

Prime Implicates and Prime Implicants in Modal Logic

Meghyn Bienvenu

IRIT, Université Paul Sabatier
31062 Toulouse Cedex, France
bienvenu@irit.fr

Abstract

The purpose of this paper is to extend the notions of prime implicates and prime implicants to the basic modal logic \mathcal{K} . We consider a number of different potential definitions of clauses and terms for \mathcal{K} , which we evaluate with respect to their syntactic, semantic, and complexity-theoretic properties. We then continue our analysis by comparing the definitions with respect to the properties of the notions of prime implicates and prime implicants that they induce. We provide algorithms and complexity results for the prime implicate generation and recognition tasks for the two most satisfactory definitions.

Introduction

Prime implicates and prime implicants are important notions in artificial intelligence. They have been used in a number of different areas of AI, among them knowledge compilation (Darwiche & Marquis 2002), abduction and diagnosis (Marquis 2000), and non-monotonic reasoning (Marquis 2000).

Traditionally, these concepts have been studied in the context of propositional logic. To our knowledge, 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, it seems natural to wonder 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 (Elsbroich, Kutz, & Sattler 2006). 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 (KBs) more accessible to knowledge engineers and end

users alike. For instance, prime implicates might aid knowledge engineers in keeping track of what information is contained in the KB (“find all of the prime implicates concerning professors”) or in evaluating the consequences of adding new pieces of information (“find all prime implicates of $\text{KB} + \text{new}$ that are not prime implicates of KB ”). More refined notions of prime implicates (in which additional restrictions are placed on the clausal consequences) could be used as the basis for rich query languages that allow users to pose general questions about the contents of a KB like “What facts are known about penguins but not about birds in general?”.

In this paper, we examine the problem of defining prime implicates/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, 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 the problem of defining clauses and terms in \mathcal{K} . As there is no obvious definition, we enumerate a list of desirable properties which we use to compare different possible definitions. In the following section, we consider the notions of prime implicate and prime implicant induced by the various definitions. 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 consider the computational aspects of the most suitable definitions, providing algorithms for generating and recognizing prime implicates and some complexity results. Note that for lack of space straightforward proofs are omitted, and proof sketches are given for the more difficult results.

Preliminary Definitions and Notation

We assume basic familiarity with propositional logic, and in particular with the notions of literal (a propositional variable or its negation), clause (disjunction of literals), term (conjunction of literals), negation normal form (negation only directly before propositional symbols), conjunctive normal form (conjunction of clauses), disjunctive normal form (disjunction of terms), prime implicates (logically strongest

clausal consequences of a formula), and prime implicants (logically weakest terms implying a formula).

We briefly recall the basics of the modal logic \mathcal{K}^1 . Formulae in \mathcal{K} are built up from a set of propositional symbols, the standard logical connectives, and the modal operators \Box and \Diamond . The modal depth of a formula ϕ , written $\delta(\phi)$, is defined as the maximal number of nested modal operators appearing in ϕ , ex. $\delta(\Diamond(a \wedge \Box a) \vee a) = 2$. Negation normal form (NNF) is defined just as in propositional logic. Every formula in \mathcal{K} can be transformed into an equivalent formula in NNF (with the same modal depth and length) in linear time.

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, $\mathcal{R} \subset \mathcal{W} \times \mathcal{W}$ is a binary relation over worlds, and $v : \mathcal{W} \times \mathcal{P} \rightarrow \{true, false\}$ is a valuation of the propositional symbols at each world. Satisfaction of a formula ϕ in a model \mathcal{M} at the world w (denoted $\mathcal{M}, w \models \phi$) is defined as follows:

- $\mathcal{M}, w \models a$ iff $v(w, a) = true$
- $\mathcal{M}, w \models \neg\phi$ iff $\mathcal{M}, w \not\models \phi$
- $\mathcal{M}, w \models \phi \wedge \psi$ iff $\mathcal{M}, w \models \phi$ and $\mathcal{M}, w \models \psi$
- $\mathcal{M}, w \models \phi \vee \psi$ iff $\mathcal{M}, w \models \phi$ or $\mathcal{M}, w \models \psi$
- $\mathcal{M}, w \models \Box\phi$ iff $\mathcal{M}, w' \models \phi$ for all w' such that $w\mathcal{R}w'$
- $\mathcal{M}, w \models \Diamond\phi$ iff $\mathcal{M}, w' \models \phi$ for some w' such that $w\mathcal{R}w'$

A formula ϕ is satisfiable if $\exists \mathcal{M}, w$ such that $\mathcal{M}, w \models \phi$, and ϕ is valid (written $\models \phi$) if $\mathcal{M}, w \models \phi$ for all \mathcal{M} and w . A formula ψ is a logical consequence of a formula ϕ (written $\phi \models \psi$) if $\mathcal{M}, w \models \phi$ implies $\mathcal{M}, w \models \psi$ for every model \mathcal{M} and world $w \in \mathcal{W}$. We remark that $\phi \models \psi$ iff $\models \neg\phi \vee \psi$.

Satisfiability and unsatisfiability of formulae in \mathcal{K} can be decided in polynomial-space using tableaux algorithms. The basic idea behind these algorithms is to try and build a model for the formula: if we succeed, then the formula is satisfiable, and if we fail, the formula is unsatisfiable.

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

The notions of prime implicants and implicants are based upon the notions of clauses and terms. Thus, if we aim to provide suitable definitions of prime implicants and implicants for the logic \mathcal{K} , we first need to come up with a suitable definition of clauses and terms in \mathcal{K} . This is not trivial, however, as there is no generally accepted definition. 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. This will not only give us a means of comparing specific definitions but will also allow us to prove some general results concerning the space of possible definitions.

Each of the properties below describes something of what it is to be a literal, clause, or term in propositional logic. Although our list cannot be considered exhaustive, we do believe it covers the main 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 formulae 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 1. *Any definition of literals, clause, and terms for \mathcal{K} that satisfies properties **P1** and **P2** cannot satisfy **P5**, and vice-versa.*

The proof of Theorem 1 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.

We remark that each definition of clauses and terms gives rise to its own particular notion of conjunctive and disjunctive normal form. A formula in \mathcal{K} is said to be in conjunctive normal form (CNF) w.r.t. a definition D iff it is a conjunction of clauses (according to D). Disjunctive normal form (DNF) w.r.t. D is defined analogously. For parsimony, when D is clear from the context, we will write simply CNF or DNF.

The first definition that we will consider is that proposed in (Mayer & Pirri 1995) in the context of abduction. The authors define terms as the set of formulae built out of propositional literals using only \wedge , \Box , and \Diamond . Modal clauses and literals are not used in the paper but can be defined in exactly the same manner, resulting in the following definition²:

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

It is easy to see that this definition satisfies properties **P1**–**P2**, **P4**, and **P6**. Property **P7** also holds as clauses (terms) can be compared using a structural subsumption algorithm. Property **P3** is not satisfied since there are clauses that are not disjunctions of literals, $\Diamond(a \vee \Box a)$ for example. From Theorem 1, we know that property **P5** can not hold.

Using a slightly different definition, we can gain **P3**:

$$\begin{aligned} L &::= a \mid \neg a \mid \Box L \mid \Diamond L \\ \mathbf{D2} \quad C &::= L \mid C \vee C \end{aligned}$$

²Note that here and in what follows, we let a ranges over propositional variables and L , C , and T range over the sets of literals, clauses, and terms respectively.

¹Refer to (Blackburn, de Rijke, & Venema 2001) or (Chellas 1980) for good introductions to modal logic.

$$T ::= L \mid T \wedge T$$

It is easily verified that definition **D2** satisfies all the properties except **P5**.

We now consider the following definition of clauses that was proposed in (Enjalbert & Farinas Del Cerro 1989):

$$\begin{aligned} \mathbf{D3} \quad C &::= a \mid \neg a \mid \Box C \mid \Diamond \text{Conj} C \mid C \vee C \\ \text{Conj} C &::= C \mid \text{Conj} C \wedge \text{Conj} C \end{aligned}$$

There are two ways of extending this definition to literals and terms so as to satisfy a maximal subset of properties. The first extension is obtained by defining terms as the negations of clauses, i.e. by enforcing **P4**, and then defining literals as the intersection of clauses and terms:

$$\begin{aligned} \mathbf{D3a} = \mathbf{D3} + \quad L &::= a \mid \neg a \mid \Box L \mid \Diamond L \\ T &::= a \mid \neg a \mid \Box \text{Disj} T \mid \Diamond T \mid T \wedge T \\ \text{Disj} T &::= T \mid \text{Disj} T \vee \text{Disj} T \end{aligned}$$

This definition satisfies **P1**, and **P4-P6** (**P5** is a consequence of Proposition 1.3 in (Enjalbert & Farinas Del Cerro 1989)). It does not satisfy **P3** as there are clauses that are not disjunctions of literals – take for instance $\Box(a \vee b)$.

To show that **D3a** does not satisfy **P7**, we modify the polynomial-size translation of QBF into \mathcal{K} used to prove PSPACE-hardness (cf. section 6.7 of (Blackburn, de Rijke, & Venema 2001)) so that the translated formula is in CNF. We then notice that a formula ϕ in CNF 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 general idea is used to show PSPACE-completeness for definitions **D3b** and **D5** below.

If we instead choose to enforce **P3**, then we get the following definition:

$$\begin{aligned} \mathbf{D3b} = \mathbf{D3} + \quad L &::= a \mid \neg a \mid \Box C \mid \Diamond \text{Conj} C \\ T &::= L \mid T \wedge T \end{aligned}$$

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. In order to show that **P5** holds for definition **D3b**, we use standard logical equivalences to rewrite formulae into equivalent CNF and DNF (this is also what we do for **D4** and **D5**).

We now propose two rather simple definitions that satisfy **P3**, **P4**, and **P5**. The first definition, which is inspired by the notion of modal atom proposed in (Giunchiglia & Sebastiani 1996), defines literals as the set of formulae in NNF that cannot be decomposed propositionally.

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

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) iff $\Diamond\phi \models \Diamond(a \wedge \neg a)$.

Definition **D4** is very liberal, imposing no structure on the formulae behind modal operators. We can also obtain a definition satisfying exactly the same properties but which is much stricter. This definition defines literals as those formulae

in NNF that cannot be decomposed *modally*.

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

Although this definition is more restrictive than **D4**, it remains expressive enough to satisfy **P5**.

A summary of our analysis of the different definitions with respect to properties **P1-P7** is provided in Figure 1.

	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	yes	yes	no (PSPACE-complete)			

Figure 1: Properties of the different definitions of literals, clauses, and terms for the modal logic \mathcal{K} .

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 2. 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 3. 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 (resp. implicant) that we get will be determined by the definition of clause (resp. 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 (resp. implicants) of a formula is finite modulo logical equivalence.

Equivalence A formula is equivalent to its set of prime implicates (resp. implicants).

Finiteness ensures that the prime implicates/implicants of a formula can be finitely represented, while **Equivalence** guarantees that no information is lost in replacing a formula by its prime implicates/implicants.

Theorem 4. *If we define prime implicates (resp. implicants) using a notion of clause (resp. term) that falsifies **P5**, then at most one of **Finiteness** and **Equivalence** holds.*

As a corollary we have that the notions of prime implicates (implicants) induced by definitions **D1** and **D2** falsify at least one of **Finiteness** and **Equivalence**. In fact, it can be shown that it is **Equivalence** which is not satisfied.

We can show that the remaining definitions satisfy both **Finiteness** and **Equivalence**.

Theorem 5. *The notions of prime implicates and prime implicants induced by definitions **D3a**, **D3b**, **D4**, and **D5** satisfy **Finiteness** and **Equivalence**.*

Proof Sketch. Equivalence: We first prove (by a constructive inductive proof) that if λ is an implicate of ϕ , then there exists some implicate λ' of ϕ such that $\delta(\lambda') \leq \delta(\phi)$ and $\lambda' \models \lambda$. We use this plus the fact that there are only finitely many non-equivalent formulae on a finite alphabet and with fixed depth to show that there can be no infinite chains of increasingly stronger implicates of ϕ , and hence that every implicate of ϕ is implied by some prime implicate of ϕ . By **P5** we can write ϕ as an equivalent set of clauses, each of which is implied by some prime implicate of ϕ , and thus by the set of prime implicates of ϕ .

Finiteness: Every prime implicate of ϕ is equivalent to a clause built from the propositional variables in ϕ of depth at most $\delta(\phi)$, of which there are only finitely many. \square

Having determined the characteristics of each of the different definitions, we are now able to render our verdict. We first eliminate from consideration those definitions which fail to satisfy **Equivalence** since they do not give rise to suitable notions of prime implicates/implicants. Among the remaining definitions, we favor those which are as close as possible to the propositional definition. Thus, we prefer definitions **D4** and **D5** to definitions **D3a** and **D3b**, as **D3a** and **D3b** satisfy strictly less properties than **D4** and **D5**. Henceforth, we will restrict our attention to **D4** and **D5**.

It is important to note that while **D4** and **D5** are indistinguishable with respect to the properties that we have enumerated, they give rise to very different notions of prime implicates and implicants. For example, the prime implicates of $\Box(a \wedge b)$ are $\Box a$ and $\Box b$ w.r.t. **D5** but just $\Box(a \wedge b)$ itself if we use **D4**. More generally, we can show the prime implicates w.r.t. **D4** are at least as strong as those induced by **D5** and the prime implicants of **D4** are no weaker than those induced by **D5**.

We close this section with two additional properties of **D4** and **D5** which we will make use of in the next section.

Theorem 6 (Implicate-Implicant Duality). *Every prime implicate of a formula is equivalent to the negation of a prime implicant of the negated formula, and vice-versa.*

Theorem 7 (Distribution Property). *The prime implicates of a disjunction are equivalent to the logically strongest disjunctions of prime implicates of the disjuncts.*

Prime Implicate Generation and Recognition

In this section, we provide some first results concerning the computational aspects our definitions of prime implicates. We focus without loss of generality on prime implicates, since by Theorem 6, any algorithm for generating or recognizing prime implicates can be easily adapted into an algorithm for generating or recognizing prime implicants.

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 size of the formula. Since \mathcal{K} includes all of propositional logic, the same must be true of \mathcal{K} .

Figure 2 presents algorithms **GENPI-4** and **GENPI-5** which compute the sets of prime implicates w.r.t definitions **D4** and **D5** respectively. Our algorithms make use of the functions **DNF-4**(ϕ) and **DNF-5**(ϕ) which return a set of satisfiable terms (w.r.t. **D4** and **D5** respectively) whose disjunction is equivalent to ϕ^3 .

Function GENPI-4(ϕ)

- (1) If ϕ is unsatisfiable, return $\{\diamond(a \wedge \neg a)\}$. Otherwise, set $\mathcal{T} = \mathbf{DNF-4}(\phi)$.
- (2) For each $T \in \mathcal{T}$: let L_T be the set of propositional literals in T , let D_T be the set of formulae ζ such that $\diamond\zeta$ is in T , and let β_T be the conjunction of formulae ψ such that $\Box\psi \in$ is in T . Set $\Delta(T) = L_T \cup \{\Box\beta_T\} \cup \{\diamond(\zeta \wedge \beta_T) \mid \zeta \in D_T\}$.
- (3) Set $\mathbf{CANDIDATES} = \{\bigvee_{T \in \mathcal{T}} \theta_T \mid \theta_T \in \Delta(T)\}$.
- (4) For each $\lambda_j \in \mathbf{CANDIDATES}$: if $\lambda_k \models \lambda_j$ for some $\lambda_k \in \mathbf{CANDIDATES}$ with $k > j$, then delete λ_j from $\mathbf{CANDIDATES}$.
- (5) Return $\mathbf{CANDIDATES}$.

To get the algorithm **GENPI-5**, modify (2) so that

$$\Delta(T) = L_T \cup \{\Box\psi \mid \psi \in \mathbf{GENPI-5}(\beta_T)\} \cup \{\bigvee_{\gamma_i \in \mathbf{DNF-5}(\zeta \wedge \beta_T)} \diamond\gamma_i \mid \zeta \in D_T\}$$

Figure 2: Algorithms for prime implicate generation.

Both algorithms work in a similar manner. 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 (w.r.t. **D4**) whose disjunction is equivalent to ϕ . We know from the distribution property that every prime implicate of ϕ is equivalent to some disjunction of prime implicates of the terms in \mathcal{T} . In Step (2) we construct a set $\Delta(T)$ of clauses for each $T \in \mathcal{T}$ in such a way that every prime implicate of T is equivalent to some element 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 $\mathbf{CANDIDATES}$. During the comparison phase in Step (4), non-prime candidates are eliminated, and exactly one prime implicate for each equivalence class will be retained.

Obviously the definition of $\Delta(T)$ in Step (2) differs depending on which definition of prime implicates we are using. For **D4**, 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 (w.r.t. **D4**) must be equivalent to one of the elements in $\Delta(T)$ (note however that some elements in $\Delta(T)$ may not be prime implicates- take for instance the case where $\Box\beta_T$ is a tautology). Step (2) of **GENPI-5** is similar, but we must pay attention to the syntactic restrictions on clauses of

³The set of terms can be produced via a straightforward application of well-known logical equivalences ($(\varphi_1 \vee \varphi_2) \wedge \psi \equiv (\varphi_1 \wedge \psi) \vee (\varphi_2 \wedge \psi)$, $\diamond(\varphi_1 \vee \varphi_2) \equiv \diamond\varphi_1 \vee \diamond\varphi_2$, etc.), followed by a removal of any unsatisfiable terms.

D5. Essentially, this means that we must break $\Box\beta_T$ into its prime implicates (giving $\{\Box\psi \mid \psi \in \text{GENPI-5}(\beta_T)\}$), and we must rewrite the $\Diamond(\zeta \wedge \beta_T)$ as clauses by putting $\zeta \wedge \beta_T$ in DNF (w.r.t. **D5**) and then distributing the \vee over the \Diamond (giving $\bigvee_{\gamma_i \in \text{DNF-5}(\zeta \wedge \beta_T)} \Diamond\gamma_i$). It can be verified that every prime implicate of T (w.r.t. **D5**) is indeed equivalent to some clause in $\Delta(T)$. We illustrate Step (2) on an example:

Example 8. Set $T = a \wedge \Diamond b \wedge \Diamond(c \vee d) \wedge \Box e \wedge \Box f$. Then $L_T = \{a\}$, $D_T = \{b, c \vee d\}$, and $\beta_T = e \wedge f$. For **GENPI-4**, we get $\Delta(T) = \{a, \Box(e \wedge f), \Diamond(b \wedge e \wedge f), \Diamond((c \vee d) \wedge e \wedge f)\}$. For **GENPI-5**, we have $\Delta(T) = \{a, \Box e, \Box f, \Diamond(b \wedge e \wedge f), \Diamond(c \wedge e \wedge f) \vee \Diamond(d \wedge e \wedge f)\}$ because **GENPI-5**(β_T) = **GENPI-5**($e \wedge f$) = $\{e, f\}$, **DNF-5**($b \wedge e \wedge f$) = $\{b \wedge e \wedge f\}$, and **DNF-5**($(c \vee d) \wedge e \wedge f$) = $\{c \wedge e \wedge f, d \wedge e \wedge f\}$.

Theorem 9. Algorithms **GENPI-4** and **GENPI-5** always terminate, and they output exactly the set of prime implicates of the input formula (w.r.t. **D4** and **D5** respectively).

Corollary 10. The size of the smallest representation of a prime implicate of a formula is at most singly exponential in the size of the formula (using either definition **D4** or **D5**).

Our algorithms correspond to the simplest possible implementation of the distribution property, and it is well-known that naive implementations of the distribution property are computationally infeasible even for propositional logic. More efficient versions of our algorithms can be obtained using techniques developed for propositional logic, cf. (Marquis 2000). For instance, instead of generating all of the candidate clauses and *then* comparing them, we can build clauses incrementally, comparing them as we go.

We next consider the problem of recognizing prime implicates. In propositional logic, this problem is BH_2 -complete (Marquis 2000), 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 11. Prime implicate recognition is PSPACE-hard for both definitions **D4** and **D5**.

In order to obtain an upper bound, we exploit Corollary 10 which tells us that there is some polynomial p such that for every formula ϕ the size of its prime implicates is bounded by $2^{p(|\phi|)}$. This leads to a simple non-deterministic procedure for determining if a clause λ is a prime implicate of a formula ϕ . We simply guess a clause λ' of size at most $2^{p(|\phi|)}$ and check 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, λ is indeed a prime implicate.

Theorem 12. Prime implicate recognition is in EXPSpace for both definitions **D4** and **D5**.

In Figure 3 we present algorithms for prime implicate recognition for definitions **D4** and **D5**. Unlike our simple non-deterministic algorithms which blindly examine every

clause with less than a certain depth, the algorithms in Figure 3 make use of the information in the input formula and clause in order to cut down on the number of clauses to test. To simplify the presentation, we use the notation $\lambda \setminus \{l_1, \dots, l_n\}$ 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 .

Function TESTPI-4(λ, ϕ)
Input: a formula ϕ and a clause λ Output: **yes** or **no**

- (1) If $\phi \not\models \lambda$, return **no**.
- (2) If $\phi \models \lambda$, return **yes** if $\lambda \models \perp$ and **no** if not.
- (3) Preprocessing: For each literal l_i in $\lambda = l_1 \vee \dots \vee l_n$, test if $\lambda \setminus \{l_i\} \equiv \lambda$, and if so, remove l_i from λ .
- (4) For each l_i in λ , check if $\phi \models \lambda \setminus \{l_i\}$, and return **no** if so.
- (5) For each $l_i = \Box\gamma$ in λ , check whether there is some term T in **DNF-4**($\phi \wedge \neg(\lambda \setminus \{l_i\})$) for which l_i is equivalent to the conjunction of \Box -literals in T , and return **no** if not.
- (6) Let $\mathcal{D} = \{\Diamond\alpha_1, \dots, \Diamond\alpha_j\}$ be the set of \Diamond -literals in λ . If \mathcal{D} is empty, return **yes**, and otherwise let $\mathcal{T} = \text{DNF-4}(\phi \wedge \neg(\lambda \setminus \mathcal{D}))$. Build a formula $\omega = \Diamond\sigma_1 \vee \dots \vee \Diamond\sigma_k$ such that:
 - (i) for each $T \in \mathcal{T}$, $T \models \omega$
 - (ii) $\omega \models \Diamond\alpha_1 \vee \dots \vee \Diamond\alpha_j$
 - (iii) for each σ_p , there is some $T \in \mathcal{T}$ such that $\Diamond\sigma_p$ is a prime implicate of T (w.r.t. **D4**) and $T \not\models \sigma_q$ for $q \neq p$

Return **yes** if $(\Diamond\alpha_1 \vee \dots \vee \Diamond\alpha_j) \models \omega$, and **no** otherwise.

To obtain **TESTPI-5**, we replace (5) by

- (5') For each $l_i = \Box\gamma$ in λ , and each $T \in \text{DNF-4}(\phi \wedge \neg(\lambda \setminus \{l_i\}))$, call **TESTPI-5**(γ, ψ) where ψ is the conjunction of formulae ψ_l such that $\Box\psi_l$ is in T . Return **no** if there is some l_i for which **TESTPI-5** returns **no** for every T .

Figure 3: Algorithms for identifying prime implicates.

Example 13. We use **TESTPI-4** to test if the clauses $\lambda_1 = b$, $\lambda_2 = a \vee \Diamond a$, $\lambda_3 = \Box b \vee \Box(e \vee f)$, $\lambda_4 = \Diamond(a \wedge b)$, and $\lambda_5 = \Diamond(a \wedge b \wedge c) \vee \Diamond(a \wedge b \wedge e) \vee \Diamond(a \wedge b \wedge f) \vee \Diamond(a \wedge b \wedge c \wedge f)$ are prime implicates of $\phi = a \wedge (\Box(b \wedge c) \vee \Box(e \vee f)) \wedge \Diamond(a \wedge b)$.

- (1) $\phi \not\models \lambda_1$, so we output **no** for λ_1 .
- (2) $\phi \not\models \perp$, so we continue to step (3).
- (3) We delete $\Diamond(a \wedge b \wedge c \wedge f)$ from λ_5 .
- (4) We return **no** for λ_2 since $\phi \models \lambda_2 \setminus \{\Diamond a\}$.
- (5) We return **no** for λ_3 since **DNF-4**($\phi \wedge \neg(\lambda_3 \setminus \{\Box b\})$) = $\{a \wedge \Box(b \wedge c) \wedge \Diamond(a \wedge b)\}$ and $\Box b \not\models \Box(b \wedge c)$.
- (6) As λ_4 and λ_5 contain only \Diamond -literals, $\mathcal{T} = \text{DNF-4}(\phi) = \{a \wedge \Box(b \wedge c) \wedge \Diamond(a \wedge b), a \wedge \Box(e \vee f) \wedge \Diamond(a \wedge b)\}$. We set $\omega = \Diamond(a \wedge b \wedge c) \vee \Diamond(a \wedge b \wedge (e \vee f))$ and verify that (i), (ii), and (iii) are satisfied (w.r.t. both λ_4 and λ_5). We return **no** for λ_4 since $\lambda_4 \not\models \omega$, and **yes** for λ_5 since $\lambda_5 \models \omega$. Notice that if we hadn't removed $\Diamond(a \wedge b \wedge c \wedge f)$ from λ_5 in Step (3), then λ_5 would have been wrongly rejected in Step (4).

For **TESTPI-5**, everything is the same except for (5):

- (5') We do not return **no** for λ_3 since **DNF-4**($\phi \wedge \neg(\lambda_3 \setminus \{\Box b\})$) = $\{a \wedge \Box(b \wedge c) \wedge \Diamond(a \wedge b)\}$ and **TESTPI-5**($b, b \wedge c$) = **yes**, and **DNF-4**($\phi \wedge \neg(\lambda_3 \setminus \{\Box(e \vee f)\})$) = $\{a \wedge \Box(e \vee f) \wedge \Diamond(a \wedge b)\}$ and **TESTPI-5**($e \vee f, e \vee f$) = **yes**. In Step (6), we return **yes** for λ_3 because it contains no \Diamond -literals.

Theorem 14. The algorithms **TESTPI-4** and **TESTPI-5**

return **yes** if λ is a prime implicate of ϕ and **no** otherwise (with respect to definitions **D4** and **D5** respectively).

Proof Sketch. Steps (1) and Step (2) treat the limit cases where λ is not an implicate or ϕ is unsatisfiable. In Step (3), we remove all redundant disjuncts from λ . In Step (4), we test whether a proper sub-clause of λ is implied by ϕ . Because of Step (3), we know that if ϕ implies a sub-clause of λ , then λ is not a prime implicate of ϕ .

In Steps (5) and (5') we verify that the \square -literals in λ are as strong as possible. To do so, we check that each $\square\gamma$ in λ is a prime implicate of $\phi \wedge \neg(\lambda \setminus \{\square\gamma\})$ (if this is not the case, then we can replace $\square\gamma$ by a stronger \square -literal to get an implicate of ϕ which is strictly stronger than λ). We first remark that $\square\gamma$ is a prime implicate of $\phi \wedge \neg(\lambda \setminus \{\square\gamma\})$ just in the case that $\square\gamma$ is equivalent to some disjunction of prime implicates $\square\theta_i$ of the terms T_i in **DNF-4**($\phi \wedge \neg(\lambda \setminus \{\square\gamma\})$). We then notice that $\square\gamma \equiv \square\theta_1 \vee \dots \vee \square\theta_n$ just in the case that $\gamma \equiv \theta_i$ for some i . This means that all we need to do is check that $\square\gamma$ is equivalent to some prime implicate of some term T in **DNF-4**($\phi \wedge \neg(\lambda \setminus \{\square\gamma\})$). For (5), we remark that a \square -literal is a prime implicate (w.r.t. **D4**) of a term just in the case that it is equivalent to the conjunction of the \square -literals in the term. For (5'), we must call **TESTPI-5** in order to test whether each \square -literal in λ is equivalent to some prime implicate (w.r.t. **D5**) of some T_i in **DNF-4**($\phi \wedge \neg(\lambda \setminus \{\square\gamma\})$).

In Step (6), we check that the \diamond -literals in λ are as strong as possible. By construction, the formula ω is a prime implicate of $\phi \wedge \neg(\lambda \setminus \{\diamond\alpha_1, \dots, \diamond\alpha_j\})$ which implies $\diamond\alpha_1 \vee \dots \vee \diamond\alpha_j$. So $\diamond\alpha_1 \vee \dots \vee \diamond\alpha_j$ must also imply ω or else we can replace $\diamond\alpha_1 \vee \dots \vee \diamond\alpha_j$ by ω to get an implicate of ϕ which is strictly stronger than λ . \square

The most straightforward implementations of the algorithms in Figure 3 run in double-exponential time and singly exponential space since in Step (6) we build the formula ω whose size is exponential in $|\phi|$ and then test its satisfiability. It is possible however to obtain a singly exponential time implementation by exploiting the structure of the formula ω :

Theorem 15. *Prime implicate recognition is in EXPTIME (for **D4** and **D5**).*

Proof Sketch. The formula ω can be constructed by examining the $T_i \in \mathcal{T}$ one by one and adding a new disjunct $\diamond(\beta_{T_i} \wedge \zeta_{T_i})$ to ω whenever $T_i \not\models \omega$. We require that the disjunct $\diamond(\beta_{T_i} \wedge \zeta_{T_i})$ satisfies (a) $\zeta_{T_i} \in D_{T_i}$, (b) there is no $\zeta'_{T_i} \in D_{T_i}$ such that $\zeta'_{T_i} \models \zeta_{T_i}$ and $\zeta_{T_i} \not\models \zeta'_{T_i}$, and (c) $\diamond(\zeta_{T_i} \wedge \beta_{T_i}) \models \diamond\alpha_1 \vee \dots \vee \diamond\alpha_j$ (there must be such a ζ_{T_i} or else we would not have $\phi \models \lambda$). Clearly any ω constructed in this fashion must satisfy conditions (i), (ii), and (iii).

We remark that the \square - and \diamond -literals appearing in the $T \in \mathcal{T}$ must all already appear in the NNF of $\phi \wedge \neg(\lambda \setminus \mathcal{D})$, and thus there is a set Θ with $|\Theta| \leq |\phi| + |\lambda|$ such that every $\beta_{T_i} \wedge \zeta_{T_i}$ in ω is the conjunction of a subset of elements in Θ . Thus, we can determine whether some T_{j+1} implies the current ω by testing whether there is some $\zeta_{T_{j+1}} \in D_{T_{j+1}}$ such that $\zeta_{T_{j+1}} \wedge \beta_{T_{j+1}}$ is inconsistent with the set of negated elements of every subset $S \subseteq \Theta$ which contains at least one conjunct of every $\beta_{T_i} \wedge \zeta_{T_i}$ ($i \leq j$). This test can be accomplished in singly exponential time since there are at most

singly exponential subsets of Θ , and it takes at most singly exponential time to test that a subset contains a conjunct of each T_i and then again singly exponential time to test whether one of the (at most $|\phi| + |\lambda|$) possible $\zeta_{T_{j+1}} \wedge \beta_{T_{j+1}}$ is inconsistent with a subset's negated elements. This same method can be used to show that $(\diamond\alpha_1 \vee \dots \vee \diamond\alpha_j) \models \omega$ can be tested in singly exponential time. \square

We leave the proof of the exact complexity of these tasks as an interesting open problem.

Conclusion and Future Work

In this paper, we demonstrated that it is possible to define reasonable notions of prime implicates and prime implicants for the modal logic \mathcal{K} , and we provided algorithms for the prime implicate generation and recognition tasks.

In future work, we plan on extending our investigation to other modal and description logics, beginning with modal logics of knowledge and belief and expressive description logics used for the semantic web. We are also interested in studying other types of prime implicates, such as \mathcal{L} -prime implicates (the strongest implicates built from a fixed set of propositional symbols), Φ -prime implicates (the strongest new consequences of Φ when expanded by a formula), and n -prime implicates (the strongest implicates with depth at most n). These more fine-grained prime implicates allow for a richer notion of abduction (in which we can place additional restrictions on the candidate explanations) and could provide the basis for advanced querying facilities for modal and description logic knowledge bases.

Acknowledgements

I would like to thank Andreas Herzig, Jérôme Lang, and Jérôme Mengin for their many helpful comments.

References

- Blackburn, P.; de Rijke, M.; and Venema, Y. 2001. *Modal Logic*. Cambridge University Press.
- Chellas, B. 1980. *Modal logic: an introduction*. Cambridge University Press.
- Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research* 17:229–264.
- Elsenbroich, C.; Kutz, O.; and Sattler, U. 2006. A case for abductive reasoning over ontologies. In *Proceedings of OWL: Experiences and Directions*.
- Enjalbert, P., and Farinas Del Cerro, L. 1989. Modal resolution in clausal form. *Theoretical Computer Science* 65(1):1–33.
- Giunchiglia, F., and Sebastiani, R. 1996. A sat-based decision procedure for ALC. In *Proc. of KR-96*, 304–314.
- Marquis, P. 2000. *Handbook on Defeasible Reasoning and Uncertainty Management Systems*, volume 5. Kluwer, chapter Consequence Finding Algorithms, 41–145.
- Mayer, M., and Pirri, F. 1995. Propositional abduction in modal logic. *Logic Journal of the IGPL* 3(6):907–919.