* of ϕ if $\mathcal{M}, w \models \phi$ implies $\mathcal{M}, w \models \psi$ for every model $\mathcal{M}$ and world w

The distinction between local and global consequence does not arise in propositional logic because each model contains a single possible world. In first-order logic, both notions of consequence exist, but local consequence is by far the most studied. In this paper, we will only consider the notion of local consequence, and we will take $\phi \models \psi$ to mean that ϕ is a local consequence of ψ. Two formulae ϕ and ψ are called *equivalent*, written $\phi \equiv \psi$, if both $\phi \models \psi$ and $\psi \models \phi$. A formula ϕ is *logically stronger* than ψ if $\phi \models \psi$ and $\psi \not\models \phi$.

We now highlight some basic properties of logical consequence and equivalence in $\mathcal{K}$ which will play an important role in the proofs of our results.

Theorem 1. *Let γ, ψ, ψ_1, ..., ψ_m, χ, χ_1, ..., χ_n be formulae in $\mathcal{K}$, and let γ be a propositional formula. Then*

1. $\psi \models \chi \Leftrightarrow \models \neg\psi \vee \chi \Leftrightarrow \psi \wedge \neg\chi \models \bot$
2. $\psi \models \chi \Leftrightarrow \Diamond\psi \models \Diamond\chi \Leftrightarrow \Box\psi \models \Box\chi$
3. $\gamma \wedge \Diamond\psi_1 \wedge ... \wedge \Diamond\psi_m \wedge \Box\chi_1 \wedge ... \wedge \Box\chi_n \models \bot \Leftrightarrow (\gamma \models \bot$ *or* $\psi_i \wedge \chi_1 \wedge ... \wedge \chi_n \models \bot$ *for some* $i)$
4. $\models \gamma \vee \Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee \Box\chi_1 \vee ... \vee \Box\chi_n \Leftrightarrow (\models \gamma$ *or* $\models \psi_1 \vee ... \vee \psi_m \vee \chi_i$ *for some* $i)$
5. $\Box\chi \models \Box\chi_1 \vee ... \vee \Box\chi_n \Leftrightarrow \chi \models \chi_i$ *for some* i
6. $\Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee \Box\chi_1 \vee ... \vee \Box\chi_n$
   $\equiv \Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee \Box(\chi_1 \vee \psi_1 \vee ... \vee \psi_m) \vee ... \vee \Box(\chi_n \vee \psi_1 \vee ... \vee \psi_m)$

Theorem 2. *Let λ be a disjunction of propositional literals and formulae of the forms $\Diamond\psi$ and $\Box\chi$. Then each of the following statements holds:*

1. *If $\lambda \models \gamma$ for some non-tautological propositional clause γ, then every disjunct of λ is either a propositional literal or a formula $\Diamond\psi$ where $\psi \models \bot$*

2. *If* $\lambda \models \Diamond\psi_1 \vee ... \vee \Diamond\psi_n$, *then every disjunct of* λ *is of the form* $\Diamond\psi$

3. *If* $\lambda \models \Box\chi_1 \vee ... \vee \Box\chi_n$ *and* $\not\models \Box\chi_1 \vee ... \vee \Box\chi_n$, *then every disjunct of* λ *is either a formula of the form* $\Box\chi$ *or a formula* $\Diamond\psi$ *where* $\psi \models \bot$

Theorem 3. *Let* $\lambda = \gamma \vee \Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee \Box\chi_1 \vee ... \vee \Box\chi_n$ *and* $\lambda' = \gamma' \vee \Diamond\psi'_1 \vee ... \vee \Diamond\psi'_p \vee \Box\chi'_1 \vee ... \vee \Box\chi'_q$ *be formulae in* $\mathcal{K}$. *If* γ *are* γ' *are both propositional and* $\not\models \lambda'$, *then*

$$\lambda \models \lambda' \Leftrightarrow \begin{cases} \gamma \models \gamma' \text{ and} \\ \psi_1 \vee ... \vee \psi_m \models \psi'_1 \vee ... \vee \psi'_p \text{ and} \\ \text{for every } i \text{ there is some } j \text{ such that } \chi_i \models \psi'_1 \vee ... \vee \psi'_p \vee \chi'_j \end{cases}$$

2.3 Complexity Theory

We recall some basic definitions and results from computational complexity theory (cf. [28]). The class P comprises all languages which can be recognized in polynomial time by a deterministic Turing machine. The class NP contains all languages which can be recognized in polynomial time by a non-deterministic Turing machine. The class co-NP is defined to be the set of all languages whose complement belongs to NP. The class BH_2 is defined as the set of languages $L = L_1 \cap L_2$ such that L_1 is in NP and L_2 is in co-NP. The class PSPACE (respectively EXPSPACE) is comprised of those problems which can be solved in polynomial (respectively single-exponential) space by a deterministic Turing machine. It is well-known that PSPACE=NPSPACE=co-PSPACE and EXSPACE=NEXPSPACE=co-EXPSPACE.

In [20], satisfiability and unsatisfiability of formulae in $\mathcal{K}$ were shown to be PSPACE-complete. For PSPACE membership, Ladner exhibited a polynomial space tableaux-style algorithm for deciding satisfiability of $\mathcal{K}$ formulae. PSPACE-hardness was proven by means of a reduction from QBF validity (the canonical PSPACE-complete problem).

3 Literals, clauses, and terms in $\mathcal{K}$

As we have seen in the previous section, the notions of prime implicates and implicants are straightforwardly defined using the notions of clauses and terms. Thus, if we aim to provide suitable definitions of prime implicates and implicants for the logic $\mathcal{K}$, we first need to decide upon a suitable definition of clauses and terms in $\mathcal{K}$. Unfortunately, whereas clauses and terms are standard notions in both propositional and first-order logic[2], there is no generally accepted definition of clauses and terms in $\mathcal{K}$. Indeed, several quite different notions of clauses and terms have been proposed in the literature for different purposes.

Instead of blindly picking a definition and hoping that it is appropriate, we prefer to list a number of characteristics of literals, clauses, and terms in propositional logic, giving us a principled means of comparing different candidate definitions. Each of the properties below describes something of what it is to be a literal, clause, or term in propositional logic. Although

[2]One might wonder why we don't simply translate our formulae in $\mathcal{K}$ into first-order formulae and then put them into clausal form. The reason is simple: we are looking to define clauses and terms *within* the language of $\mathcal{K}$, and the clauses we obtain on passing by first-order logic are generally not expressible in $\mathcal{K}$. Moreover, if we were to define clauses in $\mathcal{K}$ as those first-order clauses which are representable in $\mathcal{K}$, we would obtain a set of clauses containing no $\Diamond$ modalities, thereby losing much of the expressivity of $\mathcal{K}$.

our list cannot be considered exhaustive, we do believe that it covers the principal syntactic, semantic, and complexity-theoretic properties of the propositional definition.

P1 Literals, clauses, and terms are in negation normal form.

P2 Clauses do not contain $\wedge$, terms do not contain $\vee$, and literals contain neither $\wedge$ nor $\vee$.

P3 Clauses (resp. terms) are disjunctions (resp. conjunctions) of literals.

P4 The negation of a literal is equivalent to another literal. Negations of clauses (resp. terms) are equivalent to terms (resp. clauses).

P5 Every formula is equivalent to a finite conjunction of clauses. Likewise, every formula is equivalent to a finite disjunction of terms.

P6 The task of deciding whether a given formula is a literal, term, or clause can be accomplished in polynomial-time.

P7 The task of deciding whether a clause (resp. term) entails another clause (resp. term) can be accomplished in polynomial-time.

One may wonder whether there exist definitions of literals, clauses, and terms for $\mathcal{K}$ satisfying all of these properties. Unfortunately, we can show this to be impossible.

Theorem 4. *Any definition of literals, clause, and terms for $\mathcal{K}$ that satisfies properties **P1** and **P2** cannot satisfy **P5**.*

The proof of Theorem 4 only makes use of the fact that $\wedge$ does not distribute over $\Diamond$ and $\vee$ does not distribute over $\Box$, which means that our impossibility result holds equally well for most standard modal and description logics.

The first definition that we will consider is that proposed in [8] in the context of abduction. The authors define terms to be the formulae which can be constructed from the propositional literals using only $\wedge$, $\Box$, and $\Diamond$. Modal clauses and literals are not used in the paper but can be defined analagously, yielding the following definition[3]:

D1
$$\begin{aligned} L &::= a \mid \neg a \mid \Box L \mid \Diamond L \\ C &::= a \mid \neg a \mid \Box C \mid \Diamond C \mid C \vee C \\ T &::= a \mid \neg a \mid \Box T \mid \Diamond T \mid T \wedge T \end{aligned}$$

It is easy to see by inspection that this definition satisfies properties **P1**-**P2**, **P4**, and **P6**. Property **P3** is not satisfied, however, since there are clauses that are not disjunctions of literals – take for instance $\Box(a \vee b)$. From Theorem 4 and the fact that both **P1** and **P2** are satisfied, we can conclude that property **P5** cannot hold. At first glance, it may seem that entailment between clauses or terms could be accomplished in polynomial time, but this is not the case. In fact, we can show this problem to be NP-complete. The proof relies on the very strong resemblance between terms of **D1** and concept expressions in the description logic $\mathcal{ALE}$ (for which both unsatisfiability and deduction are known to be NP-complete).

[3]Note that here and in what follows, we let a range over propositional variables and L, C, and T range over the sets of literals, clauses, and terms respectively.

By using a slightly different definition, we can gain **P3**:

D2
$$L ::= a \mid \neg a \mid \Box L \mid \Diamond L$$
$$C ::= L \mid C \vee C$$
$$T ::= L \mid T \wedge T$$

It can be easily verified that definition **D2** satisfies properties **P1**-**P4** and **P6**. As definition **D1** does not satisfy **P5**, and definition **D2** is even less expressive, it follows that **D2** does not satisfy **P5** either. This reduced expressiveness does not however improve its computational complexity: property **P7** is still not satisfied as we can show that entailment between clauses or terms is NP-complete using the same reduction as was used for definition **D1**.

Given that even very inexpressive definitions like **D2** fail to gain us polynomial behavior, it seems reasonable to explore some more expressive options. We begin with the following definition of clauses that was proposed in [15] for the purpose of modal resolution:

D3
$$C ::= a \mid \neg a \mid \Box C \mid \Diamond ConjC \mid C \vee C$$
$$ConjC ::= C \mid ConjC \wedge ConjC$$

This definition of clauses can be extended to a definition of terms and literals which satisfies **P3** or **P4**, but there is no extension which satisfies both properties. Let us first consider one of the possible extensions which satisfies **P4** and a maximal subset of **P1**-**P7**:

D3a
$$L ::= a \mid \neg a \mid \Box L \mid \Diamond L$$
$$C ::= a \mid \neg a \mid \Box C \mid \Diamond ConjC \mid C \vee C$$
$$ConjC ::= C \mid ConjC \wedge ConjC$$
$$T ::= a \mid \neg a \mid \Box DisjT \mid \Diamond T \mid T \wedge T$$
$$DisjT ::= T \mid DisjT \vee DisjT$$

This definition satisfies **P1**, and **P4**-**P6** (**P5** is a consequence of Proposition 1.3 in [15]). It does not satisfy **P3** as there are clauses that are not disjunctions of literals – take for example $\Box(a \vee b)$. Given that definition **D3a** is strictly more expressive than definitions **D1** and **D2**, it follows that entailment between clauses or terms must be NP-hard, which means that **D3a** does not satisfy **P7**. In fact, we can show that entailment between clauses or terms of definition **D3a** is PSPACE-complete. To do so, we modify the polynomial translation of QBF into $\mathcal{K}$ used to prove PSPACE-hardness of $\mathcal{K}$ so that the translated formula is a conjunction of clauses with respect to **D3a**. We then notice that a formula ϕ is unsatisfiable if and only if $\Diamond\phi$ entails $\Diamond(a \wedge \neg a)$. We thus reduce QBF validity to entailment between clauses, making this task PSPACE-hard, and hence (being a subproblem of entailment in $\mathcal{K}$) PSPACE-complete. This same idea is used to show PSPACE-completeness for definitions **D3b** and **D5** below.

If instead we extend **D3** so as to enforce property **P3**, we obtain the following definition:

D3b
$$L ::= a \mid \neg a \mid \Box C \mid \Diamond ConjC$$
$$C ::= a \mid \neg a \mid \Box C \mid \Diamond ConjC \mid C \vee C$$
$$ConjC ::= C \mid ConjC \wedge ConjC$$
$$T ::= L \mid T \wedge T$$

This definition satisfies all of the properties except **P2**, **P4**, and **P7**. Property **P4** fails to hold because the negation of the literal $\Diamond(a \vee b)$ is not equivalent to any literal. The proof that **P5** holds is constructive: we use standard logical equivalences to rewrite formulae as equivalent conjunctions of clauses and disjunctions of terms (this is also what we do for definitions **D4** and **D5** below).

We now propose two rather simple definitions that satisfy properties **P3**, **P4**, and **P5**. The first definition, which is inspired by the notion of modal atom proposed in [18], defines literals as the set of formulae in NNF that cannot be decomposed propositionally.

D4
$$\begin{aligned} L &::= a \mid \neg a \mid \Box F \mid \Diamond F \\ C &::= L \mid C \vee C \\ T &::= L \mid T \wedge T \\ F &::= a \mid \neg a \mid F \wedge F \mid F \vee F \mid \Box F \mid \Diamond F \end{aligned}$$

D4 satisfies all of the properties except **P2** and **P7**. For **P7**, we note that an arbitrary formula ϕ in NNF is unsatisfiable (a PSPACE-complete problem) if and only if $\Diamond\phi \models \Diamond(a \wedge \neg a)$.

Definition **D4** is very liberal, imposing no structure on the formulae behind modal operators. If we define literals to be the formulae in NNF that cannot be decomposed *modally* (instead of propositionally), we obtain a much stricter definition which satisfies exactly the same properties as **D4**.

D5
$$\begin{aligned} L &::= a \mid \neg a \mid \Box C \mid \Diamond T \\ C &::= L \mid C \vee C \\ T &::= L \mid T \wedge T \end{aligned}$$

A summary of our analysis of the different definitions with respect to properties **P1**-**P7** is provided in the following table.

	D1	**D2**	**D3a**	**D3b**	**D4**	**D5**
P1	yes	yes	yes	yes	yes	yes
P2	yes	yes	no	no	no	no
P3	no	yes	no	yes	yes	yes
P4	yes	yes	yes	no	yes	yes
P5	no	no	yes	yes	yes	yes
P6	yes	yes	yes	yes	yes	yes
P7	no (NP-complete)		no (PSPACE-complete)			

Figure 1: Properties of the different definitions of literals, clauses, and terms.

Theorem 5. *The results in Figure 1 hold.*

Clearly deciding between different candidate definitions is more complicated than counting up the number of properties that the definitions satisfy, the simple reason being that some properties are more important than others. Take for instance property **P5** which requires clauses and terms to be expressive enough to represent all of the formulae in $\mathcal{K}$. If we just use the propositional definition (thereby disregarding the modal operators), then we find that it satisfies every property except **P5**, and hence more properties than any of the definitions considered in this section, and yet we would be hard-pressed to find someone who considers the propositional definition an appropriate definition for $\mathcal{K}$. This demonstrates that expressiveness is a particularly important property, so important in fact that we should be willing to sacrifice properties **P2** and **P7** to keep it. Among the definitions that satisfy **P5**, we prefer definitions **D4** and **D5** to definitions **D3a** and **D3b**. The latter definitions have less in common with the propositional definition and present no advantages over **D4** and **D5**.

Of course, when it comes down to it, the choice of a definition must depend on the particular application in mind. There may be very well be circumstances in which a less expressive or less

elegant definition may prove to be the most suitable. In this paper we are using clauses and terms to define prime implicates and prime implicants, so for us the most important criteria for choosing a definition will be the quality of the notions of prime implicates and prime implicants that the definition induces.

4 Prime Implicates/Implicants in $\mathcal{K}$

Once a definition of clauses and terms has been fixed, we can define prime implicates and prime implicants in exactly the same manner as in propositional logic:

Definition 6. A clause λ is an implicate of a formula ϕ if and only if $\phi \models \lambda$. λ is a *prime implicate* of ϕ if and only if:

1. λ is an implicate of ϕ
2. If λ' is an implicate of ϕ such that $\lambda' \models \lambda$, then $\lambda \models \lambda'$

Definition 7. A term κ is an implicant of the formula ϕ if and only if $\kappa \models \phi$. κ is a *prime implicant* of ϕ if and only if:

1. κ is an implicant of ϕ
2. If κ' is an implicant of ϕ such that $\kappa \models \kappa'$, then $\kappa' \models \kappa$

Of course, the notion of prime implicate (respectively implicant) that we get will be determined by the definition of clause (respectively term) that we have chosen. We will compare different definitions using the following well-known properties of prime implicates/implicants in propositional logic:

Finiteness The number of prime implicates (respectively prime implicants) of a formula is finite modulo logical equivalence.

Covering Every implicate of a formula is entailed by some prime implicate of the formula. Conversely, every implicant of a formula entails some prime implicant of the formula.

Equivalence A model $\mathcal{M}$ is a model of ϕ if and only if $\mathcal{M}$ is a model of all the prime implicates of ϕ if and only if $\mathcal{M}$ is a model of some prime implicant of ϕ[4].

Implicant-Implicate Duality Every prime implicant of a formula is equivalent to the negation of some prime implicate of the negated formula. Conversely, every prime implicate of a formula is equivalent to the negation of a prime implicant of the negated formula.

Distribution If λ is a prime implicate of $\phi_1 \vee ... \vee \phi_n$, then there exist prime implicates λ_1, ..., λ_n of ϕ_1, ..., ϕ_n such that $\lambda \equiv \lambda_1 \vee ... \vee \lambda_n$. Likewise, if κ is a prime implicant of $\phi_1 \wedge ... \wedge \phi_n$, then there exist prime implicants κ_1, ..., κ_n of ϕ_1, ..., ϕ_n such that $\kappa \equiv \kappa_1 \wedge ... \wedge \kappa_n$

[4]The property Equivalence is more commonly taken to mean that a formula is equivalent to the conjunction of its prime implicates and the disjunction of its prime implicants. We have chosen a model-theoretic formulation in order to allow for the possibility that the set of prime implicates/implicants is infinite.

Finiteness ensures that the prime implicates/implicants of a formula can be finitely represented, while **Covering** means the prime implicates provide a complete representation of the formula's implicates. **Equivalence** guarantees that no information is lost in replacing a formula by its prime implicates/implicants, whereas **Implicant-Implicate Duality** allows us to transfer results and algorithms for prime implicates to prime implicants, and vice-versa. Finally, **Distribution** relates the prime implicates/implicants of a formula to the prime implicates/implicants of its sub-formulae. This property will play a key role in the prime implicate generation algorithm presented in the next section.

We can show that definition **D4** satisfies all five properties. For **Finiteness** and **Covering**, we first demonstrate that every implicate λ of a formula ϕ is entailed by some implicate λ' of ϕ with $var(\lambda') \subseteq var(\phi)$ and having depth at most $\delta(\phi) + 1$ (and similarly for implicants). As there are only finitely many non-equivalent formulae on a finite language and with bounded depth, it follows that there are only finitely many prime implicates/implicants of a given formula, and that there can be no infinite chains of increasingly stronger implicates/implicants. **Equivalence** follows directly from **Covering** and the property **P5** of the previous section: we use **P5** to rewrite ϕ as a conjunction of clauses, each of which is implied by some prime implicate of ϕ because of **Covering**. The property **Implicant-Implicate Duality** is an immediate consequence of the duality between clauses and terms (**P4**). **Distribution** can be shown using **Covering** plus the fact that a disjunction of clauses is a clause and a conjunction of terms is a term (**P3**).

Theorem 8. *The notions of prime implicates and prime implicants induced by definition **D4** satisfy **Finiteness**, **Covering**, **Equivalence**, **Implicant-Implicate Duality**, and **Distribution**.*

We remark by way of contrast that in first-order logic the notion of prime implicates induced by the standard definition of clauses has been shown to falsify **Finiteness**, **Covering**, **Equivalence**, and **Distribution** [24].

We now show that definition **D4** is the only one of our definitions to satisfy all five properties. For definitions **D1** and **D2**, we prove that **Equivalence** does not hold. This is a fairly straightforward consequence of the fact that these definitions do not satisfy property **P5**. For definitions **D3a**, **D3b**, and **D5**, we prove that $\Box(\Diamond^k a) \vee \Diamond(a \wedge b \wedge \Box^k \neg a)$ is a prime implicate of $\Box(a \wedge b)$ for every $k \geq 1$. We thereby demonstrate that not only do these definitions admit formulae with infinitely many prime implicates but they also allow seemingly irrelevant clauses to be counted as prime implicates. This gives us strong grounds for dismissing these definitions as much of the utility of prime implicates in applications comes from their ability to eliminate such irrelevant consequences.

Theorem 9. *The notions of prime implicates and prime implicants induced by definitions **D1** and **D2**, do not satisfy **Equivalence**. The notions of prime implicates and prime implicants induced by **D3a**, **D3b**, and **D5** falsify **Finiteness**.*

While the comparison in the last section suggested that **D5** was at least as suitable as **D4** as a definition of clauses and terms, the results of this section rule out **D5** as a suitable definition for prime implicates and prime implicants. In the remainder of the paper, we will concentrate our attention on the notions of prime implicates and prime implicants induced by definition **D4**, as these have been shown to be the most satisfactory generalizations of the propositional case.

5 Prime Implicate Generation and Recognition

In this section, we investigate the computational aspects of modal prime implicates. As we will only be considering the notion of prime implicates induced by definition **D4**, from now on we will use the words "clause", "term", and "prime implicate" to mean clause, term, and prime implicate with respect to definition **D4**.

We remark that we can focus without loss of generality on prime implicates since by **Implicant-Implicate Duality** (Theorem 8) any algorithm for generating or recognizing prime implicates can be easily adapted into an algorithm for generating or recognizing prime implicants.

5.1 Generating Prime Implicates

We start by considering the problem of generating the set of prime implicates of a given formula. In propositional logic, this task can take an exponential amount of space and time since the number of prime implicates is potentially exponential in the length of the formula. Since $\mathcal{K}$ includes all of propositional logic, the same must be true of $\mathcal{K}$.

Figure 2 presents the algorithm **GenPI** which computes the sets of prime implicates of a given formula. The algorithm makes use of the helper function **Dnf-4**(ϕ) which returns a set of satisfiable terms with respect to **D4** whose disjunction is equivalent to ϕ. The details of this function can be found in Appendix A.

Function GenPI(ϕ)
Input: a formula ϕ *Output*: a set of clauses

(1) If ϕ is unsatisfiable, return $\{\Diamond(a \wedge \neg a)\}$. Otherwise, set $\mathcal{T} =$ **Dnf-4(ϕ)**.
(2) For each $T \in \mathcal{T}$: let L_T be the set of propositional literals in T and let D_T be the set of formulae ζ such that $\Diamond\zeta$ is in T. If there are no literals of the form $\Box\psi$ in T, then set $\Delta(T) = L_T \cup \{\Diamond\zeta \mid \zeta \in D_T\}$. Otherwise, set $\Delta(T) =$

$$L_T \cup \{\Box\beta_T\} \cup \{\Diamond(\zeta \wedge \beta_T) \mid \zeta \in D_T\}$$

where β_T is the conjunction of formulae ψ such that $\Box\psi$ is in T.
(3) Set Candidates $= \{\bigvee_{T\in\mathcal{T}} \theta_T \mid \theta_T \in \Delta(T)\}$.
(4) For each $\lambda_j \in$ Candidates: remove λ_j from Candidates if $\lambda_k \models \lambda_j$ for some $\lambda_k \in$ Candidates with $k < j$, or if both $\lambda_j \models \lambda_k$ and $\lambda_k \not\models \lambda_j$ for $k > j$.
(5) Return Candidates.

Figure 2: Algorithm for prime implicate generation.

Our algorithm works as follows: in Step (1), we check whether ϕ is unsatisfiable, outputting a contradictory clause if this is the case. For satisfiable ϕ, we set $\mathcal{T}$ equal to a set of satisfiable terms whose disjunction is equivalent to ϕ. Because of **Distribution**, we know that every prime implicate of ϕ is equivalent to some disjunction of prime implicates of the terms in $\mathcal{T}$. In Step (2) we set $\Delta(T)$ equal to the propositional literals in T (L_T) plus the strongest $\Box$-literal implied by T ($\Box\beta_T$) plus the strongest $\Diamond$-literals implied by T ($\{\Diamond(\zeta \wedge \beta_T) \mid \zeta \in D_T\}$). It is not too hard to see

that every prime implicate of T must be equivalent to one of the elements in $\Delta(T)$. This means that in Step (3) we are guaranteed that every prime implicate of the input formula is equivalent to some candidate prime implicate in CANDIDATES. During the comparison phase in Step (4), non-prime candidates are eliminated, and exactly one prime implicate of each equivalence class will be retained.

Theorem 10. *The algorithm* **GenPI** *always terminates and outputs exactly the set of prime implicates of the input formula.*

By examining the prime implicates produced by the algorithm, we can put an upper bound on the length of the smallest representation of a prime implicate.

Theorem 11. *The length of the smallest representation of a prime implicate of a formula can be no more than singly exponential in the length of the formula.*

This upper bound is optimal as we can find formulae with exponentially large prime implicates:

Theorem 12. *The length of the smallest representation of a prime implicate of a formula can be exponential in the length of the formula.*

Similarly, by considering the number of candidate prime implicates constructed by our algorithm, we can place a bound on the maximal number of non-equivalent prime implicates a formula can possess.

Theorem 13. *The number of non-equivalent prime implicates of a formula is at most doubly exponential in the length of the formula.*

This bound can also be shown to be optimal.

Theorem 14. *The number of non-equivalent prime implicates of a formula may be doubly exponential in the length of the formula.*

We now illustrate the functioning of **GenPI** on an example:

Example 15. Set $\phi = a \wedge ((\Diamond(b \wedge c) \wedge \Diamond b) \vee (\Diamond b \wedge \Diamond(c \vee d) \wedge \Box e \wedge \Box f))$. As ϕ is satisfiable, we call the function **Dnf-4** on ϕ, and it returns the following two terms: $T_1 = a \wedge \Diamond(b \wedge c) \wedge \Diamond b$ and $T_2 = a \wedge \Diamond b \wedge \Diamond(c \vee d) \wedge \Box e \wedge \Box f$. Now $L_{T_1} = \{a\}$, $D_{T_1} = \{b \wedge c, b\}$, and there are no $\Box$-literals in T_1, so we get $\Delta(T_1) = \{a, \Diamond(b \wedge c), \Diamond b\}$. For T_2, we get $L_{T_2} = \{a\}$, $D_{T_2} = \{b, c \vee d\}$, and $\beta_{T_2} = e \wedge f$, giving us $\Delta(T_2) = \{a, \Box(e \wedge f), \Diamond(b \wedge e \wedge f), \Diamond((c \vee d) \wedge e \wedge f)\}$. The set CANDIDATES will contain all the different possible disjunctions of elements in $\Delta(T_1)$ with elements in $\Delta(T_2)$, of which there are 12: $a \vee a$, $a \vee \Box(e \wedge f)$, $a \vee \Diamond(b \wedge e \wedge f)$, $a \vee \Diamond((c \vee d) \wedge e \wedge f)$, $\Diamond(b \wedge c) \vee a$, $\Diamond(b \wedge c) \vee \Box(e \wedge f)$, $\Diamond(b \wedge c) \vee \Diamond(b \wedge e \wedge f)$, $\Diamond(b \wedge c) \vee \Diamond((c \vee d) \wedge e \wedge f)$, $\Diamond b \vee a$, $\Diamond b \vee \Box(e \wedge f)$, $\Diamond b \vee \Diamond(b \wedge e \wedge f)$, and $\Diamond b \vee \Diamond((c \vee d) \wedge e \wedge f)$. In Step 4, we will remove from CANDIDATES the clauses $a \vee \Box(e \wedge f)$, $a \vee \Diamond(b \wedge e \wedge f)$, $a \vee \Diamond((c \vee d) \wedge e \wedge f)$, $\Diamond(b \wedge c) \vee a$, and $\Diamond b \vee a$ since they are strictly weaker than $a \vee a$. We will also eliminate the clauses $\Diamond b \vee \Box(e \wedge f)$, $\Diamond b \vee \Diamond(b \wedge e \wedge f)$, and $\Diamond b \vee \Diamond((c \vee d) \wedge e \wedge f)$ since they are weaker than the clauses $\Diamond(b \wedge c) \vee \Box(e \wedge f)$, $\Diamond(b \wedge c) \vee \Diamond(b \wedge e \wedge f)$, $\Diamond(b \wedge c) \vee \Diamond((c \vee d) \wedge e \wedge f)$. There are no further clauses to remove, so the algorithm will return the four remaining clauses in CANDIDATES, which are $a \vee a$, $\Diamond(b \wedge c) \vee \Box(e \wedge f)$, $\Diamond(b \wedge c) \vee \Diamond(b \wedge e \wedge f)$, and $\Diamond(b \wedge c) \vee \Diamond((c \vee d) \wedge e \wedge f)$.

Our algorithm corresponds to the simplest possible implementation of the distribution property, and it is well-known that naïve implementations of the distribution property are already computationally infeasible for propositional logic. The principal source of inefficiency is the high number of clauses that are generated, so if we want to design a more efficient algorithm, we must find a way to generate fewer candidate clauses. There are a couple of different techniques that could be used. One very simple method which could yield a smaller number of clauses is to eliminate from $\Delta(T)$ those elements which are not prime implicates of T, thereby decreasing the cardinalities of the $\Delta(T)$ and hence of CANDIDATES. To do this, we simply test whether β_T is a tautology (and remove it if it is) and then compare the $\Diamond$-literals in $\Delta(T)$, discarding any weaker elements. If we apply this technique to Example 15, we would remove $\Diamond b$ from $\Delta(T_1)$, thereby reducing the cardinality of CANDIDATES from 12 to 8.

More substantial results could be achieved by using a technique developed in the framework of propositional logic (cf. [26]) which consists in calculating the prime implicates of T_1, then the prime implicates of $T_1 \vee T_2$, then those of $T_1 \vee T_2 \vee T_3$, and so on until we get the prime implicates of the full disjunction of terms. By interleaving comparison and construction, we can eliminate early on a partial clause that cannot give rise to prime implicates instead of producing all of the extensions of the partial clause and then deleting them one by one during the comparison phase. In our example, there were only two terms, but imagine that there was a third term T_3. Then by applying this technique, we would first produce the 4 prime implicates of $T_1 \vee T_2$ and then we would compare the $4|\Delta(T_3)|$ candidate clauses of $T_1 \vee T_2 \vee T_3$. Compare this with the current algorithm which generates and then compares $12|\Delta(T_3)|$ candidate clauses.

Given that the number of elements in CANDIDATES can be doubly exponential in $|\phi|$, cutting down on the length of the input to **GenPI** could yield significant savings. As input formulae are very often conjunctions of a number of sub-formulae, one idea would be to break conjunctions of formulae into their conjuncts, and then calculate the prime implicates of each of the conjuncts. Unfortunately, however, we cannot apply this method to every formula as the prime implicates of the conjuncts are not necessarily prime implicates of the full conjunction. One solution which was proposed in the context of approximation of description logic concepts (cf. [5], [32]) is to identify simple syntactic conditions that guarantee that we will get the same result if we break the formula into its conjuncts. For instance, one possible condition is that the conjuncts do not share any propositional variables. The formula ϕ in our example satisfies this condition since the variables in a and $((\Diamond(b \wedge c) \wedge \Diamond b) \vee (\Diamond b \wedge \Diamond(c \vee d) \wedge \Box e \wedge \Box f))$ are disjoint. By generating the prime implicates of the conjuncts separately, we can directly identify the prime implicate a, and we only have 6 candidate clauses of $((\Diamond(b \wedge c) \wedge \Diamond b) \vee (\Diamond b \wedge \Diamond(c \vee d) \wedge \Box e \wedge \Box f))$ to compare. If we also remove weaker elements from the $\Delta(T_i)$ as suggested above, we get only 3 candidate clauses for $((\Diamond(b \wedge c) \wedge \Diamond b) \vee (\Diamond b \wedge \Diamond(c \vee d) \wedge \Box e \wedge \Box f))$, all of which are prime implicates of ϕ.

The second source of inefficiency in our algorithm is the comparison phase in which we compare all candidate clauses one-by-one in order to single out the strongest ones. This comparison phase is simply unnecessary in propositional logic because we can test directly whether a clause is a prime implicate or not, without considering all the other potential prime implicates. Clearly, it would be desirable to be able to do something similar for prime implicates in $\mathcal{K}$, and in the next section, we study how we might do this.

5.2 Recognizing Prime Implicates

We now consider the problem of recognizing prime implicates, that is, the problem of deciding whether a clause λ is a prime implicate of a formula ϕ. In propositional logic, this problem is BH_2-complete [26], being as hard as both satisfiability and deduction. We simply test if the clause is in fact an implicate, and then verify that no stronger clauses are implicates. In $\mathcal{K}$, satisfiability and unsatisfiability are both PSPACE-complete, so we cannot hope to find a prime implicate recognition algorithm with a complexity of less than PSPACE.

Theorem 16. *Prime implicate recognition is* PSPACE*-hard.*

In order to obtain a first upper bound, we can exploit Corollary 11 which tells us that there exists a polynomial function f such that the length of the smallest representation of a prime implicate of a formula ϕ is bounded by $2^{f(|\phi|)}$. This leads to a simple procedure for determining if a clause λ is a prime implicate of a formula ϕ. We simply check for every clause λ' of length at most $2^{f(|\phi|)}$ whether λ' is an implicate of ϕ which implies λ but is not implied by λ. If this is the case, then λ is not a prime implicate (we have found a logically stronger implicate of ϕ), otherwise, there exists no stronger implicate, so λ is a prime implicate. It is not too hard to see that this algorithm can be carried out in exponential space, which gives us an EXPSPACE upper bound.

Theorem 17. *Prime implicate recognition is in* EXPSPACE*.*

Of course the problem with this approach is that it doesn't at all take into account the structure of λ, so we end up comparing a huge amount of irrelevant clauses, which is exactly what we were hoping to avoid. The algorithm that we propose later in this section avoids this problem by using the information in the input formula and clause in order to cut down on the number of clauses to test. The key to our algorithm is the following theorem which shows us how the general problem of prime implicate recognition can be reduced to the more specialized tasks of prime implicate recognition for propositional formulae, $\Box$-literals, and $\Diamond$-literals. To simplify the presentation of the theorem, we let $\Pi(\phi)$ refer to the set of prime implicates of ϕ, and we use the notation $\lambda \setminus \{l_1, ..., l_n\}$ to refer to the clause obtained by removing each of the literals l_i from λ. For example $(a \vee b \vee \Diamond c) \setminus \{a, \Diamond c\}$ refers to the clause b.

Theorem 18. *Let ϕ be a formula of $\mathcal{K}$, and let $\lambda = \gamma_1 \vee ... \vee \gamma_k \vee \Diamond\psi_1 \vee ... \vee \Diamond\psi_n \vee \Box\chi_1 \vee ... \vee \Box\chi_m$ be a non-tautologous clause such that (a) $\chi_i \equiv \chi_i \vee \psi_1 \vee ... \vee \psi_n$ for all i, and (b) there is no literal l in λ such that $\lambda \equiv \lambda \setminus \{l\}$. Then $\lambda \in \Pi(\phi)$ if and only if the following conditions hold:*

1. $\gamma_1 \vee ... \vee \gamma_k \in \Pi(\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\}))$
2. $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_n) \in \Pi(\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\}))$ *for every i*
3. $\Diamond(\psi_1 \vee ... \vee \psi_n) \in \Pi(\phi \wedge \neg(\lambda \setminus \{\Diamond\psi_1, ..., \Diamond\psi_n\}))$

We remark that the restriction to clauses for which $\chi_i \equiv \chi_i \vee \psi_1 \vee ... \vee \psi_m$ for all i and for which $\lambda \not\equiv \lambda \setminus \{l\}$ for all l is required. If we drop the first condition, then there are some non-prime implicates that satisfy all three conditions, and if we drop the second, there are prime

implicates which fail to satisfy one of the conditions[5]. These restrictions are without loss of generality however since every clause can be transformed into an equivalent clause satisfying them. For the first condition, we replace each $\Box\chi_i$ by $\Box(\chi_i \vee \psi_1 \vee ... \vee \psi_m)$, thereby transforming a clause $\gamma_1 \vee ... \vee \gamma_k \vee \Diamond\psi_1 \vee ...\Diamond\psi_m \vee \Box\chi_1 \vee ... \vee \Box\chi_n$ into the equivalent $\gamma_1 \vee ... \vee \gamma_k \vee \Diamond\psi_1 \vee ...\Diamond\psi_m \vee \Box(\chi_1 \vee \psi_1 \vee ... \vee \psi_m) \vee ... \vee \Box(\chi_n \vee \psi_1 \vee ... \vee \psi_m)$. Then to make the clause satisfy the second condition, we simply remove from λ those literals for which $\lambda \equiv \lambda \setminus \{l\}$ until no such literal remains.

Theorem 18 shows us how prime implicate recognition can be split into three more specialized sub-tasks, but it does not tell us how to carry out these tasks. Thus, in order to turn this theorem into an algorithm for prime implicate recognition, we need to figure out how to test whether a propositional clause, a $\Box$-literal, or a $\Diamond$-literal is a prime implicate of a formula.

Determining whether a propositional clause is a prime implicate of a formula in $\mathcal{K}$ is conceptually no more difficult than determining whether a propositional clause is a prime implicate of a propositional formula. We first ensure that the clause is an implicate of the formula and then make sure that all literals appearing in the clause are necessary.

Theorem 19. *Let ϕ be a formula of $\mathcal{K}$, and let γ be a non-tautologous propositional clause such that $\phi \models \gamma$ and such that there is no literal l in γ such that $\gamma \equiv \gamma \setminus \{l\}$. Then $\gamma \in \Pi(\phi)$ if and only if $\phi \not\models \gamma \setminus \{l\}$ for all l in γ.*

We now move on to the problem of deciding whether a clause of the form $\Box\chi$ is a prime implicate of a formula ϕ. We remark that if $\Box\chi$ is implied by ϕ, then it must also be implied by each of the terms $T_i \in$**Dnf-4**(ϕ). But if $T_i \models \Box\chi$, then by Theorem 1, it must be the case that the conjunction of the $\Box$-literals in T_i implies $\Box\chi$. This means that the formula $\Box\beta_1 \vee ... \vee \Box\beta_n$ (where β_i is the conjunction of the formulae ζ such that $\Box\zeta$ is in T_i) is an implicate of ϕ which implies $\Box\chi$, and moreover it is the strongest such implicate. It follows then that $\Box\chi$ is a prime implicate of ϕ just in the case that $\Box\chi \models \Box\beta_1 \vee ... \vee \Box\beta_n$, which is true if and only if $\chi \models \beta_i$ for some i (by Theorem 1). Thus, by comparing the formula χ with the formulae β_i associated with the terms of ϕ, we can decide whether or not $\Box\chi$ is a prime implicate of ϕ.

Theorem 20. *Let ϕ be a formula of $\mathcal{K}$, and let $\lambda = \Box\chi$ be a non-tautologous clause such that $\phi \models \lambda$. Then $\lambda \in \Pi(\phi)$ if and only if there exists some term $T \in$**Dnf-4**(ϕ) such that $\chi \models \beta_T$, where β_T is the conjunction of formulae ψ such that $\Box\psi$ is in T.*

Finally let us turn to the problem of deciding whether a clause $\Diamond\psi$ is a prime implicate of a formula ϕ. Now we know by **Covering** that if $\Diamond\psi$ is an implicate of ϕ, then there must be some prime implicate π of ϕ which implies $\Diamond\psi$. It follows from Theorem 2 that π must be a disjunction of $\Diamond$-literals, and from Theorem 10 that π is equivalent to a disjunction $\bigvee_{T \in \textbf{Dnf-4}(\phi)} \Diamond d_T$ where $\Diamond d_T$ is an element of $\Delta(T)$ for every T (refer back to Figure 2 for the definition of $\Delta(T)$). According to Definition 6, $\Diamond\psi$ is a prime implicate of ϕ just in the case that $\Diamond\psi \models \bigvee_{T \in \textbf{Dnf-4}(\phi)} \Diamond d_T$, or equivalently when $\psi \models \bigvee_{T \in \textbf{Dnf-4}(\phi)} d_T$. Testing directly whether $\psi \models \bigvee_{T \in \textbf{Dnf-4}(\phi)} d_T$ could take exponential space in the worst case since there may be exponentially many terms in **Dnf-4**(ϕ).

[5]For the first condition, consider the formula $\phi = \Diamond(a \wedge b \wedge c) \vee \Box a$ and the clause $\lambda = \Diamond(a \wedge b) \vee \Box(a \wedge \neg b)$. It can be easily shown that λ is an implicate of ϕ, but λ is not a prime implicate of ϕ since there exist stronger implicates (e.g. ϕ itself). Nonetheless, it can be verified that both $\Box(a \wedge \neg b \wedge \neg(a \wedge b)) \in \Pi(\phi \wedge \neg(\lambda \setminus \{\Box(a \wedge \neg b)\}))$ and $\Diamond(a \wedge b) \in \Pi(\phi \wedge \neg(\lambda \setminus \{\Diamond(a \wedge b)\}))$. For the second condition, consider the formula a and the clause $a \vee a$. We have $a \vee a \notin \Pi(a \wedge \neg a)$ even though $a \vee a$ is a prime implicate of a.

Luckily, however, we can get around this problem by exploiting the structure of the formula π. We remark that each d_T is a conjunction of formulae ζ such that $\Box\zeta$ or $\Diamond\zeta$ appears in $\mathbf{Nnf}(\phi)$ outside the scope of modal operators – we will denote by $\mathcal{X}$ the set of formulae ζ satisfying this condition. It follows that if $\psi \not\models \bigvee_{T\in\mathbf{Dnf\text{-}4}(\phi)} d_T$ then we can find a subset $S \subseteq \mathcal{X}$ such that (a) $\psi \not\models \bigvee_{\lambda\in S} \lambda$ and (b) every d_T has at least one conjunct from the set S. This observation yields the algorithm **Test$\Diamond$PI** given in Figure 3. The basic idea behind the algorithm is to try out each of the different clauses built from the set $\mathcal{X}$ and to see if there is some clause which is not implied by $\Diamond\psi$ but is implied by some implicate of ϕ which implies $\Diamond\psi$. If we find such a clause, then we know that $\Diamond\psi$ is not a prime implicate, and if no such clause exists, this means that $\Diamond\psi$ implies the prime implicate $\bigvee_{T\in\mathbf{Dnf\text{-}4}(\phi)} \Diamond d_T$, so $\Diamond\psi$ must be a prime implicate of ϕ. The algorithm can be shown to run in polynomial time since there can be at most $|\phi|$ elements in $\mathcal{X}$, and we can consider the terms in $T \in \mathbf{Dnf\text{-}4}(\phi)$ one at a time.

Function Test$\Diamond$PI($\Diamond\psi$, ϕ)
Input: a clause $\Diamond\psi$ and a formula ϕ such that $\phi \models \Diamond\psi$ *Output:* **yes** or **no**

(0) If $\phi \models \bot$, return **yes** if $\psi \models \bot$ and **no** otherwise.
(1) Set $\mathcal{X}$ equal to the set of formulae ζ such that $\Box\zeta$ or $\Diamond\zeta$ appears in $\mathbf{Nnf}(\phi)$ outside the scope of modal operators.
(2) For each $S \subseteq \mathcal{X}$, test whether the following two conditions hold:
  (a) $\psi \not\models \bigvee_{\lambda\in S} \lambda$
  (b) for each $T_i \in \mathbf{Dnf\text{-}4}(\phi)$, there exists conjuncts $\Diamond\eta_i, \Box\mu_{i,1}, ..., \Box\mu_{i,k(i)}$ of T_i such that:
    (i) $\{\eta_i, \mu_{i,1}, ..., \mu_{i,k(i)}\} \cap S \neq \emptyset$
    (ii) $\Diamond(\eta_i \wedge \mu_{i,1} \wedge ... \wedge \mu_{i,k(i)}) \models \Diamond\psi$
  Return **no** if some S satisfies these conditions, and **yes** otherwise.

Figure 3: Algorithm for identifying prime implicates of the form $\Diamond\psi$. The helper functions **Nnf** and **Dnf-4** can be found in Appendix A.

Theorem 21. *Let ϕ be a formula, and let $\Diamond\psi$ be an implicate of ϕ. Then the algorithm* **Test$\Diamond$PI** *returns **yes** on input ($\Diamond\psi$, ϕ) if and only if $\Diamond\psi$ is a prime implicate of ϕ.*

Theorem 22. *The algorithm* **Test$\Diamond$PI** *runs in polynomial space.*

We now present two examples which illustrate the functioning of the algorithm **Test$\Diamond$PI**.

Example 23. We use **Test$\Diamond$PI** to test whether the clause $\lambda = \Diamond(a \wedge b)$ is a prime implicate of $\phi = a \wedge (\Box(b \wedge c) \vee \Box(e \vee f)) \wedge \Diamond(a \wedge b)$.

As ϕ is satisfiable, we go directly to Step (1). In this step, we set $\mathcal{X}$ equal to the set of formulae ζ such that $\Box\zeta$ or $\Diamond\zeta$ appears in $\mathbf{Nnf}(\phi)$ outside the scope of modal operators. In our case, we set $\mathcal{X} = \{b \wedge c, e \vee f, a \wedge b\}$ since $\phi = \mathbf{Nnf}(\phi)$ and $b \wedge c$, $e \vee f$, and $a \wedge b$ are the only formulae satisfying the requirements. In Step (2), we examine each of the different subsets of $\mathcal{X}$ to determine whether they satisfy conditions (a) and (b). In particular, we consider the subset $S = \{b \wedge c, e \vee f\}$. We remark that this subset satisfies condition (a) since $a \wedge b \not\models (b \wedge c) \vee (e \vee f)$. In order to check condition (b), we first call the function **Dnf-4** on ϕ which returns the two

terms $T_1 = a \wedge \Box(b \wedge c) \wedge \Diamond(a \wedge b)$ and $T_2 = a \wedge \Box(e \vee f) \wedge \Diamond(a \wedge b)$. We notice that the conjuncts $\Diamond(a \wedge b)$ and $\Box(b \wedge c)$ of T_1 satisfy conditions (i) and (ii) since $b \wedge c \in S$ and $\Diamond(a \wedge b \wedge (b \wedge c)) \models \lambda$. We then notice that the conjuncts $\Diamond(a \wedge b)$ and $\Box(e \vee f)$ of T_2 also satisfy conditions (i) and (ii) since $e \vee f \in S$ and $\Diamond(a \wedge b \wedge (e \vee f)) \models \lambda$. That means that we have found a subset S of $\mathcal{X}$ which satisfies conditions (a) and (b), so the algorithm returns **no**. This is the correct output since $\Diamond(a \wedge b \wedge ((b \wedge c) \vee (e \vee f)))$ is an implicate of ϕ which is strictly stronger than λ.

Example 24. We use **Test$\Diamond$PI** to test whether the clause $\lambda = \Diamond(a \wedge b \wedge c)$ is a prime implicate of $\phi = a \wedge (\Box(b \wedge c) \vee \Box(e \vee f)) \wedge \Diamond(a \wedge b) \wedge \neg\Box(e \vee f \vee (a \wedge b \wedge c))$.

We skip Step (0) since ϕ is satisfiable. In Step (1), we set $\mathcal{X} = \{b \wedge c, e \vee f, a \wedge b, \neg e \wedge \neg f \wedge (\neg a \vee \neg b \vee \neg c))\}$ since $\mathbf{Nnf}(\phi) = a \wedge (\Box(b \wedge c) \vee \Box(e \vee f)) \wedge \Diamond(a \wedge b) \wedge \Diamond(\neg e \wedge \neg f \wedge (\neg a \vee \neg b \vee \neg c))$. In Step (2), we check whether there is some subset of $\mathcal{X}$ satisfying conditions (a) and (b). We claim that there is no such subset. To see why, notice that $a \wedge \Box(b \wedge c) \wedge \Diamond(a \wedge b) \wedge \Diamond(\neg e \wedge \neg f \wedge (\neg a \vee \neg b \vee \neg c))$ is the only term in **Dnf-4**(ϕ). Moreover, there is only one set of conjuncts of this term which implies $\Diamond(a \wedge b \wedge c)$, namely $\{\Diamond(a \wedge b), \Box(b \wedge c)\}$. But that means that S must contain either $a \wedge b$ or $b \wedge c$ in order to satisfy condition (b)(i). As $a \wedge b \wedge c$ implies both $a \wedge b$ and $b \wedge c$, we are guaranteed that $a \wedge b \wedge c$ will imply the disjunction of elements in S, thereby falsifying condition (a). It follows that there is no subset of $\mathcal{X}$ satisfying the necessary conditions, so the algorithm returns **yes** at the end of Step (3), which is the desired result.

Function TestPI(λ, ϕ)
Input: a clause λ and a formula ϕ *Output:* **yes** or **no**

(1) If $\phi \not\models \lambda$, return **no**.
(2) If $\phi \models \bot$, then return **yes** if $\lambda \models \bot$ and **no** if not. If $\models \lambda$, then return **yes** if $\models \phi$ and **no** otherwise.
(3) For each l_i in $\lambda = l_1 \vee ... \vee l_n$, test if $\lambda \setminus \{l_i\} \equiv \lambda$, and if so, remove l_i from λ. Let $\mathcal{D} = \{\Diamond\psi_1, ..., \Diamond\psi_m\}$ be the set of $\Diamond$-literals in λ. If $\mathcal{D}$ is non-empty, replace each literal $\Box\chi$ in λ by the literal $\Box(\chi \vee \psi_1 \vee ... \vee \psi_m)$.
(4) Let $\mathcal{P}$ be the set of propositional literals which are disjuncts of λ. For each $l \in \mathcal{P}$, check whether $\phi \models \lambda \setminus \{l\}$, and return **no** if so.
(5) Let $\mathcal{B}$ be the set of $\Box$-literals appearing as disjuncts in λ. Check for each $\Box\zeta$ in $\mathcal{B}$ whether there is some T in **Dnf-4**$(\phi \wedge \neg(\lambda \setminus \{\Box\zeta\}))$ for which the formula $\Box(\zeta \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_k)$ implies the conjunction of $\Box$-literals in T, and return **no** if not.
(6) If $\mathcal{D}$ is empty, return **yes**, otherwise return **Test$\Diamond$PI**$(\Diamond(\bigvee_{i=1}^{m} \psi_i), \phi \wedge \neg(\lambda \setminus \mathcal{D}))$.

Figure 4: Algorithm for identifying prime implicates. Refer to Appendix A for the helper function **Dnf-4**.

In Figure 4, we present our algorithm for testing whether a clause λ is a prime implicate of a formula ϕ. The first two steps of the algorithm treat the limit cases where λ is not an implicate or where one or both of ϕ and λ is a tautology or contradiction. In Step (3), we apply equivalence-preserving transformations to λ to make it satisfy the requirements of Theorem 18.

Then in Steps (4), (5), and (6) we use the procedures from Theorems 19, 20, and 21 to test whether the three conditions in Theorem 18 are verified. If the three tests succeed, then by Theorem 18, the clause is a prime implicate, so we return **yes**. If some test fails, we return **no** as the clause has been shown not to be a prime implicate.

Theorem 25. *The algorithm* **TestPI** *always terminates, and it returns* ***yes*** *on input* (λ, ϕ) *if and only if* λ *is a prime implicate of* ϕ.

We demonstrate the functioning of **TestPI** on an example.

Example 26. We use **TestPI** to test if the clauses $\lambda_1 = b$, $\lambda_2 = \Box b \vee \Box(e \vee f)$, $\lambda_3 = a \vee \Diamond a$, $\lambda_4 = \Diamond(a \wedge b)$, and $\lambda_5 = \Diamond(a \wedge b \wedge c) \vee \Diamond(a \wedge b \wedge c \wedge f) \vee \Box(e \vee f)$ are prime implicates of $\phi = a \wedge (\Box(b \wedge c) \vee \Box(e \vee f)) \wedge \Diamond(a \wedge b)$.
– For λ_1, we output **no** in Step (1) since $\phi \not\models \lambda_1$.
– For λ_2, we skip Steps (1) and (2) since $\lambda \models \lambda_2$ and neither $\phi \models \bot$ nor $\models \lambda_2$. In Step (3), we make no changes to λ_2 since it contains no redundant literals nor any $\Diamond$-literals. We skip Step (4) since λ_2 has no propositional disjuncts. In Step (5), we return **no** since **Dnf-4**$(\phi \wedge \neg(\lambda_2 \backslash \{\Box b\})) = \{a \wedge \Box(b \wedge c) \wedge \Diamond(a \wedge b) \wedge \Diamond(\neg e \wedge \neg f)\}$ and $\Box b \not\equiv \Box(b \wedge c)$.
– For λ_3, we proceed directly to Step (3) since $\lambda \models \lambda_3$, $\phi \not\models \bot$, and $\not\models \lambda_3$. No modifications are made to λ_3 in Step (3) as it does not contain any redundant literals or $\Box$-literals. In Step (4), we test whether or not $\phi \models \lambda_3 \backslash \{a\}$. As $\phi \not\models \lambda_3 \backslash \{a\}$, we proceed on to Step (5), and then directly on to Step (6) since λ_3 contains no $\Box$-literals. In Step (6), we call **Test$\Diamond$PI**$(\Diamond a, \phi \wedge \neg(\lambda_3 \setminus \{\Diamond a\}))$, which outputs **no** since $\phi \wedge \neg(\lambda_3 \setminus \{\Diamond a\}) \models \bot$ and $a \not\models \bot$.
– For λ_4, Steps (1)-(5) are all inapplicable, so we skip directly to Step (6). In this step, we call **Test$\Diamond$PI** with as input the clause $\Diamond(a \wedge b)$ and the formula $\phi \wedge \neg(\lambda_4 \setminus \{\Diamond(a \wedge b)\}) = \phi$. We have already seen in Example 23 above that **Test$\Diamond$PI** returns **no** on this input, which means that **TestPI** also returns **no**.
– For λ_5, we proceed directly to Step (3), where we delete the redundant literal $\Diamond(a \wedge b \wedge c \wedge f)$ and then modify the literal $\Box(e \vee f)$. At the end of this step, we have $\lambda_5 = \Diamond(a \wedge b \wedge c) \vee \Box((e \vee f) \vee (a \wedge b \wedge c))$. Step (4) is not applicable since there are no propositional disjuncts in λ_5. In Step (5), we continue since **Dnf-4**$(\phi \wedge \neg(\lambda_5 \backslash \{\Box((e \vee f \vee (a \wedge b \wedge c))\})) = \{a \wedge \Box(e \vee f) \wedge \Diamond(a \wedge b) \wedge \Box(\neg a \vee \neg b \vee \neg c)\}$, and $\Box(((e \vee f \vee (a \wedge b \wedge c)) \wedge (\neg a \vee \neg b \vee \neg c)) \equiv \Box(e \vee f) \wedge \Box(\neg a \vee \neg b \vee \neg c)$. In Step (6), we return **yes** since we call **Test$\Diamond$PI** on input $(\Diamond(a \wedge b \wedge c), \phi \wedge \neg(\lambda_5 \setminus \{\Diamond(a \wedge b \wedge c)\}))$, and we have previously shown in Example 24 that **Test$\Diamond$PI** returns **yes** on this input.

We show in the appendix that the algorithm **TestPI** runs in polynomial space. As we have already shown that **TestPI** decides prime implicate recognition, it follows that this problem is in PSPACE:

Theorem 27. *Prime implicate recognition is in* PSPACE.

By putting together Theorems 16 and 27, we obtain a tight complexity bound for the prime implicate recognition problem.

Corollary 28. *Prime implicate recognition is* PSPACE*-complete.*

6 Conclusion and Future Work

The first contribution of this work is a detailed comparison of several different possible definitions of clauses, terms, prime implicates, and prime implicants for the modal logic $\mathcal{K}$. The results of this investigation were largely positive: although we have shown that no perfect definition exists, we did exhibit a very simple definition which satisfies most of the desirable properties of the propositional case. The second contribution of our work is a thorough investigation of the computational aspects of the selected definition. To this end, we presented a sound and complete algorithm for generating prime implicates, as well as a number of optimizations to improve the efficiency of the algorithm. An examination of the structure of the prime implicates generated by our algorithm allowed us to place bounds on the length of prime implicates and on the number of prime implicates a formula can possess. We provided a decision procedure for recognizing prime implicates, and we proved the prime implicate recognition problem to be PSPACE-complete, which is the lowest complexity that could reasonably be expected. Although the focus of the paper was on the logic $\mathcal{K}$, all of our results are easily lifted to multi-modal $\mathcal{K}$ and to concept expressions in the well-known description logic $\mathcal{ALC}$.

As was mentioned in the introduction, one of the principal applications of prime implicants in propositional logic is to the area of abduction and diagnosis, where prime implicants play the role of abductive explanations. The results of our paper can be directly applied to the problem of abduction in $\mathcal{K}$: our notion of prime implicants can be used as a definition of abductive explanations in $\mathcal{K}$, and our prime implicate generation algorithm provides a means of producing all of the abductive explanations to a given abduction problem. Moreover, because the notion of term underlying our definition of abductive explanations is more expressive than that used in [8], we are able to find explanations which are overlooked by Cialdea Mayer & Pirri's method. For instance, if we look for an explanation of the observation c given the background information $\Box(a \vee b) \rightarrow c$, we obtain $\Box(a \vee b)$, whereas their framework yields $\Box a$ and $\Box b$. This is an argument in favor of our approach since generally in abduction one is looking to find the *weakest* conditions guaranteeing the truth of the observation given the background information.

The other domain of application which was cited in the introduction was the development of tools for querying modal/description logic knowledge bases. As in general a user is interested in generating some but not all of the consequences of a knowledge base, a necessary prerequisite for the development of such tools is the introduction of more refined notions of prime implicates and algorithms for generating them. There are a couple different variants on the notion of prime implicate which we are interested in looking at in future work. The first is the notion of Φ-prime implicates which are defined to be the strongest new consequences of Φ when expanded by a formula. It is this notion of prime implicate which could be used by a knowledge engineer to examine the effects of adding a new piece of information to a knowledge base. We could also define a new variety of prime implicate which groups together all of the prime implicates of a formula which mention non-trivially the propositional variables in a specified set. This type of prime implicate would prove useful to anyone who wanted to find out what information a knowledge base contained about a particular topic of interest, like for instance our example user who wants to find out what the knowledge base says about professors. Another kind of prime implicate which we are interested in studying is the notion of prime implicate over a sub-

language. This type of prime implicate could prove useful if we were interested in discovering all of the relationships holding between a given subset of propositional variables. We expect that the algorithms that we have constructed for the standard notion of prime implicates will provide a useful starting point in the development of algorithms for these new varieties of prime implicates.

In this paper, we have studied prime implicates with respect to the local consequence relation, so it is natural to wonder how things would be different if we were to define prime implicates with respect to the global consequence relation. This question is particularly interesting given that global consequence is the type of consequence used in description logic ontologies. Unfortunately, our preliminary investigations suggest that defining and generating prime implicates with respect to the global consequence relation will likely prove more difficult than for the local consequence relation. For one thing, if we use a definition of clause which is reasonably expressive, then the notion of prime implicates we obtain does not satisfy **Covering** since we can construct infinite sequences of stronger and stronger implicates. Take for instance the formula $\neg a \vee \Diamond a$ which implies each of the formulae in the infinite sequence $\neg a \vee \Diamond a$, $\neg a \vee \Diamond(a \wedge \Diamond a)$, $\neg a \vee \Diamond(a \wedge \Diamond(a \wedge \Diamond a))$, ... This is a familiar situation for description logic practitioners since these infinite sequences are responsible for the inexistence of most specific concepts in many common DLs (cf. [19]) and the lack of uniform interpolation for $\mathcal{ALC}$ TBoxes [17]. The standard solution to this problem is to simply place a bound on the depth of formulae to be considered, effectively blocking these problematic infinite sequences. This will not allow us to regain **Covering**, but it will give us a weaker version of this property, which should be sufficient for most applications. The development of generation algorithms for the global consequence relation may also prove challenging, since it is unclear at this point whether we will be able to draw inspiration from pre-existing methods or whether we will have to start from scratch. Despite these potential difficulties, we feel that this subject is worth exploring since it could contribute to the development of more flexible ways of accessing information in description logic ontologies.

Once we will have completed our investigation of prime implicates/implicants in $\mathcal{K}$, we plan to move on to study these notions in other popular modal and description logics, beginning with modal logics of knowledge and belief and expressive description logics used for the semantic web. We are confident that the experience gained from our investigation of $\mathcal{K}$ will prove very helpful in the exploration of other modal and description logics.

Appendix A. Helper Functions

The function **Nnf** in Figure 5 is well-known in the literature. We recall a few of its properties:

Lemma A.1 *The output of* **Nnf**(ϕ) *is a formula in negation normal form which is equivalent to* ϕ*. It has depth* $\delta(\phi)$ *and contains only those propositional variables appearing in* ϕ*. The output of* **Nnf**(ϕ) *has length no greater than* $2|\phi|$*.*

We now highlight some properties of the function **Dnf-4** presented in Figure 5:

Lemma A.2 Dnf-4 *always terminates. If the input is a satisfiable formula* ϕ*, the set returned by* **Dnf-4**(ϕ) *is non-empty, and the disjunction of the formulae in this set is a disjunction of*

Function Nnf(ϕ): returns a formula in NNF which is equivalent to ϕ
Case 1: $\phi = a$ or $\phi = \neg a$. Return ϕ.
Case 2: $\phi = \phi_1 \star \phi_2$, where $\star$ is $\vee$ or $\wedge$. Return $\mathbf{Nnf}(\phi_1) \star \mathbf{Nnf}(\phi_2)$.
Case 3: $\phi = \triangle\psi$ where $\triangle$ is $\Box$ or $\Diamond$. Return $\triangle\mathbf{Nnf}(\psi)$.
Case 4a: $\phi = \neg\neg\psi$. Return $\mathbf{Nnf}(\psi)$.
Case 4b: $\phi = \neg(\phi_1 \vee \phi_2)$. Return $\mathbf{Nnf}(\neg\phi_1) \wedge \mathbf{Nnf}(\neg\phi_2)$.
Case 4c: $\phi = \neg(\phi_1 \wedge \phi_2)$. Return $\mathbf{Nnf}(\neg\phi_1) \vee \mathbf{Nnf}(\neg\phi_2)$.
Case 4d: $\phi = \neg\Box\psi$. Return $\Diamond\mathbf{Nnf}(\neg\psi)$.
Case 4e: $\phi = \neg\Diamond\psi$. Return $\Box\mathbf{Nnf}(\neg\psi)$.

Function Dnf-4(ϕ): returns a set of terms with respect to **D4** whose disjunction is equivalent to ϕ
(1) Set $\Sigma = \{\{\mathbf{Nnf}(\phi)\}\}$.
(2) Apply the following rules until no rule is applicable:
$\wedge$**-rule** If $S \in \Sigma$ and $S = \{\psi \wedge \zeta\} \cup S'$, set $\Sigma = \Sigma \setminus S \cup \{S' \cup \{\psi\} \cup \{\zeta\}\}$
$\vee$**-rule** If $S \in \Sigma$ and $S = \{\psi \vee \zeta\} \cup S'$, set $\Sigma = \Sigma \setminus S \cup \{S' \cup \{\psi\}\} \cup \{S' \cup \{\zeta\}\}$
(3) Return $\{\bigwedge_{\sigma \in S} \sigma \mid S \in \Sigma \text{ and } S \text{ is consistent}\}$

Figure 5: Helper functions for the prime implicate generation and recognition algorithms presented in Figures 2, 3, and 4.

*terms with respect to **D4** which is equivalent to ϕ.*

Proof. The lemma is a consequence of the soundness and termination of standard tableaux algorithms, but we prove it here for completeness.

To prove termination, we prove that there can be at most $2^{|\phi|} - 1$ executions of rules in Step 2. To do so, we show by induction that if a set S contains k occurrences of $\wedge$ or $\vee$ outside the scope of the modal operators, then there can be at most $2^k - 1$ applications of the rules to $\{S\}$. Clearly this holds when $k = 0$ because in this case no rule is applicable. Suppose then that the statement holds for $k \leq n$, and let S be some set with $n + 1$ occurrences of $\wedge$ or $\vee$ outside the modal operators. If we apply the $\wedge$-rule to S, we obtain a new set S' which contains only n occurrences of $\wedge$ or $\vee$ outside the scope of the modal operators. By assumption, there can be at most $2^n - 1$ rule applications to $\{S'\}$, which yields a total of at most 2^n rule applications to $\{S\}$, which is no greater than $2^{n+1} - 1$. Now suppose instead that the $\vee$-rule is applied to $\{S\}$, yielding two new sets S_1 and S_2 each having at most n occurrences of $\wedge$ or $\vee$ outside the modal operators. By the induction hypothesis, we can only apply $2^n - 1$ rules to each of S_1 and S_2. As **Dnf-4** treats different elements in Σ separately, the number of rules that can be applied to $\{S_1, S_2\}$ is simply the sum of the number of rules applied to S_1 and the number of rules applied to S_2. Thus, we find that we can apply no more than $1 + 2 * (2^n - 1) = 2^{n+1} - 1$ rules to $\{S\}$. As there can never be more than $|\phi|$ occurrences of $\wedge$ or $\vee$ in ϕ, and the transformation to NNF preserves the number of $\wedge$ or $\vee$ symbols, there can be at most $2^{|\phi|} - 1$ executions of rules in Step 2. This is enough to ensure termination of **Dnf-4** since the transformation to NNF in Step 1 clearly terminates.

We next show by induction that the disjunction of the conjunctions of formulae in the sets S in Σ is equivalent to ϕ at every moment in the execution of the algorithm. We first remark that

the statement is true initially because we start off with a single disjunct $\textsc{Nnf}(\phi)$, and we know that $\textsc{Nnf}(\phi)$ is equivalent to ϕ. Moreover, we know that if the disjunction of the conjunctions of formulae in the sets in Σ is equivalent to ϕ before the execution of the $\wedge$-rule, then the same holds after the execution of the $\wedge$-rule since we have replaced the conjunction of formulae in $\{\psi \wedge \zeta\} \cup S'$ by the conjunction of formulae in the equivalent $\{\psi\} \cup \{\zeta\} \cup S'$. Similarly, if the statement holds before executing the $\vee$-rule, it will also hold after the execution since $(\bigwedge_{\sigma \in S'} \sigma) \wedge (\phi \vee \zeta) \equiv ((\bigwedge_{\sigma \in S'} \sigma) \wedge \phi) \vee ((\bigwedge_{\sigma \in S'} \sigma) \wedge \zeta)$. It follows then that the disjunction of the conjunctions of formulae in the sets S in Σ is always equivalent to ϕ.

Suppose now that ϕ is satisfiable. Then it must also be the case that the disjunction of the conjunctions of formulae in the sets in Σ at the end of Step 2 is also satisfiable. But that means that there must be at least one $S \in \Sigma$ which is satisfiable, so there will be at least one formula $\bigwedge_{\sigma \in S} \sigma$ which is returned by the algorithm. Moreover, since ϕ is equivalent to the disjunction of the conjunctions of formulae in the sets in Σ at the end of Step 2, ϕ must also be equivalent to the the disjunction of the conjunctions of formulae in the *satisfiable* sets in Σ at the end of Step 2. It follows that the disjunction of the formulae outputted by **Dnf-4**(ϕ) is equivalent to ϕ.

To complete the proof, we need to show that the disjunction of the formulae outputted by **Dnf-4**(ϕ) is a disjunction of terms with respect to **D4**. Now every element in every set of Σ at the end of Step 2 must be in NNF since the first element in Σ is in NNF and the operations performed in Step 2 do not add any negation symbols. Moreover, when we exit Step 2, there can be no element in any set of Σ of the form $\phi_1 \wedge \phi_2$ or $\phi_1 \vee \phi_2$, which means that every set in Σ must contain only literals with respect to **D4**. Since a conjunction of literals with respect to **D4** is a term with respect to **D4**, all of the formulae returned by **Dnf-4** must be terms, hence the disjunction of formulae in **Dnf-4**(ϕ) is a disjunction of terms with respect to **D4**. □

Lemma A.3 *There are at most $2^{|\phi|}$ terms in* **Dnf-4**(ϕ)*. Each of the terms in* **Dnf-4**(ϕ) *has length at most $2|\phi|$, depth at most $\delta(\phi)$, and contains only those propositional letters appearing in ϕ.*

Proof. We remark that each application of a rule in Step 2 increases by at most one the number of elements in Σ. As we begin with a single element in Σ, and we have shown in the proof of Lemma A.2 that there can be at most $2^{|\phi|} - 1$ rule applications, it follows that there can be at most $2^{|\phi|}$ elements in Σ at the end of Step 2, and hence at most $2^{|\phi|}$ terms in **Dnf-4**(ϕ).

Let us define the length of a set of formulae to be the length of the conjunction of its elements. Initially, the unique set in Σ has length at most $2|\phi|$, since by Lemma A.1 the length of $\textsc{Nnf}(\phi)$ is no more than double that of ϕ. An application of the $\wedge$-rule to a set S gives a set of the same length. An application of the $\vee$-rule to a set gives two new sets, each having a smaller length than its parent. Thus, at the end of Step 2, all sets in Σ have length at most $2|\phi|$, which means that the terms in **Dnf-4**(ϕ) also have length at most $2|\phi|$.

The conjuncts of the terms in **Dnf-4**(ϕ) are all sub-formulae of **Nnf**(ϕ). By Lemma A.1, **Nnf**(ϕ) has precisely the same depth and propositional letters as ϕ. It follows then that the terms in **Dnf-4**(ϕ) have depth at most $\delta(\phi)$ and contain only those propositional symbols appearing in ϕ. □

Appendix B. Proofs

Theorem 1 *Let γ, ψ, ψ_1, ..., ψ_m, χ, χ_1, ..., χ_n be formulae in $\mathcal{K}$, and let γ be a propositional formula. Then*

1. $\psi \models \chi \Leftrightarrow \models \neg\psi \vee \chi \Leftrightarrow \psi \wedge \neg\chi \models \bot$
2. $\psi \models \chi \Leftrightarrow \Diamond\psi \models \Diamond\chi \Leftrightarrow \Box\psi \models \Box\chi$
3. $\gamma \wedge \Diamond\psi_1 \wedge ... \wedge \Diamond\psi_m \wedge \Box\chi_1 \wedge ... \wedge \Box\chi_n \models \bot \Leftrightarrow (\gamma \models \bot$ *or* $\psi_i \wedge \chi_1 \wedge ... \wedge \chi_n \models \bot$ *for some* $i)$
4. $\models \gamma \vee \Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee \Box\chi_1 \vee ... \vee \Box\chi_n \Leftrightarrow (\models \gamma$ *or* $\models \psi_1 \vee ... \vee \psi_m \vee \chi_i$ *for some* $i)$
5. $\Box\chi \models \Box\chi_1 \vee ... \vee \Box\chi_n \Leftrightarrow \chi \models \chi_i$ *for some* i
6. $\Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee \Box\chi_1 \vee ... \vee \Box\chi_n$
   $\equiv \Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee \Box(\chi_1 \vee \psi_1 \vee ... \vee \psi_m) \vee ... \vee \Box(\chi_n \vee \psi_1 \vee ... \vee \psi_m)$

Proof. The first statement is a well-known property of local consequence, but we prove it here for completeness:

$$\begin{aligned}
\psi \models \chi \quad &\Leftrightarrow \quad M, w \models \psi \text{ implies } M, w \models \chi \text{ for all } M, w \\
&\Leftrightarrow \quad M, w \not\models \psi \text{ or } M, w \models \chi \text{ for all } M, w \\
&\Leftrightarrow \quad M, w \models \neg\psi \text{ or } M, w \models \chi \text{ for all } M, w \\
&\Leftrightarrow \quad \models \neg\psi \vee \chi \\
&\Leftrightarrow \quad M, w \not\models \psi \wedge \neg\chi \text{ for all } M, w \\
&\Leftrightarrow \quad \psi \wedge \neg\chi \models \bot
\end{aligned}$$

For the second statement, if $\psi \not\models \chi$, then there is some $\mathcal{M}, w$ such that $\mathcal{M}, w \models \psi \wedge \neg\chi$. Create a new model $\mathcal{M}'$ from $\mathcal{M}$ by adding a new world w' and placing a single arc from w' to w. Then $\mathcal{M}', w' \models \Diamond\psi \wedge \Box\neg\chi$, which means that $\Diamond\psi \wedge \Box\neg\chi$ is satisfiable and hence $\Diamond\psi \not\models \Diamond\chi$ (since $\Box\neg\chi \equiv \neg\Diamond\chi$). For the other direction, suppose $\Diamond\psi \not\models \Diamond\chi$. Then there exists $\mathcal{M}, w$ such that $\mathcal{M}, w \models \Diamond\psi \wedge \neg\Diamond\chi \equiv \Diamond\psi \wedge \Box\neg\chi$. But this means that there is some w' for which $\psi \wedge \neg\chi$, hence $\psi \not\models \chi$. To complete the proof, we use the following chain of equivalences: $\Box\psi \models \Box\chi \Leftrightarrow \neg\Box\chi \models \neg\Box\psi \Leftrightarrow \Diamond\neg\chi \models \Diamond\neg\psi \Leftrightarrow \neg\chi \models \neg\psi \Leftrightarrow \psi \models \chi$.

For 3, suppose that $\gamma \wedge \Diamond\psi_1 \wedge ... \wedge \Diamond\psi_m \wedge \Box\chi_1 \wedge ... \wedge \Box\chi_n \not\models \bot$. Then there exist $\mathcal{M}, w$ such that $\mathcal{M}, w \models \gamma \wedge \Diamond\psi_1 \wedge ... \Diamond\psi_m \wedge \Box\chi_1 \wedge ... \wedge \Box\chi_n$. As $\mathcal{M}, w \models \gamma$, we cannot have $\gamma \models \bot$, nor can we have $\psi_i \wedge \chi_1 \wedge ... \wedge \chi_n \models \bot$ since for each i there is some w' such that $\mathcal{M}, w' \models \psi_i \wedge \chi_1 \wedge ... \wedge \chi_n$. Now for the other direction suppose that γ and all of the $\psi_i \wedge \chi_1 \wedge ... \wedge \chi_n$ are satisfiable. Then there is some propositional model w of γ, and for each i, we can find $\mathcal{M}_i, w_i$ such that $\mathcal{M}_i, w_i \models \psi_i \wedge \chi_1 \wedge ... \wedge \chi_n$. Now we construct a new Kripke structure which contains the models $\mathcal{M}_i$ and the world w and in which there are arcs going from w to each of the w_i. It is not hard to see that this new model $\mathcal{M}_{new}$ we have $\mathcal{M}_{new}, w \models \gamma \wedge \Diamond\psi_1 \wedge ... \Diamond\psi_m \wedge \Box\chi_1 \wedge ... \Box\chi_n$, so $\gamma \wedge \Diamond\psi_1 \wedge ... \Diamond\psi_m \wedge \Box\chi_1 \wedge ... \wedge \Box\chi_n \not\models \bot$.

Statement 4 follows easily from the third statement. We simply notice that $\gamma \vee \Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee \Box\chi_1 \vee ... \vee \Box\chi_n$ is a tautology just in the case that its negation $\neg\gamma \wedge \Diamond\neg\chi_1 \wedge ... \wedge \Diamond\neg\chi_n \wedge \Box\neg\psi_1 \wedge ... \wedge \Box\neg\psi_m$ is unsatisfiable.

For 5, we use statements 1 and 4 to get the following chain of equivalences:

$$\begin{array}{ll} & \Box\chi \models \Box\chi_1 \vee ... \vee \Box\chi_n \\ \Leftrightarrow & \models \Diamond\neg\chi \vee \Box\chi_1 \vee ... \vee \Box\chi_n \\ \Leftrightarrow & \models \neg\chi \vee \chi_i \text{ for some } i \\ \Leftrightarrow & \chi \models \chi_i \text{ for some } i \end{array}$$

The first implication of the equivalence in 6 is immediate since $\Diamond\psi_1 \vee ... \vee \Diamond\psi_m \models \Diamond\psi_1 \vee ... \vee \Diamond\psi_m$ and $\Box\chi_i \models \Box(\chi_i \vee \psi_1 \vee ... \vee \psi_m)$ for all i. For the other direction, we remark that by using statements 1 and 3, we get the following equivalences:

$$\begin{array}{ll} & \Box(\chi_i \vee \psi_1 \vee ... \vee \psi_m) \models \Box\chi_i \vee \Diamond\psi_1 \vee ... \vee \Diamond\psi_m \\ \Leftrightarrow & \Box(\chi_i \vee \psi_1 \vee ... \vee \psi_m) \wedge \neg(\Box\chi_i \vee \Diamond\psi_1 \vee ... \vee \Diamond\psi_m) \models \bot \\ \Leftrightarrow & \Box(\chi_i \vee \psi_1 \vee ... \vee \psi_m) \wedge \Diamond\neg\chi_i \wedge \Box\neg\psi_1 \wedge ... \wedge \Box\neg\psi_m \models \bot \\ \Leftrightarrow & (\chi_i \vee \psi_1 \vee ... \vee \psi_m) \wedge \neg\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m \models \bot \end{array}$$

As $(\chi_i \vee \psi_1 \vee ... \vee \psi_m) \wedge \neg\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m$ is clearly unsatisfiable, it follows that $\Box(\chi_i \vee \psi_1 \vee ... \vee \psi_m) \models \Box\chi_i \vee \Diamond\psi_1 \vee ... \vee \Diamond\psi_m$ for every i and hence that $\Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee \Box(\chi_1 \vee \psi_1 \vee ... \vee \psi_m) \vee ... \vee \Box(\chi_n \vee \psi_1 \vee ... \vee \psi_m) \models \Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee \Box\chi_1 \vee ... \vee \Box\chi_n$, completing the proof. □

Theorem 2 *Let λ be a disjunction of propositional literals and formulae of the forms $\Diamond\psi$ and $\Box\chi$. Then each of the following statements holds:*

1. *If $\lambda \models \gamma$ for some non-tautologous propositional clause γ, then every disjunct of λ is either a propositional literal or a formula $\Diamond\psi$ where $\psi \models \bot$*
2. *If $\lambda \models \Diamond\psi_1 \vee ... \vee \Diamond\psi_n$, then every disjunct of λ is of the form $\Diamond\psi$*
3. *If $\lambda \models \Box\chi_1 \vee ... \vee \Box\chi_n$ and $\not\models \Box\chi_1 \vee ... \vee \Box\chi_n$, then every disjunct of λ is either a formula of the form $\Box\chi$ or a formula $\Diamond\psi$ where $\psi \models \bot$*

Proof. For (1), let γ be a non-tautologous propositional clause such that $\lambda \models \gamma$, and suppose for a contradiction that λ contains a disjunct $\Box\chi$ or a disjunct $\Diamond\psi$ where $\psi \not\models \bot$. In the first case, we have $\Box\chi \models \gamma$, and hence $\models \Diamond\neg\chi \vee \gamma$. It follows from Theorem 1 that $\models \gamma$, contradicting our assumption that γ is not a tautology. In the second case, we have $\Diamond\psi \models \gamma$, and thus $\models \Box\neg\psi \vee \gamma$. By Theorem 1, either $\models \neg\psi$ or $\models \gamma$. In both cases, we reach a contradiction since we have assumed that $\psi \not\models \bot$ and $\not\models \gamma$. It follows then that λ cannot have any $\Box$-literals or satisfiable $\Diamond$-literals as disjuncts.

The proofs of (2) and (3) proceed similarly. □

Theorem 3 *Let $\lambda = \gamma \vee \Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee \Box\chi_1 \vee ... \vee \Box\chi_n$ and $\lambda' = \gamma' \vee \Diamond\psi'_1 \vee ... \vee \Diamond\psi'_p \vee \Box\chi'_1 \vee ... \vee \Box\chi'_q$ be formulae in $\mathcal{K}$. If γ are γ' are both propositional and $\not\models \lambda'$, then*

$$\lambda \models \lambda' \Leftrightarrow \begin{cases} \gamma \models \gamma' \text{ and} \\ \psi_1 \vee ... \vee \psi_m \models \psi'_1 \vee ... \vee \psi'_p \text{ and} \\ \text{for every } i \text{ there is some } j \text{ such that } \chi_i \models \psi'_1 \vee ... \vee \psi'_p \vee \chi'_j \end{cases}$$

Proof. Since we have $\not\models \lambda'$, we know that $\not\models \gamma'$ and $\not\models \psi'_1 \vee ... \vee \psi'_p \vee \chi'_i$ for all i. Using this information together with Theorem 1, we get the following equivalences:

$$\begin{array}{rcl}
\gamma \models \lambda' & \Leftrightarrow & \models \neg\gamma \vee \gamma' \vee \Diamond\psi'_1 \vee ... \vee \Diamond\psi'_p \vee \Box\chi'_1 \vee ... \vee \Box\chi'_m \\
& \Leftrightarrow & \models \neg\gamma \vee \gamma' \\
& \Leftrightarrow & \gamma \models \gamma' \\
\Diamond\psi_1 \vee ... \vee \Diamond\psi_m \models \lambda' & \Leftrightarrow & \Diamond(\psi_1 \vee ... \vee \psi_m) \models \lambda' \\
& \Leftrightarrow & \models \gamma' \vee \Diamond\psi'_1 \vee ... \vee \Diamond\psi'_p \vee \Box\neg(\psi_1 \vee ... \vee \psi_m) \vee \Box\chi'_1 \vee ... \vee \Box\chi'_q \\
& \Leftrightarrow & \models \psi'_1 \vee ... \vee \psi'_p \vee \neg(\psi_1 \vee ... \vee \psi_m) \\
& \Leftrightarrow & \psi_1 \vee ... \vee \psi_m \models \psi'_1 \vee ... \vee \psi'_p \\
\Box\chi_i \models \lambda' & \Leftrightarrow & \models \gamma' \vee \Diamond(\psi'_1 \vee ... \vee \psi'_p \vee \neg\chi_i) \vee \Box\chi'_1 \vee ... \vee \Box\chi'_q \\
& \Leftrightarrow & \text{there is some } j \text{ such that } \models \psi'_1 \vee ... \vee \psi'_p \vee \neg\chi_i \vee \chi'_j \\
& \Leftrightarrow & \text{there is some } j \text{ such that } \chi_i \models \psi'_1 \vee ... \vee \psi'_p \vee \chi'_j
\end{array}$$

To complete the proof, we use the fact $\lambda \models \lambda'$ if and only if $\gamma \models \lambda'$, $\Diamond\psi_1 \vee ... \vee \Diamond\psi_m \models \lambda'$, and $\Box\chi_i \models \lambda'$ for every i. □

Theorem 4 *Any definition of literals, clause, and terms for $\mathcal{K}$ that satisfies properties **P1** and **P2** cannot satisfy **P5**.*

Proof. We remark that the set of clauses (respectively terms) with respect to definition **D1** is precisely the set of formulae in NNF which do not contain $\wedge$ (respectively $\vee$), i.e. **D1** is the most expressive definition satisfying both **P1** and **P2**. Thus, to show the result, it suffices to show that **D1** does not satisfy **P5**.

Suppose for a contradiction that **D1** does satisfy **P5**. Then there must exist clauses $\lambda_1, ..., \lambda_n$ such that $\Diamond(a \wedge b) \equiv \lambda_1 \wedge ... \wedge \lambda_n$. Each of the clauses λ_i is a disjunction $l_{i,1} \vee .... \vee l_{i,p_i}$. By distributing $\wedge$ over $\vee$, we obtain the following:

$$\Diamond(a \wedge b) \equiv \bigvee_{(j_1,...,j_n)\in\{1,...,p_1\}\times...\times\{1,...,p_n\}} \bigwedge_{i=1}^{n} l_{i,j_i}$$

from which we can infer that for each $(j_1, ..., j_n) \in \{1, ..., p_1\} \times ... \times \{1, ..., p_n\}$ we have

$$\bigwedge_{i=1}^{n} l_{i,j_i} \models \Diamond(a \wedge b)$$

The formulae l_{i,j_i} are either propositional literals or formulae of the form $\Box\kappa$ or $\Diamond\kappa$ for some clause κ. Thus each $\bigwedge_{i=1}^{n} l_{i,j_i}$ must have the following form:

$$\gamma_1 \wedge ... \wedge \gamma_k \wedge \Diamond\psi_1 \wedge ... \wedge \Diamond\psi_m \wedge \Box\chi_1 \wedge ... \wedge \Box\chi_n$$

where γ_1, ..., γ_k are propositional literals and ψ_1, ..., ψ_m, χ_1, ..., χ_n are clauses with respect to **D1**. As we know that $\bigwedge_{i=1}^{n} l_{i,j_i} \models \Diamond(a \wedge b)$, by Theorem 1 it must be either be the case that $\bigwedge_{i=1}^{n} l_{i,j_i}$ is inconsistent or there must be some $\Diamond\psi_q$ such that

$$\Diamond\psi_q \wedge \Box\chi_1 \wedge ... \wedge \Box\chi_n \models \Diamond(a \wedge b)$$

It is evident that we cannot have $\Diamond\psi_q \models \Diamond(a \wedge b)$ since ψ_q is a clause with respect to **D1** and no such clause can imply $a \wedge b$. But this means that there must be some χ_r which is not a tautology (or else we wouldn't have $\Diamond\psi_q \wedge \Box\chi_1 \wedge ...\Box\chi_l \models \Diamond(a \wedge b)$). Let us then consider the formula

$$\tau = \bigvee_{\{(j_1,...,j_n)|\bigwedge_{i=1}^{n} l_{i,j_i} \neq \bot\}} \Box\chi_{j_1,...,j_n}$$

where $\Box\chi_{j_1,...,j_n}$ is a non-tautological $\Box$-formula appearing in $\bigwedge_{i=1}^{n} l_{i,j_i}$. Now clearly it must be the case that

$$\bigvee_{(j_1,...,j_n)\in\{1,...,p_1\}\times...\times\{1,...,p_n\}} \bigwedge_{i=1}^{n} l_{i,j_i} \models \tau$$

and hence that

$$\Diamond(a \wedge b) \models \tau$$

since $\Diamond(a \wedge b) \equiv \bigvee_{(j_1,...,j_n)\in\{1,...,p_1\}\times...\times\{1,...,p_n\}} \bigwedge_{i=1}^{n} l_{i,j_i}$. But according to Theorem 2, a satisfiable $\Diamond$-literal cannot imply a disjunction of $\Box$-literals unless that disjunction is a tautology, so we must have $\models \tau$. But then by Theorem 1 there must be some $\chi_{j_1,...,j_n}$ which is a tautology, contradicting our earlier assumption that the $\chi_{j_1,...,j_n}$ are all non-tautologous. We can thus conclude that there is no set of clauses $\lambda_1, ..., \lambda_n$ with respect to **D1** such that $\Diamond(a \wedge b) \equiv \lambda_1 \wedge ... \wedge \lambda_n$, and hence that any definition which satisfies **P1** and **P2** must not satisfy **P5**. □

In order to prove Theorem 5, we will make use of the following lemmas:

Lemma 5.1 *Definition **D5** satisfies **P5**.*

Proof. We demonstrate that any formula in $\mathcal{K}$ in NNF is equivalent to a formula in conjunction of clauses with respect to definition **D5**. The restriction to formulae in NNF is without loss of generality as every formula is equivalent to a formula in NNF. The proof proceeds by induction on the structural complexity of formulae. The base case is propositional literals, which are already conjunctions of clauses since every propositional literal is a clause with respect to **D5**. We now suppose that the statement holds for formulae ψ_1 and ψ_2 and show that it holds for more complex formulae.

We first consider $\phi = \psi_1 \wedge \psi_2$. By assumption, we can find clausal concepts ρ_i and ζ_j such that $\psi_1 \equiv \rho_1 \wedge ... \wedge \rho_n$ and $\psi_2 \equiv \zeta_1 \wedge ... \wedge \zeta_m$. Thus, ϕ is equivalent to the formula $\rho_1 \wedge ... \wedge \rho_n \wedge \zeta_1 \wedge ... \wedge \zeta_m$, which is a conjunction of clauses with respect to **D5**.

Next we consider $\phi = \psi_1 \vee \psi_2$. By the induction hypothesis, we have $\psi_1 \equiv \rho_1 \wedge ... \wedge \rho_n$ and $\psi_2 \equiv \zeta_1 \wedge ... \wedge \zeta_m$ for some clausal concepts ρ_i and ζ_j. Thus, $\phi \equiv (\rho_1 \wedge ... \wedge \rho_n) \vee (\zeta_1 \wedge ... \wedge \zeta_m)$, which can be written equivalently as $\phi \equiv \wedge_{(i,j)\in\{1,...,n\}\times\{1,...,m\}}(\rho_i \vee \zeta_j)$. Since the union of two clauses produces another clause, all of the $\rho_i \vee \zeta_j$ are clauses, completing the proof.

We now consider the case where $\phi = \Box\psi_1$. By assumption, $\psi_1 \equiv \rho_1 \wedge ... \wedge \rho_n$, where each ρ_i is a clause. So $\phi \equiv \Box(\rho_1 \wedge ... \wedge \rho_n)$. But we also know that $\Box(\rho_1 \wedge ... \wedge \rho_n) \equiv \Box\rho_1 \wedge ... \wedge \Box\rho_n$. It follows that ϕ is equivalent to $\Box\rho_1 \wedge ... \wedge \Box\rho_n$, which is a conjunction of clauses since the $\Box\rho_i$ are all clauses.

Finally, we consider $\phi = \Diamond\psi_1$. Using the induction hypothesis, we have $\phi \equiv \Diamond(\rho_1 \wedge ... \wedge \rho_n)$ for clauses ρ_i. But since the ρ_i are clauses, each ρ_i is a disjunction of literals $l_{i,1} \vee ... \vee l_{i,p_i}$. After

distributing $\wedge$ over $\vee$ and $\vee$ over $\Diamond$, we find that ϕ is equivalent to the formula

$$\bigvee_{(j_1,\ldots,j_n)\in\{1,\ldots,p_1\}\times\ldots\times\{1,\ldots,p_n\}} \Diamond(l_{1,j_1} \wedge l_{2,j_2} \wedge \ldots \wedge l_{n,j_n})$$

which is a clause with respect to **D5**.

The proof that every formula is equivalent to a disjunction of terms with respect to **D5** proceeds analagously. □

Lemma 5.2 *Every clause (respectively term) with respect to **D5** is a clause (respectively term) with respect to definitions **D3a**, **D3b**, and **D4**.*

Proof. We will show by induction that every clause C (resp. term T) with respect to **D5** is a clause (resp. term) with respect to definitions **D3a**, **D3b**, and **D4** *and* a disjunction of terms with respect to **D3a** (resp. conjunction of clauses with respect to **D3a** and **D3b**). We need this stronger formulation of the statement to prove some of the sub-cases. The base case is when C and T are propositional literals, in which case both statements clearly hold since propositional literals are both clauses and terms with respect to definitions **D3a**, **D3b**, and **D4** (and hence also disjunctions of terms with respect to **D3a** and conjunctions of clauses with respect to **D3a** and **D3b**). We now suppose that the statement holds for all proper sub-clauses and sub-terms appearing in C and T, and we aim to show that the statement holds for C and T.

We first consider the clause C. Now C is a clause with respect to **D5** so it can either be a propositional literal or a formula of the form $C_1 \vee C_2$ for clauses C_1 and C_2, $\Box C_1$ for some clause C_1, or $\Diamond T_1$ for some term T_1. The case where C is a propositional literal has already been treated in the base case. Let us thus consider the case where $C = C_1 \vee C_2$. The first part of the statement holds since by the induction hypothesis both C_1 and C_2 are clauses with respect to definitions **D3a**, **D3b**, and **D4**, and for all three definitions the disjunction of two clauses is a clause. The second half of the statement is also verified since both C_1 and C_2 are disjunctions of terms with respect to **D3a**, and thus so is their disjunction $C_1 \vee C_2$. We next consider the case where $C = \Box C_1$ for some clause C_1 with respect to **D5**. The first part of the statement follows easily as we know that C_1 must also be a clause with respect to **D3a**, **D3b**, and **D4**, and for all of these definitions putting a $\Box$ before a clause yields another clause. The second part of the statement holds as well since C_1 is a disjunction of terms with respect to **D3a** and thus $\Box C_1$ is a term with respect to this same definition. We now suppose that $C = \Diamond T_1$ for some term T_1 with respect to **D5**. For definitions **D3a** and **D3b**, we know from the induction hypothesis that T_1 is a conjunction of clauses with respect to **D3a** and **D3b** and hence that $\Diamond T_1$ is a clause with respect to these definitions. For **D4**, the result obviously holds since we are allowed to put anything behind $\Diamond$. The second part of the statement holds since by the induction hypothesis T_1 is a term with respect to **D3a** and hence $\Diamond T_1$ is also a term with respect to this definition.

We now consider the term T, which is known to be either a propositional literal or a formula of the form $T_1 \wedge T_2$ for terms T_1 and T_2, $\Box C_1$ for some clause C_1, or $\Diamond T_1$ for some term T_1. If $T = T_1 \wedge T_2$, the first half of the statement holds since we know T_1 and T_2 to be terms with respect to **D3a**, **D3b**, and **D4**, and conjunctions of terms are also terms for all three definitions. The second half is also verified since both T_1 and T_2 are assumed to be conjunctions of clauses with respect to **D3a** and **D3b**, which means that T is also a conjunction of clauses with respect to these definitions. Next suppose that $T = \Box C_1$. For definitions **D3b** and **D4**, it is easy to

see that T is a literal and hence a term. For **D3a**, the induction hypothesis tell us that C_1 is a disjunction of terms, from which we can deduce that $\Box C_1$ is a term. Moreover, since C_1 is known to be a clause with respect to **D3a** and **D3b**, then $\Box C_1$ must also be a clause with respect to these definitions, so T is a conjunction of clauses with respect to both **D3a** and **D3b**. Finally, we treat the case where $T = \Diamond T_1$. For **D3a**, we use the fact that T_1 is a term with respect to **D3a**, which means that $\Diamond T_1$ must also be a term. For **D3b**, we use the supposition that T_1 is a conjunction of clauses with respect to **D3b**, from which we get that $\Diamond T_1$ is a literal and hence a term. The first part of the statement clearly holds for **D4** as well since any formula behind $\Diamond$ yields a literal and thus a term. The second half of the statement follows from the fact that by the induction hypothesis T_1 is a conjunction of clauses with respect to **D3a** and **D3b**, so $\Diamond T_1$ is a clause (and hence a conjunction of clauses) with respect to these definitions. □

$$\phi_{\mathcal{U},\mathcal{S}} = \phi_{1,1} \wedge ... \wedge \phi_{1,m} \wedge \psi$$

where the $\phi_{i,j}$ are defined inductively as follows

$$\phi_{i,j} = \begin{cases} \Diamond\phi_{i+1,j}, \text{if either } i \leq n, u_i \in S_j, \text{ or } i > n \text{ and } u_{i-n} \in S_j \\ \Box\phi_{i+1,j}, \text{if either } i \leq n, u_i \notin S_j, \text{ or } i > n \text{ and } u_{i-n} \notin S_j \end{cases}$$

for $i \in \{1, ..., 2n\}$ and $\phi_{2n+1,j} = \top$, and $\psi = \underbrace{\Box...\Box}_{2n} \bot$.

Figure 6: The formula $\phi_{\mathcal{U},\mathcal{S}}$ which codes an instance $\mathcal{U} = \{u_1, ..., u_n\}$, $\mathcal{S} = \{S_1, ..., S_m\}$ of the exact cover problem.

Lemma 5.3 *Entailment between terms or clauses is* NP*-complete for both definitions **D1** and **D2**.*

Proof. In the proofs of both NP-membership and NP-hardness, we will exploit the relationship between terms in **D1** and **D2** and concept expressions in the description logic $\mathcal{ALE}$ (cf. [2]). We recall that concept expressions in this logic are constructed as follows (we use a modal logic syntax and assume a single modal operator in order to facilitate comparison between the formalisms):

$$\phi ::= \top \mid \bot \mid a \mid \neg a \mid \phi \wedge \phi \mid \Box\phi \mid \Diamond\phi$$

The semantics of the symbols $\top$ and $\bot$ is as one would expect: $\mathcal{M}, w \models \top$ and $\mathcal{M}, w \not\models \bot$ for every model $\mathcal{M}$ and world w. The semantics of atomic literals, conjunctions, and universal and existential modalities is exactly the same as for $\mathcal{K}$.

It is not hard to see that every term with respect to **D1** or **D2** is a concept expression in $\mathcal{ALE}$. As entailment between $\mathcal{ALE}$ expressions is decidable in nondeterministic polynomial time (cf. [12]), it follows that deciding entailment between terms with respect to either **D1** or **D2** can also be accomplished in nondeterministic polynomial time, i.e. these problems belong to NP.

It remains to be shown that these problems are NP-hard. To prove this, we show how the polynomial-time reduction in [11] (adapted from the original proof in [12]) of the NP-complete exact cover (XC) problem (cf. [16]) to unsatisfiability in $\mathcal{ALE}$ can be modified so as to give a polynomial-time reduction from XC to entailment between terms with respect to **D1** or **D2**.

The exact cover problem is the following: given a set $\mathcal{U} = \{u_1, ..., u_n\}$ and a set $\mathcal{S} = \{S_1, ..., S_m\}$ of subsets of $\mathcal{U}$, determine whether there exists an exact cover, that is, a subset $\{S_{i_1}, ..., S_{i_q}\}$ of $\mathcal{S}$ such that $S_{i_h} \cap S_{i_k} = \emptyset$ for $h \neq k$ and $\bigcup_{k=1}^{q} S_{i_k} = \mathcal{U}$. It has been demonstrated in [11] that $\mathcal{U}, \mathcal{S}$ has an exact cover if and only if the formula $\phi_{\mathcal{U},\mathcal{S}}$ pictured in Figure 6 is unsatisfiable. Notice that $\phi_{\mathcal{U},\mathcal{S}}$ is not a term with respect to either **D1** and **D2** as it uses the symbols $\top$ and $\bot$. We would like to find a similar formula which is a term with respect to our definitions and which is satisfiable if and only if $\phi_{\mathcal{U},\mathcal{S}}$ is. Consider the formula

$$\phi'_{\mathcal{U},\mathcal{S}} = \phi'_{1,1} \wedge ... \wedge \phi'_{1,m} \wedge \psi'$$

where $\phi'_{i,j}$ and ψ' are defined exactly like $\phi_{i,j}$ and ψ except that we replace $\top$ by a and $\bot$ by $\neg a$. It is easy to verify that $\phi'_{\mathcal{U},\mathcal{S}}$ is indeed a term with respect to both **D1** and **D2**. Moreover, it is not too hard to see that $\phi_{1,1} \wedge ... \wedge \phi_{1,m} \models \Diamond^{2n}\top$ if and only if $\phi'_{1,1} \wedge ... \wedge \phi'_{1,m} \models \Diamond^{2n} a$ and hence that $\phi_{\mathcal{U},\mathcal{S}}$ and $\phi'_{\mathcal{U},\mathcal{S}}$ are equisatisfiable. As $\mathcal{U}, \mathcal{S}$ has an exact cover if and only if $\phi_{\mathcal{U},\mathcal{S}}$ is unsatisfiable, and $\phi_{\mathcal{U},\mathcal{S}}$ is unsatisfiable just in the case that $\phi'_{\mathcal{U},\mathcal{S}}$ is, it follows that $\mathcal{U}, \mathcal{S}$ has an exact cover if and only if $\phi'_{\mathcal{U},\mathcal{S}}$ is unsatisfiable. Moreover, $\phi'_{\mathcal{U},\mathcal{S}}$ can be produced in linear time from $\phi_{\mathcal{U},\mathcal{S}}$, so we have a polynomial-time reduction from XC to unsatisfiability of terms in **D1** or **D2**. But a formula is unsatisfiable just in the case that it entails the term $a \wedge \neg a$. So, XC can be polynomially-reduced to entailment between terms with respect to to either **D1** or **D2**, making these problems NP-hard and hence NP-complete.

In order to show the NP-completeness of clausal entailment, we remark that for both definitions **D1** and **D2**, the function **Nnf** transforms negations of clauses into terms and negations of terms into clauses. This means that we can test whether a clause λ entails a clause λ' by testing whether the term $\mathbf{Nnf}(\neg\lambda')$ entails the term $\mathbf{Nnf}(\neg\lambda)$. Likewise, we can test whether a term κ entails another term κ' by testing whether the clause $\mathbf{Nnf}(\neg\kappa')$ entails the clause $\mathbf{Nnf}(\neg\kappa)$. As the NNF transformation is polynomial, it follows that entailment between clauses is exactly as difficult as entailment between terms, so clausal entailment is NP-complete. □

(i) q_0
(ii) $\bigwedge_{i=0}^{m}((q_i \rightarrow \wedge_{j\neq i}\neg q_j) \wedge \Box(q_i \rightarrow \wedge_{j\neq i}\neg q_j) \wedge ... \wedge \Box^m(q_i \rightarrow \wedge_{j\neq i}\neg q_j))$
(iiia) $\bigwedge_{i=0}^{m}((q_i \rightarrow \Diamond q_{i+1}) \wedge \Box(q_i \rightarrow \Diamond q_{i+1}) \wedge ... \wedge \Box^m(q_i \rightarrow \Diamond q_{i+1}))$
(iiib) $\bigwedge_{\{i|Q_i=\forall\}} \Box^i(q_i \rightarrow (\Diamond(q_{i+1} \wedge p_{i+1}) \wedge \Diamond(q_{i+1} \wedge \neg p_{i+1})))$
(iv) $\bigwedge_{i=1}^{m-1}(\bigwedge_{j=i}^{m-1} \Box^j((p_i \rightarrow \Box p_i) \wedge (\neg p_i \rightarrow \Box\neg p_i)))$
(v) $\Box^m(q_m \rightarrow \theta)$

Figure 7: The formula $f(\beta)$ is the conjunction of the above formulae.

Lemma 5.4 *Entailment between clauses or terms of definition **D5** is* PSPACE*-complete.*

Proof. Membership in PSPACE is immediate since entailment between arbitrary formulae in $\mathcal{K}$ can be decided in polynomial space. To prove PSPACE-hardness, we adapt an existing proof of PSPACE-hardness of $\mathcal{K}$.

Figure 7 presents an encoding of a QBF $\beta = Q_1 p_1 ... Q_m p_m \theta$ in a $\mathcal{K}$-formula $f(\beta)$ that is used in [4] to demonstrate the PSPACE-hardness of $\mathcal{K}$. The formula $f(\beta)$ has the property that it is

(i') q_0
(ii') $\bigwedge_{i=0}^{m}(\bigwedge_{j\neq i}((\neg q_i \vee \neg q_j) \wedge \Box(\neg q_i \vee \neg q_j) \wedge ... \wedge \Box^m(\neg q_i \vee \neg q_j)))$
(iiia') $\bigwedge_{i=0}^{m}((\neg q_i \vee \Diamond q_{i+1}) \wedge \Box(\neg q_i \vee \Diamond q_{i+1}) \wedge ... \wedge \Box^m(\neg q_i \vee \Diamond q_{i+1}))$
(iiib') $\bigwedge_{\{i|Q_{i=\forall}\}} \Box^i(\neg q_i \vee \Diamond(q_{i+1} \wedge p_{i+1})) \wedge \Box^i(\neg q_i \vee \Diamond(q_{i+1} \wedge \neg p_{i+1}))$
(iv') $\bigwedge_{i=1}^{m-1}(\bigwedge_{j=i}^{m-1}(\Box^j(\neg p_i \vee \Box p_i) \wedge \Box^j(p_i \vee \Box\neg p_i)))$
(v') $\Box^m(\neg q_m \vee \theta_1) \wedge .... \wedge \Box^m(\neg q_m \vee \theta_l)$

Figure 8: The formula $f'(\beta)$ is the conjunction of the above formulae, where the formulae θ_i in (v') are propositional clauses such that $\theta \equiv \theta_1 \wedge ... \wedge \theta_l$.

satisfiable just in the case that β is a QBF-validity. As the formula $f(\beta)$ can be generated in polynomial-time from β, and the QBF-validity problem is known to be PSPACE-hard, it follows that satisfiability of formulae in $\mathcal{K}$ is PSPACE-hard as well.

In Figure 8, we show a modified encoding. We claim that the following:

(1) $f(\beta)$ and $f'(\beta)$ are logically equivalent

(2) if θ is in CNF, then $f'(\beta)$ is a conjunction of clauses with respect to **D5**

(3) if θ is in CNF, then $f'(\beta)$ can be generated in polynomial time from $f(\beta)$

To show (1), it suffices to show that (i)≡(i'), (ii)≡(ii'), (iiia)≡(iiia'), (iiib)≡(iiib'), (iv)≡(iv'), and (v)≡(v'). The first equivalence is immediate since (i) and (i') are identical. (ii)≡(ii') follows from the fact that $\Box^k(q_i \rightarrow \wedge_{j\neq i}\neg q_j) \equiv \wedge_{j\neq i}\Box^k(\neg q_i \vee \neg q_j)$. (iiia)≡(iiia') holds since (iiia') is just (iiia) with $q_i \rightarrow \Diamond q_{i+1}$ replaced with $\neg q_i \vee \Diamond q_{i+1}$. We have (iiib)≡(iiib') since $\Box^i(q_i \rightarrow (\Diamond(q_{i+1} \wedge p_{i+1}) \wedge \Diamond(q_{i+1} \wedge \neg p_{i+1}))) \equiv \Box^i(\neg q_i \vee \Diamond(q_{i+1} \wedge p_{i+1})) \wedge \Box^i(\neg q_i \vee \Diamond(q_{i+1} \wedge \neg p_{i+1}))$. The equivalence (iv)≡(iv') holds as $\Box^j((p_i \rightarrow \Box p_i) \wedge (\neg p_i \rightarrow \Box\neg p_i)) \equiv \Box^j(\neg p_i \vee \Box p_i) \wedge \Box^j(p_i \vee \Box\neg p_i)$. Finally, we have (v)≡(v') since $\theta \equiv \theta_1 \wedge ... \wedge \theta_l$. Thus, $f(\beta)$ and $f'(\beta)$ are logically equivalent.

To prove (2), we show that each of the component formulae in $f'(\beta)$ is a conjunction of clauses with respect to **D5**, provided that θ is in CNF. Clearly this is the case for (i') as (i') is a propositional literal. The formula (ii') is also a conjunction of clauses with respect to **D5** since it a conjunction formulae of the form $\Box^k(\neg q_i \vee \neg q_j)$. Similarly, (iiia'), (iiib'), and (iv') are all conjunctions of clauses since the formulae $\Box^k(\neg q_i \vee \Diamond q_{i+1})$, $\Box^i(\neg q_i \vee \Diamond(q_{i+1} \wedge p_{i+1}))$, $\Box^i(\neg q_i \vee \Diamond(q_{i+1} \wedge \neg p_{i+1}))$, $\Box^k(\neg p_i \vee \Box p_i)$, and $\Box^k(p_i \vee \Box\neg p_i)$ are all clauses with respect to **D5**. The formula (v') must also be a conjunction of clauses since the θ_i are assumed to be propositional clauses, making each $\Box^m(\neg q_m \vee \theta_i)$ a clause with respect to **D5**, and (v') a conjunction of clauses with respect to **D5**.

For (3), it is clear that we can transform (i), (iiia), (iiib), and (iv) into (i'), (iiia'), (iiib'), and (iv') in polynomial time as the transformations involve only simple syntactic operations and the resulting formulae are at most twice as large. The transformation from (ii) to (ii') is very slightly more involved, but it is not too hard to see the resulting formula is at most m times as large as the original (and m can be no greater than the length of $f(\beta)$). The only step which could potentially result in an exponential blow-up is the transformation from (v) to (v'), as we put θ into CNF. But under the assumption that θ is already in CNF, the transformation can be

executed in polynomial time and space, as all we have to do is separate θ into its conjuncts and rewrite the $(q_m \rightarrow \theta_i)$ as $(\neg q_m \vee \theta_i)$.

Now let $\beta = Q_1 p_1 ... Q_m p_m \theta$ be some QBF such that $\theta = \theta_1 \wedge ... \wedge \theta_l$ for some propositional clauses θ_i. Let $f'(\beta)$ be the formula as defined in Figure 8. By (2) above, we know that $f'(\beta) = \lambda_1 \wedge ... \wedge \lambda_p$ for some clauses λ_i with respect to **D5**. Now consider the following formula

$$\zeta = \Diamond(\Box\lambda_1 \wedge ... \wedge \Box\lambda_p \wedge \Diamond(\Box(a \vee \neg a)))$$

We can show that $f'(\beta)$ is satisfiable if and only if ζ is satisfiable as follows:

$$\begin{array}{ll} & \zeta \text{ is unsatisfiable} \\ \Leftrightarrow & \Box\lambda_1 \wedge ... \wedge \Box\lambda_p \wedge \Diamond(\Box(a \vee \neg a) \text{ is unsatisfiable} \\ \Leftrightarrow & \lambda_1 \wedge ... \wedge \lambda_l \wedge \Box(a \vee \neg a) \text{ is unsatisfiable} \\ \Leftrightarrow & \lambda_1 \wedge ... \wedge \lambda_l \text{ is unsatisfiable} \\ \Leftrightarrow & f'(\beta) \text{ is unsatisfiable} \end{array}$$

But we also know from (1) above that $f'(\beta) \equiv f(\beta)$, and from [4] that $f(\beta)$ is satisfiable just in the case that β is a QBF validity. It is also easy to see that ζ is satisfiable if and only if ζ does not entail the contradiction $\Diamond(a \wedge \neg a)$. Putting this altogether, we find that β is valid just in the case that ζ does not entail $\Diamond(a \wedge \neg a)$. As ζ and $\Diamond(a \wedge \neg a)$ are both clauses and terms with respect to **D5**, we have shown that the QBF-validity problem for QBF with propositional formulae in CNF can be reduced to the problems of entailment of clauses or terms with respect to **D5**. Moreover, this is a polynomial time reduction since it follows from (3) that the transformation from β to ζ can be accomplished in polynomial time. This suffices to show PSPACE-hardness, since it is well-known that QBF-validity remains PSPACE-hard even when we restrict the propositional part θ to be a formula in CNF (cf. [28]). □

Theorem 5 *The results in Figure 1 hold.*

Proof. The satisfaction or dissatisfaction of properties **P1** and **P2** can be immediately determined by inspection of the definitions, as can the satisfaction of **P3** by definitions **D2**, **D3b**, **D4**, and **D5**. Counterexamples to **P3** for definitions **D1** and **D3a** were provided in body of the paper: the formula $\Box(a \vee b)$ is a clause but not a disjunction of literals with respect to both definitions.

In order to show that definition **D3b** does not satisfy **P4**, we remark that the negation of the literal $\Diamond(a \vee b)$ is equivalent to $\Box(\neg a \wedge \neg b)$ which cannot be expressed as a literal in **D3b**. For each of the other definitions, it can be shown (by a straightforward inductive proof) that $\mathbf{Nnf}(\neg L)$ is a literal whenever L is a literal, that $\mathbf{Nnf}(\neg C)$ is a term whenever C is a clause, and that $\mathbf{Nnf}(\neg T)$ is a clause whenever T is a term. This is enough to prove that these definitions satisfy **P4** since $\mathbf{Nnf}(\phi)$ is equivalent to ϕ (by Lemma A.1).

Since we know that definitions **D1** and **D2** satisfy both properties **P1** and **P2**, it follows by Theorem 4 that these definitions do not satisfy **P5**. We have seen in Lemma 5.1 that definition **D5** does satisfy **P5**, i.e. that every formula is equivalent to some conjunction of clauses with respect to **D5** and some disjunction of terms with respect to **D5**. As every clause (resp. term) of **D5** is also a clause (resp. term) with respect to definitions **D3a**, **D3b**, and **D4** (by Lemma 5.2),

it follows that every formula is equivalent to some conjunction of clauses and some disjunction of terms with respect to these definitions, which means they all satisfy **P5**.

It is easy to see that property **P6** is satisfied by all of the definitions since all of our definitions are context-free grammars, and it is well-known that deciding membership for context-free grammars can be accomplished in polynomial time.

From Lemma 5.3, we know that deciding entailment between clauses or terms with respect to either **D1** or **D2** is NP-complete (and hence not in P, unless P=NP). Entailment between clauses/terms is PSPACE-complete for **D5** (Lemma 5.4). As every clause (resp. term) of **D5** is also a clause (resp. term) with respect to definitions **D3a**, **D3b**, and **D4** (from Lemma 5.2), it follows that entailment between clauses or terms is PSPACE-hard for these definitions. Membership in PSPACE is immediate since entailment between arbitrary $\mathcal{K}$ formulae is in PSPACE. □

We prove Theorem 8 in several steps:

Lemma 8.1 *The notions of prime implicates and prime implicants induced by **D4** satisfy **Implicant-Implicate Duality**.*

Proof. Suppose for a contradiction that we have a prime implicant κ of some formula ϕ which is not equivalent to the negation of a prime implicate of $\neg\phi$. Let λ be a clause which is equivalent to $\neg\kappa$ (there must exist such a clause because of property **P4**, cf. Theorem 5). The clause λ is an implicate of $\neg\phi$ since $\kappa \models \phi$ and $\lambda \equiv \neg\kappa$. Since we have assumed that λ is not a prime implicate, there must be some implicate λ' of $\neg\phi$ such that $\lambda' \models \lambda$ and $\lambda \not\models \lambda'$. But then let κ' be a term equivalent to $\neg\lambda'$ (here again we use **P4**). Now κ' must be an implicant of ϕ since $\neg\phi \models \neg\kappa'$. Moreover, κ' is strictly weaker than κ since $\lambda' \models \lambda$ and $\lambda' \not\models \lambda$ and $\kappa \equiv \neg\lambda$ and $\kappa' \equiv \neg\lambda'$. But this means that κ cannot be a prime implicant, contradicting our earlier assumption. Hence, we can conclude that every prime implicant of a formula ϕ is equivalent to the negation of some prime implicate of $\neg\phi$. The proof that every prime implicate of a formula ϕ is equivalent to the negation of a prime implicant of $\neg\phi$ proceeds analagously. □

Lemma 8.2 *Every implicate λ (w.r.t. definition **D4**) of a formula ϕ is entailed by some implicate λ' (w.r.t. definition **D4**) of ϕ with $var(\lambda') \subseteq var(\phi)$ and with depth at most $\delta(\phi)+1$. Likewise every implicant κ (w.r.t. definition **D4**) of ϕ entails an implicant κ' (w.r.t. definition **D4**) of ϕ with $var(\kappa') \subseteq var(\phi)$ and depth at most $\delta(\phi)+1$.*

Proof. We intend to show that the following statement holds: for any formula ϕ and any implicate λ of ϕ, there exists a clause λ' such that $\phi \models \lambda' \models \lambda$ and $var(\lambda') \subseteq var(\phi)$ and $\delta(\lambda) \leq \delta(\phi)+1$. So let ϕ be an arbitrary formula, and let λ be some implicate of ϕ. If ϕ is a tautology, then we can set $\lambda' = a \vee \neg a$ (where $a \in var(\phi)$). If $\lambda \equiv \bot$, then we can set $\lambda' = \Diamond(a \wedge \neg a)$ (where $a \in var(\phi)$), as this clause verifies all of the necessary conditions. Now we consider the case where neither ϕ nor λ is a tautology or a falsehood, and we show how to construct the clause λ'. The first thing we do is use **Dnf-4** from Appendix A to rewrite ϕ as a disjunction of satisfiable terms T_i with respect to **D4** such that the T_i contain only the variables appearing in ϕ and have depth at most $\delta(\phi)$:

$$\phi = T_1 \vee ... \vee T_z$$

As $\phi \models \lambda$, it must be the case that $T_i \models \lambda$ for every T_i. Our aim is to find a clause λ_i for each of the terms T_i such that $T_i \models \lambda_i \models \lambda$ and $var(\lambda_i) \subseteq var(T_i)$ and $\delta(\lambda_i) \leq \delta(T_i)$. So consider some T_i. Since T_i is a term, it has the form $\gamma_1 \wedge ... \wedge \gamma_k \wedge \Diamond\psi_1 \wedge ... \wedge \Diamond\psi_m \wedge \Box\chi_1 \wedge ... \wedge \Box\chi_n$, where γ_1, ..., γ_k are propositional literals. As λ is a clause, it must be of the form $\rho_1 \vee ... \vee \rho_p \vee \Diamond\epsilon_1 \vee ... \vee \Diamond\epsilon_q \vee \Box\zeta_1 \vee ... \vee \Box\zeta_r$, where ρ_1, ..., ρ_p are propositional literals. As $T_i \models \lambda$, it must be the case that the formula

$$\gamma_1 \wedge ... \wedge \gamma_k \wedge \Diamond\psi_1 \wedge ... \wedge \Diamond\psi_m \wedge \Box\chi_1 \wedge ... \wedge \Box\chi_n \wedge$$
$$\neg\rho_1 \wedge ... \wedge \neg\rho_p \wedge \Box\neg\epsilon_1 \wedge ... \wedge \Box\neg\epsilon_q \wedge \Diamond\neg\zeta_1 \wedge ... \wedge \Diamond\neg\zeta_r$$

is unsatisfiable. By Theorem 1, one of the following must hold:

(a) there exists γ_u and ρ_v such that $\gamma_u \equiv \rho_v$

(b) there exists ψ_u such that $\psi_u \wedge \chi_1 \wedge ... \wedge \chi_n \wedge \neg\epsilon_1 \wedge ... \wedge \neg\epsilon_q \models \bot$

(c) there exists ζ_u such that $\neg\zeta_u \wedge \chi_1 \wedge ... \wedge \chi_n \wedge \neg\epsilon_1 \wedge ... \wedge \neg\epsilon_q \models \bot$

Now if (a) holds, we can set $\lambda_i = \gamma_u$ since $T_i \models \gamma_u \models \lambda$, $\delta(\gamma_u) = 0 \leq \delta(T_i)$, and $var(\gamma_u) = \{\gamma_u\} \subseteq var(T_i)$. If it is (b) that holds, then it must be the case that

$$\psi_u \wedge \chi_1 \wedge ... \wedge \chi_n \models \epsilon_1 \vee ... \vee \epsilon_q$$

and hence that

$$\Diamond(\psi_u \wedge \chi_1 \wedge ... \wedge \chi_n) \models \Diamond\epsilon_1 \vee ... \vee \Diamond\epsilon_q \models \lambda$$

We can set $\lambda_i = \Diamond(\psi_u \wedge \chi_1 \wedge ... \wedge \chi_n)$, since $T_i \models \Diamond(\psi_u \wedge \chi_1 \wedge ... \wedge \chi_n) \models \lambda$, $\delta(\Diamond(\psi_u \wedge \chi_1 \wedge ... \wedge \chi_n)) \leq \delta(T_i)$, and $var(\Diamond(\psi_u \wedge \chi_1 \wedge ... \wedge \chi_n)) \subseteq var(T_i)$. Finally, if (c) holds, then it must be the case that

$$\chi_1 \wedge ... \wedge \chi_n \models \epsilon_1 \vee ... \vee \epsilon_q \vee \zeta_u$$

and hence that

$$\Box(\chi_1 \wedge ... \wedge \chi_n) \models \Diamond\epsilon_1 \vee ... \vee \Diamond\epsilon_q \vee \Box\zeta_u \models \lambda$$

So we can set $\lambda_i = \Box(\chi_1 \wedge ... \wedge \chi_n)$, as $T_i \models \Box(\chi_1 \wedge ... \wedge \chi_n) \models \lambda$, $\delta(\Box(\chi_1 \wedge ... \wedge \chi_n)) \leq \delta(T_i)$, and $var(\Box(\chi_1 \wedge ... \wedge \chi_n)) \subseteq var(T_i)$. Thus, we have shown that for every T_i, there is some λ_i such that $T_i \models \lambda_i \models \lambda$ and $var(\lambda_i) \subseteq var(T_i)$ and $\delta(\lambda_i) \leq \delta(T_i)$. But then $\lambda_1 \vee ... \vee \lambda_z$ is a clause implied by every T_i, and hence by ϕ, and such that $var(\lambda_i) \subseteq \cup_i var(T_i) \subseteq var(\phi)$ and $\delta(\lambda_i) \leq max_i\, \delta(T_i) \leq \delta(\phi)$.

Now let κ be an implicant of ϕ, and let λ be the formula $\mathbf{Nnf}(\neg\kappa)$. We know from Lemma A.1 that $\lambda \equiv \neg\kappa$, and it is straightforward to show that λ must be a clause with respect to **D4**. But then λ is an implicate of $\neg\phi$, so there must be some clause λ' with $var(\lambda') \subseteq var(\neg\phi) = var(\phi)$ and depth at most $\delta(\neg\phi) + 1 = \delta(\phi) + 1$ such that $\neg\phi \models \lambda' \models \lambda$. Let κ' be $\mathbf{Nnf}(\neg\lambda')$. It can be easily verified that κ' is a term. Moreover, by Lemma A.1, we have $\kappa' \equiv \neg\lambda'$, $var(\kappa') = var(\neg\lambda') = var(\lambda')$, and $\delta(\kappa') = \delta(\neg\lambda') = \delta(\lambda')$. But then κ' is a term such that $var(\kappa') \subseteq var(\phi)$, $\delta(\kappa') \leq \delta(\phi) + 1$, and $\kappa \models \kappa' \models \phi$. □

Lemma 8.3 *The notions of prime implicates and prime implicants induced by **D4** satisfy **Finiteness**.*

Proof. Consider an arbitrary formula ϕ. From Lemma 8.2, we know that for each prime implicate λ of ϕ, there must be an implicate λ' of ϕ containing only those propositional atoms appearing in ϕ and such that $\delta(\lambda') \leq \delta(\phi) + 1$ and $\lambda' \models \lambda$. But since λ is a prime implicate, we must also have $\lambda \models \lambda'$ and hence $\lambda \equiv \lambda'$. Thus, every prime implicate of ϕ is equivalent to some clause built from the finite set of propositional symbols in ϕ and having depth at most $\delta(\phi) + 1$. As there are only finitely many non-equivalent formulae on a finite alphabet and with fixed depth, it follows that there can be only finitely many distinct prime implicates. By Lemma 8.1, every prime implicant of ϕ is equivalent to the negation of some prime implicate of $\neg\phi$. It follows then that every formula can only have finitely many distinct prime implicants. □

Lemma 8.4 *The notions of prime implicates and prime implicants induced by* ***D4*** *satisfy* ***Covering****.*

Proof. Let ϕ be an arbitrary formula. From Lemma 8.2, we know that every implicate of ϕ is entailed by some implicate of ϕ whose propositional variables are contained in $var(\phi)$ and whose depth is at most $\delta(\phi) + 1$. Now consider the following set

$$\Sigma = \{\sigma \mid \phi \models \sigma, \sigma \text{ is a clause}, var(\sigma) \subseteq var(\phi), \delta(\sigma) \leq \delta(\phi) + 1\}$$

and define another set Π from Σ as follows:

$$\Pi = \{\sigma \in \Sigma \mid \nexists \sigma' \in \Sigma.\ \sigma' \models \sigma \text{ and } \sigma \not\models \sigma'\}$$

In other words, Π is the set of all of the logically strongest implicates of ϕ having depth at most $\delta(\phi) + 1$ and built from the propositional letters in ϕ. We claim the following:

(1) every $\pi \in \Pi$ is a prime implicate of ϕ

(2) for every implicate λ of ϕ, there is some $\pi \in \Pi$ such that $\pi \models \lambda$

We begin by proving (1). Suppose that (1) does not hold, that is, that there is some $\pi \in \Pi$ which is not a prime implicate of ϕ. Since π is by definition an implicate of ϕ, it follows that there must be some implicate λ of ϕ such that $\lambda \models \pi$ and $\pi \not\models \lambda$. But by Lemma 8.2, there is some implicate λ' of ϕ such that $\delta(\lambda') \leq \delta(\phi)+1$, $var(\lambda') \subseteq var(\phi)$, and $\lambda' \models \lambda$. But that means that λ' is an element of Σ which implies but is not implied by π, contradicting the assumption that π is in Π. We can thus conclude that every element of Π must be a prime implicate of ϕ.

For (2): let λ be some implicate of ϕ. Then by Lemma 8.2, there exists some clause $\lambda' \in \Sigma$ such that $\lambda' \models \lambda$. If $\lambda' \in \Pi$, we are done. Otherwise, there must exist some $\sigma \in \Sigma$ such that $\sigma \models \lambda'$ and $\lambda' \not\models \sigma$. If $\sigma \in \Pi$, we are done, otherwise, we find another stronger member of Σ. But as Σ has finitely many elements modulo equivalence, after a finite number of steps, we will find some element which is in Π and which implies λ. Since we have just seen that all members of Π are prime implicates of ϕ, it follows that every implicate of ϕ is implied by some prime implicate of ϕ.

For the second part of **Covering**, let κ be an implicant of ϕ, and let λ be a clause equivalent to $\neg\kappa$ (there must be one because **D4** satisfies **P4**). Now since $\kappa \models \phi$, we must also have $\neg\phi \models \lambda$. According to what we have just shown, there must be some prime implicate π of $\neg\phi$ such that $\neg\phi \models \pi \models \lambda$. By Lemma 8.1, π must be equivalent to the negation of some prime implicant ρ of ϕ. But since $\rho \equiv \neg\pi$ and $\pi \models \lambda$ and $\lambda \equiv \neg\kappa$, it follows that $\kappa \models \rho$, completing the proof. □

Lemma 8.5 *The notions of prime implicates and prime implicants induced by **D4** satisfy **Equivalence**.*

Proof. Let ϕ be some formula in $\mathcal{K}$, and suppose that $\mathcal{M}$ is a model of every prime implicate of ϕ. As **D4** is known to satisfy property **P5** (by Theorem 5), we can find a conjunction of clauses which is equivalent to ϕ. By **Covering** (Lemma 8.3), each of these clauses is implied by some prime implicate of ϕ, so $\mathcal{M}$ must be a model of each of these clauses. It follows that $\mathcal{M}$ is a model of ϕ. For the other direction, we simply note that by the definition of prime implicates if $\mathcal{M}$ is a model of ϕ, then it must also be a model of every prime implicate of ϕ. We have thus shown that $\mathcal{M}$ is a model of ϕ if and only if it is a model of every prime implicate of ϕ. Using a similar argument, we can show that $\mathcal{M}$ is a model of ϕ if and only if it is a model of some prime implicant of ϕ. □

Lemma 8.6 *The notions of prime implicates and prime implicants induced by **D4** satisfy **Distribution**.*

Proof. Let λ be a prime implicate of $\phi_1 \vee ... \vee \phi_n$. Now for each ϕ_i, we must have $\phi_i \models \lambda$. From **Covering**, we know that there must exist some prime implicate λ_i for each ϕ_i such that $\lambda_i \models \lambda$. This means that the formula $\lambda_1 \vee ... \vee \lambda_n$ (which is a clause because it is the disjunction of clauses) entails λ. But since λ is a prime implicate, it must also be the case that $\lambda \models \lambda_1 \vee ... \vee \lambda_n$, and hence $\lambda \equiv \lambda_1 \vee ... \vee \lambda_n$. The proof for prime implicants is entirely similar. □

Theorem 8 *The notions of prime implicates and prime implicants induced by definition **D4** satisfy **Finiteness**, **Covering**, **Equivalence**, **Implicant-Implicate Duality**, and **Distribution**.*

Proof. Follows directly from Lemmas 8.1-8.6. □

Lemma 9.1 *The notions of prime implicates and prime implicants induced by definitions **D1** and **D2** do not satisfy **Equivalence**.*

Proof. The proof is the same for both definitions. Suppose that **Equivalence** holds. Then for every formula ϕ, the set Π of prime implicates of ϕ is equivalent to ϕ. But this means that the set $\Pi \cup \{\neg\phi\}$ is inconsistent, and hence by compactness of $\mathcal{K}$ that there is some finite subset $S \subseteq \Pi \cup \{\neg\phi\}$ which is inconsistent. If $\phi \not\equiv \bot$, then we know that the set S must contain $\neg\phi$ because the set of prime implicates of ϕ cannot be inconsistent. But then the conjunction of elements in $S \setminus \{\neg\phi\}$ is a conjunction of clauses which is equivalent to ϕ. It follows that every formula ϕ is equivalent to some conjunction of clauses. As we have shown earlier in the proof of Theorem 4 that there are formulae which are not equivalent to a conjunction of clauses with respect to **D1** or **D2**, it follows that **Equivalence** cannot hold for these definitions. □

Lemma 9.2 *The notions of prime implicates and prime implicants induced by definitions **D3a**, **D3b**, and **D5** do not satisfy **Finiteness**.*

Proof. Suppose that clauses are defined with respect to definition **D3a**, **D3b**, or **D5** (the proof is the same for all three definitions). Consider the formula $\phi = \Box(a \wedge b)$. It follows from Theorem 3 that ϕ implies $\lambda_k = \Box(\Diamond^k a) \vee \Diamond(a \wedge b \wedge \Box^k \neg a)$ for every $k \geq 1$. As the formulae λ_k are clauses (with respect to **D3a**, **D3b**, and **D5**), the λ_k are all implicates of ϕ. To complete the proof,

we show that every λ_k is a prime implicate of ϕ. Since the λ_k are mutually non-equivalent, it follows that ϕ has infinitely many prime implicates modulo equivalence.

Consider some λ_k and some implicate $\mu = \Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee \Box\chi_1 \vee ... \vee \Box\chi_n$ of ϕ that implies it (by Theorem 2 there cannot be any propositional literals in μ). Using Theorem 3 and the fact that $\phi \models \mu \models \lambda_k$, we get the following:

(a) $a \wedge b \models \chi_i \vee \psi_i \vee ... \vee \psi_m$ for some χ_i

(b) $\chi_i \models (\Diamond^k a) \vee (a \wedge b \wedge \Box^k \neg a)$ for every χ_i

(c) $\psi_1 \vee ... \vee \psi_m \models a \wedge b \wedge \Box^k \neg a$

Let χ_i be such that $a \wedge b \models \chi_i \vee \psi_i \vee ... \vee \psi_m$. We remark that χ_i must be satisfiable since otherwise we can combine (a) and (c) to get $a \wedge b \models a \wedge b \wedge \Box^k \neg a$. Now by (b), we know that $\chi_i \models (\Diamond^k a) \vee (a \wedge b \wedge \Box^k \neg a)$ and hence that $\chi_i \wedge (\Box^k \neg a) \wedge (\neg a \vee \neg b \vee \Diamond^k a)$ is inconsistent. It follows that both $\chi_i \wedge (\Box^k \neg a) \wedge \neg a$ and $\chi_i \wedge (\Box^k \neg a) \wedge \neg b$ are inconsistent. Using Theorem 1, we find that either $\chi_i \models \Diamond^k a$ or $\chi_i \models a \wedge b$. As χ_i is a satisfiable clause with respect to definitions **D3a**, **D3b**, and **D5**, it cannot imply $a \wedge b$, so we must have $\chi_i \models \Diamond^k a$. By putting (a) and (c) together, we find that

$$a \wedge b \wedge \neg\chi_i \models \psi_1 \vee ... \vee \psi_m \models a \wedge b \wedge \Box^k \neg a$$

It follows that $\neg\chi_i \models \Box^k \neg a$, i.e. $\Diamond^k a \models \chi_i$. We thus have $\chi_i \equiv \Diamond^k a$ and $\psi_1 \vee ... \vee \psi_m \equiv a \wedge b \wedge \Box^k \neg a$. As $\Diamond^k a \models \chi_i$ and $a \wedge b \wedge \Box^k \neg a \models \psi_1 \vee ... \vee \psi_m$, by Theorem 3 we get $\Box(\Diamond^k a) \vee \Diamond(a \wedge b \wedge \Box^k \neg a) \models \Box\chi_i \vee \Diamond\psi_i \vee ... \vee \Diamond\psi_m \models \mu$ and hence $\lambda_k \equiv \mu$. We have thus shown that any implicate of ϕ which implies λ_k must be equivalent to λ_k. This means that each λ_k is a prime implicate of ϕ, completing the proof. □

Theorem 9 *The notions of prime implicates and prime implicants induced by definitions **D1** and **D2**, do not satisfy **Equivalence**. The notions of prime implicates and prime implicants induced by **D3a**, **D3b**, and **D5** falsify **Finiteness**.*

Proof. Follows directly from Lemmas 9.1 and 9.2. □

In the proof of Theorem 10, we will make use of the following lemmas:

Lemma 10.1 *The algorithm* **GenPI** *always terminates.*

Proof. We know from Lemma A.2 that the algorithm **Dnf-4** always terminates and returns a finite set of formulae. This means that there are only finitely many terms T to consider. For each T, the set $\Delta(T)$ contains only finitely many elements (this is immediate given the definition of $\Delta(T)$), which means that the set CANDIDATES also has finite cardinality. In the final step, we compare at most once each pair of elements in CANDIDATES. As the comparison always terminates, and there are only finitely many pairs to check, it follows that the algorithm **GenPI** terminates. □

Lemma 10.2 *The algorithm* **GenPI** *outputs exactly the set of prime implicates of the input formula.*

Proof. We first prove that every prime implicate of a term T is equivalent to some element in $\Delta(T)$. Let $T = \gamma_1 \wedge ... \wedge \gamma_k \wedge \Diamond\psi_1 \wedge ... \wedge \Diamond\psi_m \wedge \Box\chi_1 \wedge ... \wedge \Box\chi_n$ be some term, and let $\lambda = \rho_1 \vee ... \vee \rho_p \vee \Diamond\epsilon_1 \vee ... \vee \Diamond\epsilon_q \vee \Box\zeta_1 \vee ... \vee \Box\zeta_r$ be one of its prime implicates. As $T \models \lambda$, it must be the case that

$$\gamma_1 \wedge ... \wedge \gamma_k \wedge \Diamond\psi_1 \wedge ... \wedge \Diamond\psi_m \wedge \Box\chi_1 \wedge ... \wedge \Box\chi_n \wedge$$
$$\neg\rho_1 \wedge ... \wedge \neg\rho_p \wedge \Box\neg\epsilon_1 \wedge ... \wedge \Box\neg\epsilon_q \wedge \Diamond\neg\zeta_1 \wedge ... \wedge \Diamond\neg\zeta_r$$

is unsatisfiable. By Theorem 1, one of the following must hold:

(a) there exists γ_u and ρ_v such that $\gamma_u \equiv \rho_v$

(b) there exists ψ_u such that $\psi_u \wedge \chi_1 \wedge ... \wedge \chi_n \models \epsilon_1 \vee ... \vee \epsilon_q$

(c) there exists ζ_u such that $\chi_1 \wedge ... \wedge \chi_n \models \zeta_u \vee \epsilon_1 \vee ... \vee \epsilon_q$

If (a) holds, then $\gamma_u \models \lambda$, so λ must be equivalent to γ_u or else we would have found a stronger implicate, contradicting our assumption that λ is a prime implicate of T. But then the result holds since γ_u is in $\Delta(T)$. If (b) holds, then the formula $\Diamond(\psi_u \wedge \chi_1 \wedge ... \wedge \chi_n)$ is an implicate of T which implies λ, so $\lambda \equiv \Diamond(\psi_u \wedge \chi_1 \wedge ... \wedge \chi_n)$. We are done since $\Diamond(\psi_u \wedge \chi_1 \wedge ... \wedge \chi_r)$ is a member of $\Delta(T)$. Finally we consider the case where (c) holds. In this case, $\Box(\chi_1 \wedge ... \wedge \chi_n)$ is an implicate of T which implies λ, and so is equivalent to λ (as λ is a prime implicate). But then we have the desired result since $\Box(\chi_1 \wedge ... \wedge \chi_n)$ is one of the elements in $\Delta(T)$. Thus we can conclude that every prime implicate of a term T is equivalent to some element in $\Delta(T)$. By Lemma A.2, the elements in **Dnf-4**(ϕ) are terms, and their disjunction is equivalent to ϕ. As **D4** satisfies **Distribution**, it follows that every prime implicate of the input ϕ is equivalent to some element in CANDIDATES. This means that if an element λ_i in CANDIDATES is not a prime implicate of ϕ, then there is some prime implicate π of ϕ that implies but is not implied by λ_i, and hence some $\lambda_j \in$ CANDIDATES such that $\lambda_j \models \lambda_i$ and $\lambda_i \not\models \lambda_j$. Thus, during the comparison phase, this clause will be removed from CANDIDATES. Now suppose that the clause λ is a prime implicate of ϕ. Then we know that there must be some $\lambda_i \in$ CANDIDATES such that $\lambda_i \equiv \lambda$, and moreover, we can choose λ_i so that there is no λ_j with $j < i$ such that $\lambda_j \models \lambda_i$. When in the final step we compare λ_i with all the clauses λ_j with $j \neq i$, we will never find that $\lambda_j \models \lambda_i$ for $j < i$, nor can we have $\lambda_j \models \lambda_i \not\models \lambda_j$ for some $j > i$, otherwise λ would not be a prime implicate. It follows then that λ_i remains in the set CANDIDATES which is returned by the algorithm. We have thus shown that the output of **GenPI** with input ϕ is precisely the set of prime implicates of ϕ. □

Theorem 10 *The algorithm* **GenPI** *always terminates and outputs exactly the set of prime implicates of the input formula.*

Proof. Follows directly from Lemmas 10.1 and 10.2. □

Theorem 11 *The length of the smallest representation of a prime implicate of a formula can be no more than singly exponential in the length of the formula.*

Proof. Prime implicates generated by **GenPI** can have at most $2^{|\phi|}$ disjuncts as there are at most $2^{|\phi|}$ terms in **Dnf-4**(ϕ) by Lemma A.3. Moreover, each disjunct has length at most $2|\phi|$ (also by Lemma A.3). This gives us a total of $2|\phi| * 2^{|\phi|}$ symbols, to which we must add the at

most $2^{|\phi|} - 1$ disjunction symbols connecting the disjuncts. We thus find that the length of the smallest representation of a prime implicate of a formula ϕ is at most $2|\phi| * 2^{|\phi|} + (2^{|\phi|} - 1)$. $\square$

Theorem 12 *The length of the smallest representation of a prime implicate of a formula can be exponential in the length of the formula.*

Proof. Consider the formula

$$\phi = (\bigwedge_{i=1}^{n} (\Box a_{i1} \vee \Box a_{i2})) \wedge \Diamond(\neg a_{11} \wedge \neg a_{21} \wedge ... \wedge \neg a_{n1})$$

and the clause

$$\lambda = \bigvee_{\substack{(i_1,...,i_n) \in \{1,2\}^n \\ (i_1,...,i_n) \neq (1,1,...,1)}} \Box(a_{1i_1} \wedge a_{2i_2} \wedge ... \wedge a_{ni_n})$$

where $a_{ij} \neq a_{kl}$ whenever $i \neq k$ or $j \neq l$. We aim to show that (a) λ is a prime implicate of ϕ, and (b) any clause equivalent to λ must have length at least $|\lambda|$. This is enough to prove the result since λ clearly has size exponential in n, whereas the size of ϕ is only linear in n.

We begin by proving that λ is a prime implicate of ϕ. Let λ' be a clause such that $\phi \models \lambda'$ and $\lambda' \models \lambda$. Now since $\lambda' \models \lambda$, it follows from Theorem 2 that λ' is equivalent to a clause of the form $\Box\chi_1 \vee ... \vee \Box\chi_m$. As $\phi \models \lambda'$, then we must have $T \models \lambda'$ for every $T \in \mathbf{Dnf\text{-}4}(\phi)$. Since λ' contains only $\Box$-literals, this means that $\Box\beta_T \models \lambda'$ for every $T \in \mathbf{Dnf\text{-}4}(\phi)$, where β_T is the conjunction of formulae ρ such that $\Box\rho$ is a conjunct of T. We thus get that $\bigvee_T \Box\beta_T \models \lambda'$. But we remark that the disjuncts of λ are exactly the strongest $\Box$-literals of the terms in $\mathbf{Dnf\text{-}4}(\phi)$, i.e. $\lambda \equiv \bigvee_T \Box\beta_T$. It follows that $\lambda' \equiv \bigvee_T \Box\beta_T \equiv \lambda$, from which we can conclude that λ is a prime implicate of ϕ.

We now show that there are no more compact ways of representing λ. Let λ' be a shortest clause which is equivalent to λ. As λ' is equivalent to λ, it follows from Theorem 2 that λ' is a disjunction of $\Box$-literals and of inconsistent $\Diamond$-literals. But since λ' is assumed to be a shortest representation of λ, it cannot contain any inconsistent $\Diamond$-literals or any redundant $\Box$-literals, since we could remove them to find an equivalent shorter clause. So λ' must be of the form $\Box\chi_1 \vee ... \vee \Box\chi_m$, where $\chi_l \not\models \chi_j$ whenever $l \neq j$. Now since $\lambda' \models \lambda$, every disjunct $\Box\chi_p$ must also imply λ. As λ is a disjunction of $\Box$-literals, it follows from Theorem 3 that every disjunct $\Box\chi_p$ of λ' implies some disjunct $\Box\delta_q$ of λ. But that means that every $\Box\chi_p$ must have length at least $2n$, since each χ_p is a satisfiable formula which implies a conjunction of n distinct propositional variables. We also know that every disjunct $\Box\delta_q$ of λ implies some disjunct $\Box\chi_p$ of λ' since $\lambda \models \lambda'$. We now wish to show that no two disjuncts of λ imply the same disjunct of λ'. Suppose that this is not the case, that is, that there are distinct disjuncts $\Box\delta_1$ and $\Box\delta_2$ of λ and some disjunct $\Box\chi_p$ of λ' such that $\Box\delta_1 \models \Box\chi_p$ and $\Box\delta_2 \models \Box\chi_p$. Now since $\Box\delta_1$ and $\Box\delta_2$ are distinct disjuncts, there must be some i such that $\Box\delta_1 \models a_{i1}$ and $\Box\delta_2 \models a_{i2}$ or $\Box\delta_1 \models a_{i2}$ and $\Box\delta_2 \models a_{i1}$. We know that $\Box\chi_p \models \Box\delta_q$ for some δ_q, and that every δ_q implies either a_{i1} or a_{i2}, so either $\Box\chi_p \models \Box a_{i1}$ or $\Box\chi_p \models \Box a_{i2}$. But we know that the $\Box\delta_q$ each imply either $\Box a_{i1}$ or $\Box a_{i2}$ but not both, so one of $\Box\delta_1$ and $\Box\delta_2$ must not imply $\Box\chi_p$. This contradicts our earlier assumption that $\Box\delta_1 \models \Box\chi_p$ and $\Box\delta_2 \models \Box\chi_p$, so each disjunct of λ must imply a distinct disjunct of λ'. We have thus demonstrated that λ' contains just as many disjuncts as λ. As we have already shown that the disjuncts of λ' are no shorter than the disjuncts of λ, it follows that $|\lambda'| \geq |\lambda|$, and hence

$|\lambda'| = |\lambda|$. We conclude that every clause equivalent to λ has length at least $|\lambda|$, completing the proof. □

Theorem 13 *The number of non-equivalent prime implicates of a formula is at most doubly exponential in the length of the formula.*

Proof. We know from Theorem 10 that every prime implicate of ϕ is equivalent to some clause returned by **GenPI**. Every such clause is of the form $\bigvee_{T \in \textbf{Dnf-4}(\phi)} \theta_T$ where $\theta_T \in \Delta(T)$. As there can be at most $2^{|\phi|}$ terms in **Dnf-4**(ϕ)) by Lemma A.3, these clauses can have no more than $2^{|\phi|}$ disjuncts. Moreover, there are at most $2|\phi|$ choices for each disjunct θ_T since the cardinality of $\Delta(T)$ is bounded above by the size of T, which we know from Lemma 1.3 to be no more than $2|\phi|$. It follows then that there are at most $(2|\phi|)^{2^{|\phi|}}$ clauses returned by **GenPI**, hence at most $(2|\phi|)^{2^{|\phi|}}$ non-equivalent prime implicates of ϕ. □

Theorem 14 *The number of non-equivalent prime implicates of a formula may be doubly exponential in the length of the formula.*

Proof. Let n be some natural number, and let a_{11}, a_{12}, ..., a_{n1}, a_{n2}, b_{11}, b_{12}, b_{12}, ..., b_{n1}, b_{n2} be distinct propositional variables. Consider the formula ϕ defined as

$$\bigwedge_{i=1}^{n}((\Diamond a_{i1} \wedge \Box b_{i1}) \vee (\Diamond a_{i2} \wedge \Box b_{i2}))$$

It is not hard to see that there will be 2^n terms in **Dnf-4**(ϕ), corresponding to the 2^n ways of deciding for each $i \in \{1, ..., n\}$ whether to take the first or second disjunct. Each term $T \in$ **Dnf-4**(ϕ) will be of the form

$$\bigwedge_{i=1}^{n}(\Diamond a_{i\, f(i,T)} \wedge \Box b_{i\, f(i,T)})$$

where $f(i, T) \in \{1, 2\}$ for all i. For each T, denote by $\mathcal{D}(T)$ the set of formulae $\{\Diamond(a_{\, f(i,T)} \wedge b_{1\, f(1,T)} \wedge ... \wedge b_{n\, f(n,T)})) \,|\, 1 \leq i \leq n\}$. Now consider the set of clauses $\mathcal{C}$ defined as

$$\{\bigvee_{T \in \textbf{Dnf-4}(\phi)} d_T \,|\, d_T \in \mathcal{D}(T)\}$$

Notice that there are n^{2^n} clauses in $\mathcal{C}$ since each clause corresponds to a choice of one of the n elements in $\mathcal{D}(T)$ for each of the 2^n terms T in **Dnf-4**(ϕ). This number is doubly-exponential in $|\phi|$ since the length of ϕ is linear in n. In order to complete the proof, we show that (i) all of the clauses in $\mathcal{C}$ are prime implicates of ϕ and (ii) that the clauses in $\mathcal{C}$ are mutually non-equivalent.

We begin by showing that $\lambda_1 \not\models \lambda_2$ for every pair of distinct elements λ_1 and λ_2 in $\mathcal{C}$. This immediately gives us (ii) and will prove useful in the proof of (i). Let λ_1 and λ_2 be distinct clauses in $\mathcal{C}$. As λ_1 and λ_2 are distinct, there must be some term $T \in$ **Dnf-4**(ϕ) for which λ_1 and λ_2 choose different elements from $\mathcal{D}(T)$. Let d_1 be the disjunct from $\mathcal{D}(T)$ appearing as a disjunct in λ_1, let d_2 be the element in $\mathcal{D}(T)$ which is a disjunct in λ_2, and let $a_{j,k}$ be the a-literal which appears in d_2 (and hence not in d_1). Consider the formula $\rho = \Box(\neg a_{j,k} \wedge \neg b_{1,k_1} \wedge ... \wedge \neg b_{n,k_n})$,

where the tuple $(k_1, ..., k_n)$ is just like the tuple associated with T except that the 1's and 2's are inversed. Clearly $d_1 \wedge \rho$ is consistent, since the variables in ρ do not appear in d_1. But ρ is inconsistent with every disjunct in λ_2, since by construction every disjunct in λ_2 contains a literal whose negation appears in ρ. It follows that $\lambda_2 \models \neg\rho$ but $\lambda_1 \not\models \neg\rho$, and hence $\lambda_1 \not\models \lambda_2$.

We now prove (i). Let λ be a clause in $\mathcal{C}$, and let π be a prime implicate of ϕ which implies λ. By Theorem 10, we know that π must be equivalent to one of the clauses outputted by **GenPI**, and more specifically to a clause outputted by **GenPI** which is a disjunction of $\Diamond$-literals (because of Theorem 2). We remark that the set $\mathcal{C}$ is composed of exactly those candidate clauses which are disjunctions of $\Diamond$-literals, so π must be equivalent to some clause in $\mathcal{C}$. But we have just shown that the only element in $\mathcal{C}$ which implies λ is λ itself. It follows that $\pi \equiv \lambda$, which means that λ is a prime implicate of ϕ.

□

Theorem 16 *Prime implicate recognition is* PSPACE*-hard.*

Proof. The reduction is simple: a formula ϕ is unsatisfiable if and only if $\Diamond(a \wedge \neg a)$ is a prime implicate of ϕ. This suffices as the problem of checking the unsatisfiability of formulae in $\mathcal{K}$ is known to be PSPACE-complete. □

Theorem 17 *Prime implicate recognition is in* EXPSPACE*.*

Proof. Consider the following algorithm for determining whether a clause λ is a prime implicate of ϕ: check for each clause λ' of length at most $2|\phi| * 2^{|\phi|} + (2^{|\phi|} - 1)$ which is an implicate of ϕ whether both $\lambda' \models \lambda$ and $\lambda \not\models \lambda'$. If there is some λ' satisfying these conditions, return **no**, otherwise return **yes**. Notice that if this algorithm returns **yes**, then there is no implicate of length at most $2|\phi| * 2^{|\phi|} + (2^{|\phi|} - 1)$ that is strictly stronger than λ, and hence by Corollary 11 no strictly stronger implicate of any length, making λ a prime implicate. If the algorithm returns **no**, then we have found a clause implied by ϕ which is strictly stronger than λ, so λ is not a prime implicate. The algorithm is thus both correct and complete. As the algorithm consists solely in testing the satisfiability and unsatisfiability of formulae having length at most singly-exponential in $|\phi| + |\lambda|$, and both tasks can be accomplished in polynomial space in the size of the input, the algorithm can be executed in exponential space. □

We will need the following two lemmas for Theorem 18:

Lemma 18.1 *Let ϕ be a formula of $\mathcal{K}$, and let $\lambda = \gamma_1 \vee ... \vee \gamma_k \vee \Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee \Box\chi_1 \vee ... \vee \Box\chi_n$ be a non-tautologous clause. Suppose furthermore that there is no literal l in λ such that $\lambda \equiv \lambda \setminus \{l\}$. If $\lambda \in \Pi(\phi)$, then $\gamma_1 \vee ... \vee \gamma_k \in \Pi(\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\}))$ and $\Diamond(\psi_1 \vee ... \vee \psi_n) \in \Pi(\phi \wedge \neg(\lambda \setminus \{\Diamond\psi_1, ..., \Diamond\psi_m\}))$ and for every i, $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m) \in \Pi(\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\}))$.*

Proof. We will prove the contrapositive: if $\gamma_1 \vee ... \vee \gamma_k \notin \Pi(\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\}))$ or $\Diamond(\psi_1 \vee ... \vee \psi_n) \notin \Pi(\phi \wedge \neg(\lambda \setminus \{\Diamond\psi_1, ..., \Diamond\psi_m\}))$ or there is some i for which $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m) \notin \Pi(\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\}))$, then $\lambda \notin \Pi(\phi)$. We will only consider the case where $\phi \models \lambda$ because if $\phi \not\models \lambda$ then we immediately get $\lambda \notin \Pi(\phi)$.

Let us first suppose that $\gamma_1 \vee ... \vee \gamma_k \notin \Pi(\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\}))$. Since $\phi \models \lambda$, we must also have $\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\}) \models \gamma_1 \vee ... \vee \gamma_k$, so $\gamma_1 \vee ... \vee \gamma_k$ is an implicate of $\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\})$. As $\gamma_1 \vee ... \vee \gamma_k$ is known not to be a prime implicate of $\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\})$, it follows that there must

be some clause λ' such that $\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\}) \models \lambda' \models \gamma_1 \vee ... \vee \gamma_k \not\models \lambda'$. Consider the clause $\lambda'' = \lambda' \vee \Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee \Box\chi_1 \vee ... \vee \Box\chi_n$. We know that $\phi \models \lambda''$ since $\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\}) \models \lambda'$ and that $\lambda'' \models \lambda$ since $\lambda' \models \gamma_1 \vee ... \vee \gamma_k$. We also have that $\lambda \not\models \lambda''$ since λ' must be equivalent to a propositional clause (by Theorem 2) and the propositional part of λ does not imply λ'. It follows then that $\phi \models \lambda'' \models \lambda \not\models \lambda''$, so $\lambda \notin \Pi(\phi)$.

Next suppose that $\Diamond(\psi_1 \vee ... \vee \psi_n) \notin \Pi(\phi \wedge \neg(\lambda \setminus \{\Diamond\psi_1, ..., \Diamond\psi_m\}))$. Now $\Diamond(\psi_1 \vee ... \vee \psi_n)$ must be an implicate of $\phi \wedge \neg(\lambda \setminus \{\Diamond\psi_1, ..., \Diamond\psi_m\})$ since we have assumed that $\phi \models \lambda$. As $\Diamond(\psi_1 \vee ... \vee \psi_n)$ is not a prime implicate of $\phi \wedge \neg(\lambda \setminus \{\Diamond\psi_1, ..., \Diamond\psi_m\})$, it follows that there is some λ' such that $\phi \wedge \neg(\lambda \setminus \{\Diamond\psi_1, ..., \Diamond\psi_m\}) \models \lambda' \models \Diamond(\psi_1 \vee ... \vee \psi_n) \not\models \lambda'$. Let $\lambda'' = \gamma_1 \vee ... \vee \gamma_k \vee \lambda' \vee \Box\chi_1 \vee ... \vee \Box\chi_n$. Because of Theorem 2, we know that λ' is a disjunction of $\Diamond$-literals, so according to Theorem 3 we must have $\lambda \not\models \lambda''$ since $\Diamond(\psi_1 \vee ... \vee \psi_n) \not\models \lambda'$. We also know that $\phi \models \lambda''$ since $\phi \wedge \neg(\lambda \setminus \{\Diamond\psi_1, ..., \Diamond\psi_m\}) \models \lambda'$ and that $\lambda'' \models \lambda$ since $\lambda' \models \Diamond(\psi_1 \vee ... \vee \psi_n)$. That means that $\phi \models \lambda'' \models \lambda \not\models \lambda''$, so $\lambda \notin \Pi(\phi)$.

Finally consider the case where there is some i for which $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m) \notin \Pi(\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\}))$. We know that $\phi \models \lambda$ and hence that $\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\}) \models \Box\chi_i$. Moreover, since $\neg(\lambda \setminus \{\Box\chi_i\}) \models \neg\Diamond\psi_j$ for all j, we have $\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\}) \models \Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m)$. Thus, if $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m) \notin \Pi(\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\}))$, it must mean that there is some λ' such that $\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\}) \models \lambda' \models \Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m) \not\models \lambda'$. By assumption, λ is not a tautology, so $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m)$ cannot be a tautology either. As $\lambda' \models \Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m)$ and $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m)$ is not a tautology, it follows from Theorem 2 that λ' is equivalent to some formula $\Box\zeta_1 \vee ... \vee \Box\zeta_p$. Let $\lambda'' = \gamma_1 \vee ... \vee \gamma_k \vee \Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee \Box\chi_1 \vee ... \vee \Box\chi_{i-1} \vee (\Box\zeta_1 \vee ... \vee \Box\zeta_p) \vee \Box\chi_{i+1} \vee ... \vee \Box\chi_n$. As $\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\}) \models \Box\zeta_1 \vee ... \vee \Box\zeta_p$, it must be the case that $\phi \models \lambda''$. Also, we know that there can be no j such that $\chi_i \models \zeta_j \vee \psi_1 \vee ... \vee \psi_m$ because otherwise we would have $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m) \models \Box\zeta_1 \vee ... \vee \Box\zeta_p$. Similarly, there can be no $k \neq i$ such that $\Box\chi_i \models \Box(\chi_k \vee \psi_1 \vee ... \vee \psi_m)$ because this would mean that $\lambda \equiv \lambda \setminus \{\Box\chi_i\}$, contradicting one of our assumptions. It follows then by Theorem 3 that $\lambda \not\models \lambda''$. Thus, $\phi \models \lambda'' \models \lambda \not\models \lambda''$, which means $\lambda \notin \Pi(\phi)$. □

Lemma 18.2 *Let ϕ be a formula of $\mathcal{K}$, and let $\lambda = \gamma_1 \vee ... \vee \gamma_k \vee \Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee \Box\chi_1 \vee ... \vee \Box\chi_n$ be a non-tautologous clause. Suppose furthermore that there is no literal l in λ such that $\lambda \equiv \lambda \setminus \{l\}$. Then if $\lambda \notin \Pi(\phi)$, either $\gamma_1 \vee ... \vee \gamma_k \notin \Pi(\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\}))$ or $\Diamond(\psi_1 \vee ... \vee \psi_m) \notin \Pi(\phi \wedge \neg(\gamma_1 \vee ... \vee \gamma_k \vee \Box(\chi_1 \vee \psi_1 \vee ... \vee \psi_m) \vee ... \vee \Box(\chi_n \vee \psi_1 \vee ... \vee \psi_m)))$ or $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m) \notin \Pi(\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\}))$ for some i.*

Proof. We will only consider the case where $\phi \models \lambda$ because if $\phi \not\models \lambda$ then we immediately get the result. Suppose then that $\lambda \notin \Pi(\phi)$ and $\phi \models \lambda$. By Definition 6, there must be some $\lambda' = \gamma'_1 \vee ... \gamma'_o \vee \Diamond\psi'_1 \vee ... \vee \Diamond\psi'_p \vee \Box\chi'_1 \vee ... \vee \Box\chi'_q$ such that $\phi \models \lambda' \models \lambda \not\models \lambda'$. Since $\lambda \not\models \lambda'$, by Proposition 3 we know that either $\gamma_1 \vee ... \vee \gamma_k \not\models \gamma'_1 \vee ... \vee \gamma'_o$ or $\psi_1 \vee ... \vee \psi_m \not\models \psi'_1 \vee ... \vee \psi'_p$ or there is some i for which $\chi_i \not\models \chi'_j \vee \psi'_1 \vee ... \vee \psi'_p$ for all j.

We begin with the case where $\gamma_1 \vee ... \vee \gamma_k \not\models \gamma'_1 \vee ... \vee \gamma'_o$. As $\lambda' \models \lambda$, by Theorem 3, $\psi'_1 \vee ... \vee \psi'_p \models \psi_1 \vee ... \vee \psi_m$ and for every i there is some j such that $\chi'_i \models \psi_1 \vee ... \vee \psi_m \vee \chi_j$. It follows then (also by Theorem 3) that $\lambda' \models \gamma'_1 \vee ... \vee \gamma'_o \vee \Diamond\psi_1 \vee ... \Diamond\psi_m \vee \Box\chi_1 \vee ... \vee \Box\chi_n$, and hence that $\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\}) \models \gamma'_1 \vee ... \vee \gamma'_o$. As $\gamma'_1 \vee ... \vee \gamma'_o \models \gamma_1 \vee ... \vee \gamma_k \not\models \gamma'_1 \vee ... \vee \gamma'_o$, we have found an implicate of $\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\})$ which is stronger than $\gamma_1 \vee ... \vee \gamma_k$, so $\gamma_1 \vee ... \vee \gamma_k \notin \Pi(\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\}))$.

Next suppose that $\psi_1 \vee ... \vee \psi_m \not\models \psi'_1 \vee ... \vee \psi'_p$. As $\lambda' \models \lambda$, it follows from Theorem 3 that $\gamma'_1 \vee ... \vee \gamma'_o \models \gamma_1 \vee ... \vee \gamma_k$ and that for every i there is some j such that $\chi'_i \models \psi_1 \vee ... \vee \psi_m \vee \chi_j$. We thereby obtain $\phi \models \lambda' \models \gamma_1 \vee ... \vee \gamma_k \vee \Diamond\psi'_1 \vee ... \vee \Diamond\psi'_p \vee \Box(\chi_1 \vee \psi_1 \vee ... \vee \psi_m) \vee ... \vee \Box(\chi_n \vee \psi_1 \vee ... \vee \psi_m)$. From this, we can infer that $\phi \wedge \neg(\gamma_1 \vee ... \vee \gamma_k \vee \Box(\chi_1 \vee \psi_1 \vee ... \vee \psi_m) \vee ... \vee \Box(\chi_n \vee \psi_1 \vee ... \vee \psi_m)) \models \Diamond\psi'_1 \vee ... \vee \Diamond\psi'_p \models \Diamond\psi_1 \vee ... \vee \Diamond\psi_m \not\models \Diamond\psi'_1 \vee ... \vee \Diamond\psi'_p$. As $\Diamond\psi_1 \vee ... \vee \Diamond\psi_m \equiv \Diamond(\psi_1 \vee ... \vee \psi_m)$, it follows that $\Diamond(\psi_1 \vee ... \vee \psi_m) \notin \Pi(\phi \wedge \neg(\gamma_1 \vee ... \vee \gamma_k \vee \Box(\chi_1 \vee \psi_1 \vee ... \vee \psi_m) \vee ... \vee \Box(\chi_n \vee \psi_1 \vee ... \vee \psi_m)))$.

Finally suppose that $\chi_i \not\models \chi'_j \vee \psi'_1 \vee ... \vee \psi'_p$ for all j and furthermore that $\psi_1 \vee ... \vee \psi_m \models \psi'_1 \vee ... \vee \psi'_p$ (we have already shown the result holds when $\psi_1 \vee ... \vee \psi_m \not\models \psi'_1 \vee ... \vee \psi'_p$). Now $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m)$ is an implicate of $\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\}))$ so to show that $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m)$ is not a prime implicate of $\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\}))$, we must find some stronger implicate. If $\phi \models \lambda \setminus \{\Box\chi_i\}$, then $\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\})) \models \bot$ so any contradictory clause is a stronger implicate than $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m)$. If instead we have $\phi \not\models \lambda \setminus \{\Box\chi_i\}$, then consider the clause $\bigvee_{s \in S} \Box\chi'_s$ where $S = \{s \in \{1, ..., q\} : \chi'_s \models \chi_i \vee \psi_1 \vee ... \vee \psi_m$ and $\chi'_s \not\models \chi_k \vee \psi_1 \vee ... \vee \psi_m$ for $k \neq i\}$. We note that there must be at least one element in S as $\phi \not\models \lambda \setminus \{\Box\chi_i\}$. Now since $\gamma'_1 \vee ... \vee \gamma'_o \models \gamma_1 \vee ... \vee \gamma_k$, $\psi'_1 \vee ... \vee \psi'_p \models \psi_1 \vee ... \vee \psi_m$, for every $s \notin S$ there is some $r \neq i$ such that $\chi'_s \models \chi_r \vee \psi_1 \vee ... \vee \psi_m$, and $\chi'_s \models \chi'_s$ for $s \in S$, we get $\phi \models \lambda' \models \gamma_1 \vee ... \vee \gamma_k \vee \Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee (\bigvee_{j \neq i} \Box\chi_j) \vee (\bigvee_{s \in S} \Box\chi'_s)$. It follows that $\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\}) \models \bigvee_{s \in S} \Box(\chi'_s \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m)$, which means that $\bigvee_{s \in S} \Box(\chi'_s \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m)$ is an implicate of $\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\})$. Moreover, $\bigvee_{s \in S} \Box(\chi'_s \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m) \models \Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m)$ since by construction $\chi'_s \models \chi_i \vee \psi_1 \vee ... \vee \psi_m$ for every $s \in S$.

It remains to be shown that $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m) \not\models \bigvee_{s \in S} \Box(\chi'_s \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m)$. Suppose for a contradiction that the contrary holds. Then $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m) \models \bigvee_{s \in S} \Box(\chi'_s \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m)$, so by Theorem 1, there must be some $s \in S$ for which $\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m \models \chi'_s \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m$. But then $\chi_i \models \chi'_s \vee \psi_1 \vee ... \vee \psi_m$, and thus $\chi_i \models \chi'_s \vee \psi'_1 \vee ... \vee \psi'_p$ since we have assumed $\psi_1 \vee ... \vee \psi_m \models \psi'_1 \vee ... \vee \psi'_p$. This contradicts our earlier assumption that $\chi_i \not\models \chi'_j \vee \psi'_1 \vee ... \vee \psi'_p$ for all j. Thus, we have shown that $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m) \not\models \bigvee_{s \in S} \Box(\chi'_s \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m)$, so $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_m) \notin \Pi(\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\}))$. □

Theorem 18 *Let ϕ be a formula of $\mathcal{K}$, and let $\lambda = \gamma_1 \vee ... \vee \gamma_k \vee \Diamond\psi_1 \vee ... \vee \Diamond\psi_n \vee \Box\chi_1 \vee ... \vee \Box\chi_m$ be a non-tautologous clause such that (a) $\chi_i \equiv \chi_i \vee \psi_1 \vee ... \vee \psi_n$ for all i, and (b) there is no literal l in λ such that $\lambda \equiv \lambda \setminus \{l\}$. Then $\lambda \in \Pi(\phi)$ if and only if the following conditions hold:*

1. $\gamma_1 \vee ... \vee \gamma_k \in \Pi(\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\}))$
2. $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_n) \in \Pi(\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\}))$ *for every i*
3. $\Diamond(\psi_1 \vee ... \vee \psi_n) \in \Pi(\phi \wedge \neg(\lambda \setminus \{\Diamond\psi_1, ..., \Diamond\psi_n\}))$

Proof. The forward direction was shown in Lemma 18.1. The other direction follows from Lemma 18.2 together with the hypothesis that $\chi_i \equiv \chi_i \vee \psi_1 \vee ... \vee \psi_n$ for all i (which ensures that $\phi \wedge \neg(\gamma_1 \vee ... \vee \gamma_k \vee \Box(\chi_1 \vee \psi_1 \vee ... \vee \psi_m) \vee ... \vee \Box(\chi_n \vee \psi_1 \vee ... \vee \psi_m)) \equiv \phi \wedge \neg(\lambda \setminus \{\Diamond\psi_1, ..., \Diamond\psi_n\}))$. □

Theorem 19 *Let ϕ be a formula of $\mathcal{K}$, and let γ be a non-tautologous propositional clause such that $\phi \models \gamma$ and such that there is no literal l in γ such that $\gamma \equiv \gamma \setminus \{l\}$. Then $\gamma \in \Pi(\phi)$ if and only if $\phi \not\models \gamma \setminus \{l\}$ for all l in γ.*

Proof. Consider a formula ϕ and a non-tautologous propositional clause λ such that $\phi \models \lambda$ and such that there is no literal l in λ such that $\lambda \equiv \lambda \setminus \{l\}$. Suppose that $\phi \models \lambda \setminus \{l\}$ for some l in λ.

As we know that $\lambda \not\equiv \lambda \setminus \{l\}$, it follows that $\lambda \setminus \{l\}$ is an implicate of ϕ which is strictly stronger than λ, so λ is not a prime implicate of ϕ. For the other direction, suppose that $\lambda \notin \Pi(\phi)$. Then it must be the case that there is some clause ρ such that $\phi \models \rho \models \lambda \not\models \rho$. Since $\rho \models \lambda$, it follows from Theorem 2 that each literal in ρ is a propositional literal of λ or is inconsistent. If all of the literals in ρ are inconsistent, then both ρ and ϕ must be inconsistent, so clearly $\phi \models \gamma \setminus \{l\}$ for every l in γ. Otherwise, ρ is equivalent to a propositional clause, and more specifically to a propositional clause containing only those literals appearing in λ (since $\rho \models \lambda$). As ρ is strictly stronger than λ, there must be some literal l in λ which does not appear in ρ. But that means $\rho \models \lambda \setminus \{l\}$ and so $\phi \models \lambda \setminus \{l\}$, completing the proof. □

Theorem 20 *Let ϕ be a formula of $\mathcal{K}$, and let $\lambda = \Box\chi$ be a non-tautologous clause such that $\phi \models \lambda$. Then $\lambda \in \Pi(\phi)$ if and only if there exists some term $T \in$**Dnf-4**(ϕ) such that $\chi \models \beta_T$, where β_T is the conjunction of formulae ψ such that $\Box\psi$ is in T.*

Proof. Let ϕ be some formula, and let $\lambda = \Box\chi$ be a non-tautologous clause such that $\phi \models \lambda$. For the first direction, suppose that there is no term $T \in$**Dnf-4**(ϕ) such that $\chi \models \beta_T$, where β_T is the conjunction of formulae ψ such that $\Box\psi$ is in T. There are two cases: either there are no terms in **Dnf-4**(ϕ) because ϕ is unsatisfiable, or there are terms but none satisfy the condition. In the first case, $\Box\chi$ is not a prime implicate of ϕ, since any contradictory clause (e.g. $\Diamond(a \wedge \neg a)$) is stronger. In the second case, consider the clause $\lambda' = \bigvee_T \Box\beta_T$, where β_T is the conjunction of formulae ψ such that $\Box\psi$ is in T. Now for every T we must have $\Box\beta_T \models \Box\chi$, otherwise we would have $T \not\models \Box\chi$, and hence $\phi \not\models \Box\chi$. Moreover, $\phi \models \bigvee_T \Box\beta_T$ since $T \models \Box\beta_T$ for every T. But by Theorem 1, $\Box\chi \not\models \bigvee_T \Box\beta_T$ since $\chi \not\models \beta_T$ for all T. So we have $\phi \models \lambda' \models \lambda \not\models \lambda'$, which means that λ is not a prime implicate of ϕ.

For the other direction, suppose that $\Box\chi$ is not a prime implicate of ϕ and that $\phi \not\models \bot$. Then **Dnf-4**(ϕ) is non-empty. As $\phi \models \Box\chi$, we must have $T \models \Box\chi$ for all $T \in$**Dnf-4**(ϕ), so $\bigvee_T \Box\beta_T$ also implies $\Box\chi$. We now show that $\bigvee_T \Box\beta_T$ is a prime implicate of T. We let κ be some clause which implies $\bigvee_T \Box\beta_T$. Now since $\kappa \models \bigvee_T \Box\beta_T$ it follows from Theorem 2 that $\kappa \equiv \Box\zeta_1 \vee ... \vee \Box\zeta_n$ for some formulae ζ_i. As $\phi \models \kappa$, we must have $T \models \Box\zeta_1 \vee ... \vee \Box\zeta_n$ for all $T \in$**Dnf-4**(ϕ). But that can only be the case if $\Box\beta_T \models \Box\zeta_1 \vee ... \vee \Box\zeta_n$ for all T, which means $\bigvee_T \Box\beta_T \models \Box\zeta_1 \vee ... \vee \Box\zeta_n$. As $\bigvee_T \Box\beta_T$ implies every implicate of ϕ that implies it, $\bigvee_T \Box\beta_T$ must be a prime implicate of ϕ. But this means that $\Box\chi \not\models \bigvee_T \Box\beta_T$, since we have assumed that $\Box\chi$ is not a prime implicate of ϕ. It follows from Theorem 1 that $\chi \not\models \beta_T$ for all $T \in$**Dnf-4**(ϕ). □

In order to show Theorem 21 we will need the following lemmas:

Lemma 21.1 *If $\Diamond\psi$ is not a prime implicate of ϕ with respect to **D4**, the algorithm* **Test$\Diamond$PI** *returns **no** on input ($\Diamond\psi$, ϕ).*

Proof. Suppose that $\Diamond\psi$ is not a prime implicate of ϕ. Then either $\Diamond\psi$ is not an implicate of ϕ (in which case the algorithm immediately returns **no**), or there must be some clause λ such that $\phi \models \lambda \models \Diamond\psi$ but $\Diamond\psi \not\models \lambda$. As $\lambda \models \Diamond\psi$, it follows from Theorem 2 that λ is equivalent to a disjunction of $\Diamond$-formulae, and hence to some clause $\Diamond\psi'$.

We know from Lemma A.2 that ϕ is equivalent to the disjunction of terms in **Dnf-4**(ϕ). It must thus be the case that $T_i \models \Diamond\psi'$ for all $T_i \in$ **Dnf-4**(ϕ). Since each T_i is a satisfiable conjunction of propositional literals and formulae of the forms $\Diamond\sigma$ and $\Box\sigma$, it follows that there

exists a set $\{\Diamond\eta_i, \Box\mu_{i,1}, ..., \Box\mu_{i,k(i)}\}$ of conjuncts of T_i such that $\Diamond(\eta_i \wedge \mu_{i,1} \wedge ... \wedge \mu_{i,k(i)}) \models \Diamond\psi'$, otherwise T_i would fail to imply $\Diamond\psi'$. Moreover, all of the elements of $\{\Diamond\eta_i, \Box\mu_{i,1}, ..., \Box\mu_{i,k(i)}\}$ must appear in the NNF of ϕ outside modal operators so the formulae $\eta_i, \mu_{i,1}, ..., \mu_{i,k(i)}$ must all be elements of the set $\mathcal{X}$. It is immediate that both

$$\Diamond \bigvee_i (\eta_i \wedge \mu_{i,1} \wedge ... \wedge \mu_{i,k(i)}) \models \Diamond\psi' \models \Diamond\psi \tag{1}$$

and

$$\Diamond\psi \not\models \Diamond \bigvee_i (\eta_i \wedge \mu_{i,1} \wedge ... \wedge \mu_{i,k(i)}) \tag{2}$$

The latter implies that the formula $\Diamond\psi \wedge \neg(\Diamond \bigvee_i (\eta_i \wedge \mu_{i,1} \wedge ... \wedge \mu_{i,k(i)}))$ must be consistent, which means that

$$\begin{aligned} & \psi \wedge \neg(\bigvee_i (\eta_i \wedge \mu_{i,1} \wedge ... \wedge \mu_{i,k(i)})) \\ \equiv \quad & \psi \wedge \bigwedge_i (\neg\eta_i \vee \neg\mu_{i,1} \vee ... \vee \neg\mu_{i,k(i)})) \end{aligned}$$

must be consistent as well. But then it must be the case that we can select for each i some $\sigma_i \in \{\eta_i, \mu_{i,1}, ..., \mu_{i,k(i)}\}$ such that $\psi \wedge \bigwedge_i \neg\sigma_i$ is consistent. Let S be the set of σ_i. The set S satisfies the condition of the algorithm since:

- $S \subseteq \mathcal{X}$
- $\psi \not\models \bigvee_{\sigma\in S} \sigma$ (because we know $\psi \wedge \bigwedge_i \neg\sigma_i$ to be consistent)
- for each $T_i \in$ **Dnf-4**(ϕ), we have found a set $\{\Diamond\eta_i, \Box\mu_{i,1}, ..., \Box\mu_{i,k(i)}\} \subseteq T_i$ such that:
  - $\{\eta_i, \mu_{i,1}, ..., \mu_{i,k(i)}\} \cap S \neq \emptyset$ (since S contains $\sigma_i \in \{\eta_i, \mu_{i,1}, ..., \mu_{i,k(i)}\}$)
  - $\Diamond(\eta_i \wedge \mu_{i,1} \wedge ... \wedge \mu_{i,k(i)}) \models \Diamond\psi$ (follows from (1) above)

Since there exists a set $S \subseteq \mathcal{X}$ satisfying these conditions, the algorithm returns **no**. □

Lemma 21.2 *If the algorithm* **Test◇PI** *returns* ***no*** *on input (◇ψ, φ), then ◇ψ is not a prime implicate of φ with respect to* ***D4****.*

Proof. There are two cases in which the algorithm returns **no**: either $\phi \not\models \Diamond\psi$, or there is some $S \subseteq \mathcal{X}$ which satisfies both conditions (a) and (b). In the first case, $\Diamond\psi$ is clearly not a prime implicate. We now examine the second case in more detail.

Suppose that there is some $S \subseteq \mathcal{X}$ satisfying:
(a) $\psi \not\models \bigvee_{\lambda\in S} \lambda$
(b) for each $T_i \in$ **Dnf-4**(ϕ), there exists $\{\Diamond\eta_i, \Box\mu_{i,1}, ..., \Box\mu_{i,k(i)}\} \subseteq T_i$ such that:
  (i) $\{\eta_i, \mu_{i,1}, ..., \mu_{i,k(i)}\} \cap S \neq \emptyset$
  (ii) $\Diamond(\eta_i \wedge \mu_{i,1} \wedge ... \wedge \mu_{i,k(i)}) \models \Diamond\psi$

Let α be the clause $\bigvee_i \Diamond(\eta_i \wedge \mu_{i,1} \wedge ... \wedge \mu_{i,k(i)})$. We remark that for each T_i, we have $T_i \models$

$\Diamond(\eta_i \wedge \mu_{i,1} \wedge ... \wedge \mu_{i,k(i)})$, and hence $\bigvee_i T_i \models \bigvee_i \Diamond(\eta_i \wedge \mu_{i,1} \wedge ... \wedge \mu_{i,k(i)})$. From the definition of **Dnf-4**(ϕ), we also have $\phi \equiv \bigvee_i T_i$. It immediately follows that $\phi \models \bigvee_i \Diamond(\eta_i \wedge \mu_{i,1} \wedge ... \wedge \mu_{i,k(i)})$ and hence $\phi \models \alpha$. From 2 (b) (ii), we have that $\Diamond(\eta_i \wedge \mu_{i,1} \wedge ... \wedge \mu_{i,k(i)}) \models \Diamond\psi$ for every i, and hence $\bigvee_i \Diamond(\eta_i \wedge \mu_{i,1} \wedge ... \wedge \mu_{i,k(i)}) \models \Diamond\psi$ which yields $\alpha \models \Diamond\psi$. From 2 (b) (i), we have that $\{\eta_i, \mu_{i,1}, ..., \mu_{i,k(i)}\} \cap S \neq \emptyset$ and hence that for every i there is some $\lambda \in S$ such that $\eta_i \wedge \mu_{i,1} \wedge ... \wedge \mu_{i,k(i)} \models \lambda$. From this we can infer that $\bigvee_i \Diamond(\eta_i \wedge \mu_{i,1} \wedge ... \wedge \mu_{i,k(i)}) \models \bigvee_{\lambda \in S} \Diamond\lambda$, and hence $\alpha \models \Diamond \bigvee_{\lambda \in S} \lambda$. But we know from 2 (a) and Theorem 1 that $\Diamond\psi \not\models \Diamond \bigvee_{\lambda \in S} \lambda$. It follows then that $\Diamond\psi \not\models \alpha$. Putting all this together, we see that there exists a clause α such that $\phi \models \alpha \models \Diamond\psi$ but $\Diamond\psi \not\models \alpha$, and hence that $\Diamond\psi$ is not a prime implicate of ϕ. □

Theorem 21 *Let ϕ be a formula, and let $\Diamond\psi$ be an implicate of ϕ. Then the algorithm* **Test$\Diamond$PI** *returns* ***yes*** *on input ($\Diamond\psi$, ϕ) if and only if $\Diamond\psi$ is a prime implicate of ϕ.*

Proof. It is clear that **Test$\Diamond$PI** terminates since unsatisfiability testing and the NNF transformation always terminate, and there are only finitely many S and T_i. Lemmas 21.1 and 21.2 show us that the algorithm always gives the correct response. □

Theorem 22 *The algorithm* **Test$\Diamond$PI** *runs in polynomial space.*

Proof. Checking whether $\phi \models \Diamond\psi$ can obviously be done in polynomial space in the length of $|\phi|$ and $|\psi|$ using standard unsatisfiability algorithms. We next remark that the sum of the lengths of the elements in $\mathcal{X}$ is bounded by the length of the formula $\textsc{Nnf}(\phi)$, and hence by Lemma A.3 the sum of the lengths of the elements of a particular $S \subseteq \mathcal{X}$ cannot exceed $2|\phi|$. Testing whether $\psi \not\models \bigvee_{\lambda \in S} \lambda$ can thus be accomplished in polynomial space in the length of ϕ and ψ as it involves testing the satisfiability of the formula $\psi \wedge \bigwedge_{\lambda \in S} \neg\lambda$ whose length is clearly polynomial in ϕ and ψ.

Now let us turn to Step 2 (b). We notice that it is not necessary to keep all of the T_i in memory at once, since we can generate and test the T_i one at a time. By Lemma A.3, the length of any T_i in **Dnf-4**(ϕ) can be at most $2|\phi|$. It follows that checking whether $\{\eta_i, \mu_{i,1}, ..., \mu_{i,k(i)}\} \cap S \neq \emptyset$, or whether $\Diamond(\eta_i \wedge \mu_{i,1} \wedge ... \wedge \mu_{i,k(i)}) \models \Diamond\psi$ can both be accomplished in polynomial space in the length of ϕ and ψ. We conclude that the algorithm **Test$\Diamond$PI** runs in polynomial space. □

In order to show Theorem 25, we use the following lemmas:

Lemma 25.1 *If λ is a clause that is not a prime implicate of ϕ, then* **TestPI** *outputs* ***no*** *on this input.*

Proof. Let us begin by considering a formula λ which is a clause but that is not a prime implicate of ϕ. There are two possible reasons for this: either λ is not an implicate of ϕ, or it is an implicate but there exists some stronger implicate. In the first case, **TestPI** returns **no** in Step 1, as desired. We will now focus on the case where λ is an implicate but not a prime implicate. We begin by treating the limit cases where one or both of ϕ and λ is a tautology or contradiction. Given that we know λ to be a non-prime implicate of ϕ, there are only two possible scenarios: either $\not\models \phi$ and $\models \lambda$, or $\phi \models \bot$ and $\lambda \not\models \bot$. In both cases, the algorithm returns **no** in Step 2.

If λ is an implicate of ϕ, and neither ϕ nor λ is a tautology or contradiction, then the algorithm will continue on to Step 3. In this step, any redundant literals will be deleted from λ, and if λ contains $\Diamond$-literals, we add an extra disjunct to the $\Box$-literals so that λ satisfies the

syntactic requirements of Theorem 18. Let $\gamma_1 \vee ...\gamma_k \vee \Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee \Box\chi_1 \vee ... \vee \Box\chi_n$ be the clause λ at the end of Step 3 once all modifications have been made. As the transformations in Step 3 are equivalence-preserving (Theorem 1), the modified λ is equivalent to the original, so λ is still a non-tautologous non-prime implicate of ϕ. This means ϕ and λ now satisfy all of the conditions of Theorem 18. It follows then that one of the following holds:

(a) $\gamma_1 \vee ... \vee \gamma_k \notin \Pi(\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\})$

(b) $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_n) \notin \Pi(\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\}))$ for some i

(c) $\Diamond(\psi_1 \vee ... \vee \psi_n) \notin \Pi(\phi \wedge \neg(\lambda \setminus \{\Diamond\psi_1, ..., \Diamond\psi_n\}))$

Suppose that (a) holds. Now $\gamma_1 \vee ... \vee \gamma_k$ is a non-tautologous propositional clause implied by $\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\})$ which contains no redundant literals. This means that $\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\})$ and $\gamma_1 \vee ... \vee \gamma_k$ satisfy the conditions of Theorem 19. According to this theorem, as $\gamma_1 \vee ... \vee \gamma_k \notin \Pi(\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\})$, then there must be some γ_j such that $\phi \wedge \neg(\lambda \setminus \{\gamma_1, ..., \gamma_k\}) \models \gamma_1 \vee ... \vee \gamma_{j-1} \vee \gamma_{j+1} \vee ... \vee \gamma_k$. This means that $\phi \models \lambda \setminus \{\gamma_j\}$, so the algorithm returns **no** in Step 4.

Suppose next that (b) holds, and let i be such that $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_n) \notin \Pi(\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\}))$. By Theorem 20, this means that there is no $T \in$**Dnf-4**(ϕ) such that $\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_n \models \beta_T$, so the algorithm returns **no** in Step 5.

Finally consider the case where neither (a) nor (b) holds but (c) does. Then in Step 6, we will call **Test**$\Diamond$**PI**$(\Diamond(\bigvee_{i=1}^{m} \psi_i), \phi \wedge \neg(\lambda \setminus \{\Diamond\psi_1, ..., \Diamond\psi_m\}))$. As $\Diamond(\bigvee_{i=1}^{m} \psi_i)$ is not a prime implicate of $\phi \wedge \neg(\lambda \setminus \{\Diamond\psi_1, ..., \Diamond\psi_m\}))$ and we have shown **Test**$\Diamond$**PI** to be correct (Theorem 21), **Test**$\Diamond$**PI** will return **no**, so the **TestPI** will return **no** as well. As we have covered each of the possible cases, we can conclude that if λ is a clause that is not a prime implicate of ϕ with respect to **D4**, then **TestPI** outputs **no**. □

Lemma 25.2 *If* **TestPI** *outputs* ***no*** *with input (λ, ϕ) and λ is a clause, then λ is not a prime implicate of ϕ.*

Proof. There are 5 different ways to return **no** (these occur in Steps 1, 2, 4, 5, and 6). Let us consider each of these in turn. The first way that the algorithm can return **no** is in Step 1 if we find that $\phi \not\models \lambda$. This is correct since λ cannot be a prime implicate if it is not a consequence of ϕ. In Step 2, we return **no** if ϕ is unsatisfiable but λ is not, or if λ is a tautology but ϕ is not. This is also correct since in both cases λ cannot be a prime implicate since there exist stronger implicates (any contradictory clause if $\phi \equiv \bot$, and any non-tautologous implicate of ϕ if $\lambda \equiv \top$). In Step 3, we may modify λ, but the resulting formula is equivalent to the original, and so it is a prime implicate just in the case that the original clause was. Let $\gamma_1 \vee ...\gamma_k \vee \Diamond\psi_1 \vee ... \vee \Diamond\psi_m \vee \Box\chi_1 \vee ... \vee \Box\chi_n$ be the clause at the end of Step 3. Now in Step 4, we return **no** if we find some propositional literal l in λ for which $\phi \models \lambda \setminus \{l\}$. Now since in Step 3, we have removed redundant literals from λ, we can be sure that $\lambda \setminus \{l\}$ is strictly stronger than λ. So we have $\phi \models \lambda \setminus \{l\} \models \lambda$ and $\lambda \not\models \lambda \setminus \{l\}$, which means that λ is not a prime implicate of ϕ. We now consider Step 5 of **TestPI**. In this step, we return **no** if for some literal $\Box\chi_i$ there is no term T_i in **Dnf-4**$(\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\}))$ for which $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_k)$ is equivalent to the conjunction of $\Box$-literals in T_i. According to Theorem 20, this means that $\Box(\chi_i \wedge \neg\psi_1 \wedge ... \wedge \neg\psi_k)$ is not a prime implicate of $\phi \wedge \neg(\lambda \setminus \{\Box\chi_i\})$, which means that λ is not a prime implicate of ϕ by

Theorem 18. Finally let us consider Step 6. In this step, we return **no** if **Test$\Diamond$PI** returns **no** on input $(\Diamond(\bigvee_{i=1}^{k}\psi_i), \phi \wedge \neg(\lambda \setminus \{\Diamond\psi_1, ..., \Diamond\psi_m\}))$. By Theorem 21, we know that this happens just in the case that $\Diamond(\bigvee_{i=1}^{k}\psi_i)$ is not a prime implicate of $\phi \wedge \neg(\lambda \setminus \{\Diamond\psi_1, ..., \Diamond\psi_m\})$. It follows from Theorem 18 that λ is not a prime implicate of ϕ. □

Theorem 25 *The algorithm* **TestPI** *always terminates, and it returns* ***yes*** *on input (λ, ϕ) if and only if λ is a prime implicate of ϕ.*

Proof. The algorithm **TestPI** clearly terminates because Steps 1 to 5 involve a finite number of syntactic operations on λ and a finite number of entailment checks. Moreover, the call to **Test$\Diamond$PI** in Step 6 is known to terminate (Theorem 21). Correctness and completeness have already been shown in Lemmas 25.1 and 25.2. □

We make use of the following lemma in the proof of Theorem 27:

Lemma 27.1 *The algorithm* **TestPI** *provided in Figure 4 runs in polynomial space in the length of the input.*

Proof. It is clear that steps (1) through (5) can be carried out in polynomial space in the length of the input, since they simply involve testing the satisfiability of formulae whose lengths are polynomial in $|\lambda| + |\phi|$. Step (6) can also be carried out in polynomial space since by Theorem 22 deciding whether the formula $\Diamond(\bigvee_{i=1}^{k}\psi_i)$ is a prime implicate of $\phi \wedge \neg(\lambda \setminus \{\psi_1, ..., \psi_k\}))$ takes only polynomial space in $|\Diamond(\bigvee_{i=1}^{k}\psi_i)| + |\phi \wedge \neg(\lambda \setminus \{\Diamond\psi_1, ..., \Diamond\psi_k\}))|$, and hence in $|\lambda| + |\phi|$. We can thus conclude that the algorithm **TestPI** runs in polynomial space in the length of the input. □

Theorem 27 *Prime implicate recognition is in* PSPACE.

Proof. We have show in Theorem 25 that **TestPI** always terminates and returns **yes** whenever the clause is a prime implicate and **no** otherwise. This means that **TestPI** is a decision procedure for prime implicate recognition. Since the algorithm has been shown to run in polynomial space (Lemma 27.1), we can conclude that prime implicate recognition is in PSPACE. □

Corollary 28 *Prime implicate recognition is* PSPACE*-complete.*

Proof. Follows directly from Theorems 16 and 27. □

References

[1] P. Adjiman, P. Chatalic, F. Goasdoué, M. Rousset, and L. Simon. Distributed reasoning in a peer-to-peer setting: Application to the semantic web. *Journal of Artificial Intelligence Research*, 25:269–314, 2006.

[2] F. Baader, D. L. McGuiness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2003.

[3] G. Bittencourt. Combining syntax and semantics through prime form representation. *Journal of Logic and Computation*, 2007. Advance Access published September 4, 2007, doi:10.1093/logcom/exm051.

[4] P. Blackburn, M. de Rijke, and Y. Venema. *Modal logic*. Cambridge University Press, 2001.

[5] S. Brandt and A.-Y. Turhan. An approach for optimized approximation. In *Proceedings of the KI-2002 Workshop on Applications of Description Logics (KIDLWS'01)*, 2002.

[6] M. Cadoli and F. Donini. A survery on knowledge compilation. *AI Communications*, 10(3-4):137–150, 1997.

[7] B. Chellas. *Modal logic: an introduction*. Cambridge University Press, 1980.

[8] M. Cialdea Mayer and F. Pirri. Propositional abduction in modal logic. *Logic Journal of the IGPL*, 3(6):907–919, 1995.

[9] A. Darwiche and P. Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[10] J. de Kleer, A. K. Mackworth, and R. Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56:197–222, 1992.

[11] F. Donini. *The Description Logic Handbook*, chapter Complexity of Reasoning. Cambridge University Press, 2003.

[12] F. M. Donini, B. Hollunder, M. Lenzerini, A. Marchetti Spaccamela, D. Nardi, and W. Nutt. The complexity of existential qualification in concept languages. *Artificial Intelligence*, 53:309–327, 1992.

[13] T. Eiter and K. Makino. On computing all abductive explanations. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI02)*, pages 62–67, 2002.

[14] C. Elsenbroich, O. Kutz, and U. Sattler. A case for abductive reasoning over ontologies. In *Proceedings of OWL: Experiences and Directions (OWLED'06)*, November 2006.

[15] P. Enjalbert and L. Fariñas del Cerro. Modal resolution in clausal form. *Theoretical Computer Science*, 65(1):1–33, 1989.

[16] M. Garey and D. Johnson. *Computers and intractability. A guide to the theory of NP-completeness*. W. H. Freeman, 1979.

[17] S. Ghilardi, C. Lutz, and F. Wolter. Did I damage my ontology? A case for conservative extensions in description logics. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR06)*, pages 187–197, 2006.

[18] F. Giunchiglia and R. Sebastiani. A sat-based decision procedure for ALC. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR96)*, pages 304–314, 1996.

[19] R. Kusters and R. Molitor. Approximating most specific concepts in logics with existential restrictions. *AI Communications*, (15):47–59, 2002.

[20] R. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal of Computing*, 6(3):467–480, 1977.

[21] Gerhard Lakemeyer. All you ever wanted to know about Tweety (but were afraid to ask). In *Proceedings of the Third International Conference on the Principles of Knowledge Representation and Reasoning (KR 92)*, pages 639–648, 1992.

[22] Gerhard Lakemeyer. A logical account of relevance. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 95)*, pages 853–861, 1995.

[23] J. Lang, P. Liberatore, and P. Marquis. Propositional independence: Formula-variable independence and forgetting. *Journal of Artificial Intelligence Research*, 18:391–443, 2003.

[24] P. Marquis. *Contribution á l'étude des méthodes de construction d'hypothèses en intelligence artificielle*. PhD thesis, Université de Nancy I, 1991.

[25] P. Marquis. Extending abduction from propositional to first-order logic. In *Proceedings of Fundamentals of Artificial Research Workshop*, pages 141–155, 1991.

[26] P. Marquis. *Handbook on Defeasible Reasoning and Uncertainty Management Systems*, volume 5, chapter Consequence Finding Algorithms, pages 41–145. Kluwer, 2000.

[27] Maurice Pagnucco. Knowledge compilation for belief change. In *Proceedings of the Australian Conference on Artificial Intelligence*, pages 90–99, 2006.

[28] C. Papadimitriou. *Computational Complexity*. Addison Welsey, 1994.

[29] T. C. Przymusinski. An algorithm to compute circumscription. *Artificial Intelligence*, 38(1):49–73, 1989.

[30] A. Ramesh and N. Murray. Computing prime implicants/implicates for regular logics. In *Proceedings of the 24th IEEE International Symposium on Multiple-Valued Logic*, pages 115–123, 1994.

[31] Klaus Schild. A correspondence theory for terminological logics: preliminary report. In *Proceedings of IJCAI-91, 12th International Joint Conference on Artificial Intelligence*, pages 466–471, 1991.

[32] A.-Y. Turhan and Y. Bong. Speeding up approximation with nicer concepts. In *Proceedings of the Twentieth International Description Logic Workshop (DL'07)*, 2007.