Modal Resolution: Proofs, Layers and Refinements

CLÁUDIA NALON, University of Brasília
CLARE DIXON, University of Liverpool
ULLRICH HUSTADT, University of Liverpool

Resolution-based provers for multimodal normal logics require pruning of the search space for a proof in order to ameliorate the inherent intractability of the satisfiability problem for such logics. We present a clausal modal-layered hyper-resolution calculus for the basic multimodal logic, which divides the clause set according to the modal level at which clauses occur in order to reduce the number of possible inferences. We show that the calculus is complete for the logics being considered. We also show that the calculus can be combined with other strategies. In particular, we discuss the completeness of combining modal layering with negative and ordered resolution and provide experimental results comparing the different refinements.

CCS Concepts: •**Theory of computation** →**Modal and temporal logics; Automated reasoning;**

Additional Key Words and Phrases: Normal modal logics, Resolution method, Proof Strategies

ACM Reference format:
Cláudia Nalon, Clare Dixon, and Ullrich Hustadt. YYYY. Modal Resolution: Proofs, Layers and Refinements. *ACM Trans. Comput. Logic* VV, NN, Article AAA (YYYY), 38 pages.
DOI: 0000001.0000001

1 INTRODUCTION

Modal logics have long been used in Computer Science for describing and reasoning about complex systems, including programming languages [52], knowledge representation and reasoning [26, 53], verification of distributed systems [23, 24] and hardware [25]. The most basic normal multimodal language, known as K_n, extends the classical language with new operators, $\boxed{a}$ and $\langle a \rangle$, with $a \in A_n = \{1, \ldots, n\}$, a fixed finite set of indexes, where $\boxed{a}\varphi$ (resp. $\langle a \rangle\varphi$) reads as "φ is necessary (resp. possible) from the point of view of agent a", for a formula φ and index $a \in A_n$. This logic has received a great deal of attention, as it is able to express non-trivial problems in Artificial Intelligence and other areas. For instance, it is well-known that the description logic ALC [6], which has been applied to terminological representation, is a syntactic variant of K_n [56]. Problems in Quantified Boolean Logic, which is a very active area in the SAT community, can also be translated into K_n [26, 32] (and vice versa [49]).

The reasoning tasks for K_n are, however, far from trivial. The evaluation of modal formulae depends on a set of interpretations, also known as *possible worlds*, and a set of *accessibility relations* R_a over the set of worlds, for each $a \in A_n = \{1, \ldots, n\}$. Worlds and the accessibility relations define a structure known as a *Kripke model* (or *Kripke structure*). Given a Kripke model and a world w, a formula of the form $\boxed{a}\varphi$ is satisfied at w, if φ is satisfied at all worlds accessible from w through the relation R_a; and a formula of the form $\langle a \rangle\varphi$ is satisfied at w, if φ is satisfied at some world accessible from w via the relation R_a. Given a formula φ, the *local satisfiability problem* consists of showing that there is a world in a model that satisfies φ. A formula φ is *globally satisfiable* if there is a model such that all worlds in this model satisfy φ. Given a set of formulae Γ and a formula φ, the *local satisfiability of φ under the global constraints* Γ consists of showing that there is a model that globally satisfies the formulae in Γ and that there is a world in this model that satisfies φ. The local satisfiability problem for the multimodal propositional case is PSPACE-complete [26]. The global satisfiability and the local satisfiability under global constraint problems for K_n are EXPTIME-complete [63]. Given the inherent intractability of the reasoning problems and also the wide range of applications to which those logics can be applied, the development of automatic, efficient tools for theorem proving is highly desirable.

Several proof methods and tools for reasoning in K_n exist, either in the form of methods applied directly to the modal language or obtained by translation into more expressive languages (First-Order Logic, for instance). In this case, some information about the structure of the formula, which is evident in the original logic, may be lost in the translated version of the problem. Translation-based methods may benefit not only from the existence of available theorem provers, therefore not requiring a lot of effort for implementation, but also from the fact that strategies available for the object language can be almost immediately applied to the translated problem [28]. However, the translation into a more expressive logic combined with a standard proof method for that logic may incur some computational overhead that a direct method does not have and the combination may not necessarily be a decision procedure for the set of translated modal formulae [28]. On the other hand, the design of direct proof methods for modal logics requires the development of strategies to deal with the underlying normal forms and inference rules.

Here we present a novel resolution-based proof method for the modal logic K_n. The approach is clausal: formulae are firstly translated into a specific normal form before the set of inference rules can be applied. The design of the calculus was primarily motivated by efficiency and practical aspects of its implementation: the transformation into the normal form can be implemented in linear time on the size of the input formula; clauses sets already into normal form do not need reprocessing; the structure of the clauses help to build efficient structures of indexes for retrieving candidates for the resolution rules and its refinements. The structure of the normal form restricts resolution inferences whilst remaining complete. The resulting implementation of the calculus, K_SP [41, 43, 44], incorporates several resolution strategies and refinements and outperforms other modal provers for formulae that have a high degree of nesting of modal formulae. This paper extends the results in [40] providing full details of the calculus and proofs of soundness, completeness and termination. Further, we investigate the completeness of two different refinements, negative and ordered resolution, and also provide new experimental results concerning the performance of the prover using the different refinements.

The present method builds on previous work in several aspects. Firstly, as in our previous work [39], the calculus requires a translation into a more expressive modal language, but instead of using the universal modality, here labels are used to express semantic properties of a formula. That is, in [39], new propositional symbols are introduced to represent subformulae and the definition of these symbols as the formulae they replace holds at every reachable world. Here, labels prefixing a formula are used to make precise that the new propositional symbols and the definition of these symbols hold only where they are needed: at the worlds whose distance from the root of the model corresponds to the modal nesting in which they occur. We take advantage of the fact that K_n not only enjoys the finite and tree model properties [14], but that the satisfiability of subformulae of a formula only depends on the *modal layer*, that is the distance from the root in the model, where they occur [2]. Moreover, the modal layer where a subformula is satisfied corresponds to the *modal level* of that subformula, that is the number of modal operators in which scope the subformula occurs. Labels in the extended language given here correspond to the modal level of a formula. Secondly, the calculus makes use of labelled resolution in order to avoid unnecessary applications of the inference rules. For instance, in the unrestricted resolution method for K_n [39], the translation of $\diamondsuit_a \diamondsuit_a p \wedge \boxed{a}\neg p$ into the normal form results in the set $\{\mathbf{start} \Rightarrow t_0, t_0 \Rightarrow \diamondsuit_a t_1, t_1 \Rightarrow \diamondsuit_a p, t_0 \Rightarrow \boxed{a}\neg p\}$. From $t_1 \Rightarrow \diamondsuit_a p$ and $t_0 \Rightarrow \boxed{a}\neg p$, the clause $\mathbf{true} \Rightarrow \neg t_0 \vee \neg t_1$ is derived. This inference step is not necessary, as $\diamondsuit_a p$ and $\boxed{a}\neg p$ occur at different modal levels and are not, in fact, contradicting each other. The translation into the new normal form results in the set $\{0 : t_0, 0 : t_0 \Rightarrow \diamondsuit_a t_1, 1 : t_1 \Rightarrow \diamondsuit_a p, 0 : t_0 \Rightarrow \boxed{a}\neg p\}$. Resolution cannot be applied to the clauses $1 : t_1 \Rightarrow \diamondsuit_a p$ and $0 : t_0 \Rightarrow \boxed{a}\neg p$ as their labels are not the same.

The translation into the labelled normal form leads to the direct implementation of the layered modal heuristic given in [2]. However, in [2] the modal levels are hard-coded in the names of the translated propositional symbols, making the application of both local and global reasoning more difficult. Besides, using our approach, we can easily partition the clause set, restricting the application of the inference rules to (possibly) smaller sets, which may improve the performance of reasoners [60].

Labelling is widely used in proof methods for modal and other non-classical logics [11, 13, 17, 18, 67, 68]. Although it is not particularly common within resolution-based proof methods, this technique has also been used in some of those proof systems. For instance, in [5], a labelled non-clausal resolution-based proof method for K_n and ALC is given. Formulae are labelled by either constants a, which correspond to names of worlds in a model, or by pairs (a, b) representing the relation between two worlds named by a and b, respectively. Our calculus is similar, but labels correspond to modal levels instead of worlds. Having worlds as labels might require repeated applications of global reasoning for worlds at the same modal level. Labelled resolution is also used in e.g. [8, 9, 22], where (sets of) labels express the semantic constraints in multi-valued logics. For the calculus presented here, we have chosen to keep the labels simple so unification only requires a simple check.

The paper is organised as follows. Section 2 presents the language of K_n. The normal form and the modal-layered calculus are presented in Section 3 and Section 4. Correctness is proved in Section 5. In Section 6, we introduce two refinements for the presented calculus. In Section 6.1, we show that the application of the calculus can be restricted to negative resolution. Ordering refinements, discussed in Section 6.2, are not complete in general. We show, however, that there is a particular ordering for which the calculus given here is complete. Section 7 briefly describes the implementation of the calculus, and shows the results of the experimental evaluation of the prover K_SP using the different refinements and strategies. Section 8 considers related work and Section 9 provides conclusions and further work.

This paper extends [40], the implementation of the prover based on this calculus is presented in [41, 43, 44] and the software is available at [42].

2 LANGUAGE

The set $\mathsf{WFF}_{\mathsf{K}_n}$ of *well-formed formulae* of the logic K_n is constructed from a denumerable set of *propositional symbols*, $P = \{p, q, p', q', p_1, q_1, \ldots, t, t', t_0, t_1, \ldots\}$, the constants **true** and **false**, the negation symbol $\neg$, the conjunction symbol $\wedge$, the disjunction symbol $\vee$, the implication symbol $\Rightarrow$, and the unary connectives $\boxed{a}$ and $\diamondsuit_a$ for each index a in a finite, fixed set $A_n = \{1, \ldots, n\}$, $n \in \mathbb{N}$.

Definition 2.1. The *set of well-formed formulae*, $\mathsf{WFF}_{\mathsf{K}_n}$, is the least set such that $p \in P$ is in $\mathsf{WFF}_{\mathsf{K}_n}$; the constants **true** and **false** are in $\mathsf{WFF}_{\mathsf{K}_n}$; if φ and ψ are in $\mathsf{WFF}_{\mathsf{K}_n}$, then so are $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \Rightarrow \psi)$, $\boxed{a}\varphi$, and $\diamondsuit_a \varphi$ for each $a \in A_n$.

Instead of restricting ourselves to a minimal, expressively complete set of connectives, we have chosen to consider all usual operators as primitive, as they are all used in the normal form presented in Section 3. When $n = 1$, we often omit the index, that is, $\square\varphi$ stands for $\boxed{1}\varphi$. A *literal* is either a propositional symbol or its negation; the set of literals is denoted by L. We denote by $\neg l$ the *complement* of the literal $l \in L$, that is, $\neg l$ denotes $\neg p$ if l is the propositional symbol p, and $\neg l$ denotes p if l is the literal $\neg p$. A *modal literal* is either $\boxed{a}l$ or $\Diamond_{a} l$, where $l \in L$ and $a \in A_n$. The modal depth of a formula is recursively defined as follows:

Definition 2.2. Let $\varphi, \psi \in \mathsf{WFF}_{\mathsf{K}_n}$ be well-formed formulae. We define $\mathsf{mdepth} : \mathsf{WFF}_{\mathsf{K}_n} \longrightarrow \mathbb{N}$ as $\mathsf{mdepth}(p) = \mathsf{mdepth}(\mathbf{true}) = \mathsf{mdepth}(\mathbf{false}) = 0$, for constants and propositional symbols $p \in P$; $\mathsf{mdepth}(\neg\varphi) = \mathsf{mdepth}(\varphi)$; $\mathsf{mdepth}(\varphi \wedge \psi) = \mathsf{mdepth}(\varphi \vee \psi) = \mathsf{mdepth}(\varphi \Rightarrow \psi) = \max(\mathsf{mdepth}(\varphi), \mathsf{mdepth}(\psi))$; and $\mathsf{mdepth}(\boxed{a}\varphi) = \mathsf{mdepth}(\Diamond_{a}\varphi) = 1 + \mathsf{mdepth}(\varphi)$.

The modal level of a subformula is given relative to its position in the syntactic tree.

Definition 2.3. Let φ, φ' be well-formed formulae. Let Σ be the alphabet $\{1, 2\}$ and Σ^* the set of all finite sequences over Σ. Denote by ε the empty sequence over Σ. Let $\tau : \mathsf{WFF}_{\mathsf{K}_n} \times \Sigma^* \times \mathbb{N} \longrightarrow \mathcal{P}(\mathsf{WFF}_{\mathsf{K}_n} \times \Sigma^* \times \mathbb{N})$ be the function inductively defined as follows (where $\lambda \in \Sigma^*$, $ml \in \mathbb{N}$):

- $\tau(\mathbf{true}, \lambda, ml) = \{(\mathbf{true}, \lambda, ml)\}$;
- $\tau(\mathbf{false}, \lambda, ml) = \{(\mathbf{false}, \lambda, ml)\}$;
- $\tau(p, \lambda, ml) = \{(p, \lambda, ml)\}$, for $p \in P$;
- $\tau(\neg\varphi, \lambda, ml) = \{(\neg\varphi, \lambda, ml)\} \cup \tau(\varphi, \lambda.1, ml)$;
- $\tau(\varphi \wedge \varphi', \lambda, ml) = \{(\varphi \wedge \varphi', \lambda, ml)\} \cup \tau(\varphi, \lambda.1, ml) \cup \tau(\varphi', \lambda.2, ml)$;
- $\tau(\varphi \vee \varphi', \lambda, ml) = \{(\varphi \vee \varphi', \lambda, ml)\} \cup \tau(\varphi, \lambda.1, ml) \cup \tau(\varphi', \lambda.2, ml)$;
- $\tau(\varphi \Rightarrow \varphi', \lambda, ml) = \{(\varphi \Rightarrow \varphi', \lambda, ml)\} \cup \tau(\varphi, \lambda.1, ml) \cup \tau(\varphi', \lambda.2, ml)$;
- $\tau(\boxed{a}\varphi, \lambda, ml) = \{(\boxed{a}\varphi, \lambda, ml)\} \cup \tau(\varphi, \lambda.1, ml + 1)$;
- $\tau(\Diamond_{a}\varphi, \lambda, ml) = \{(\Diamond_{a}\varphi, \lambda, ml)\} \cup \tau(\varphi, \lambda.1, ml + 1)$.

The function τ applied to $(\varphi, \varepsilon, 0)$ returns the *annotated syntactic tree for* φ, where each node is uniquely identified by a subformula, its path order (or its position) in the tree, and its modal level. For instance, p occurs twice in the formula $\boxed{a}\boxed{a}(p \wedge \boxed{a}p)$, at the position 111 and modal level 2, and also at the position 1121 and modal level 3.

Definition 2.4. Let φ be a formula and let $\tau(\varphi, \varepsilon, 0)$ be its annotated syntactic tree. If $(\varphi', \lambda, ml) \in \tau(\varphi, \varepsilon, 0)$, then $\mathsf{mlevel}(\varphi, \lambda) = ml$.

If $\mathsf{mlevel}(\varphi, \lambda) = ml$ we say that the subformula φ' at the position λ of φ occurs at the modal level ml. In the example above, we have that p occurs at the modal levels 2 and 3. We note that the modal depth of a formula φ can be related to the maximal modal level ml for any subformula φ' of φ, i.e. $\mathsf{mdepth}(\varphi) = \max\{ml \mid (\varphi', \lambda, ml) \in \tau(\varphi, \epsilon, 0)\}$.

We present the semantics of K_n, as usual, in terms of Kripke structures.

Definition 2.5. A *Kripke model M for n agents over P* is given by a tuple $(W, w_0, R_1, \ldots, R_n, \pi)$, where W is a set of possible *worlds* with a distinguished world w_0, each R_a, $a \in A_n = \{1, \ldots, n\}$, is a binary relation on W, and $\pi : W \longrightarrow (P \longrightarrow \{\mathit{true}, \mathit{false}\})$ is a function which associates to each world $w \in W$ a truth-assignment to propositional symbols.

We write $\langle M, w\rangle \models \varphi$ (resp. $\langle M, w\rangle \not\models \varphi$) to say that φ is satisfied (resp. not satisfied) at the world w in the Kripke model M.

Definition 2.6. Let φ, ψ be well-formed formulae. Satisfaction of a formula at a given world w of a model M is inductively defined by:

- $\langle M, w\rangle \models \mathbf{true}$;
- $\langle M, w\rangle \not\models \mathbf{false}$;
- $\langle M, w\rangle \models p$ if, and only if, $\pi(w)(p) = \mathit{true}$, where $p \in P$;
- $\langle M, w\rangle \models \neg\varphi$ if, and only if, $\langle M, w\rangle \not\models \varphi$;
- $\langle M, w\rangle \models (\varphi \wedge \psi)$ if, and only if, $\langle M, w\rangle \models \varphi$ and $\langle M, w\rangle \models \psi$;
- $\langle M, w\rangle \models (\varphi \vee \psi)$ if, and only if, $\langle M, w\rangle \models \varphi$ or $\langle M, w\rangle \models \psi$;
- $\langle M, w\rangle \models (\varphi \Rightarrow \psi)$ if, and only if, $\langle M, w\rangle \models \neg\varphi$ or $\langle M, w\rangle \models \psi$;
- $\langle M, w\rangle \models \boxed{a}\,\varphi$ if, and only if, for all w', wR_aw' implies $\langle M, w'\rangle \models \varphi$;
- $\langle M, w\rangle \models \Diamond\!\!\!\!\raisebox{0.3ex}{\scriptsize a}\;\varphi$ if, and only if, there is w', wR_aw' and $\langle M, w'\rangle \models \varphi$.

Let $M = (W, w_0, R_1, \ldots, R_n, \pi)$ be a model. For local satisfiability, formulae are interpreted with respect to the root of M, that is, w_0. A formula φ is *locally satisfied in* M, denoted by $M \models_L \varphi$, if $\langle M, w_0\rangle \models \varphi$. The formula φ is *locally satisfiable* if there is a model M such that $\langle M, w_0\rangle \models \varphi$. A formula φ is *globally satisfied in* M, if for all $w \in W$, $\langle M, w\rangle \models \varphi$. A formula φ is said to be *globally satisfiable* if there is a model M such that M globally satisfies φ, denoted by $M \models_G \varphi$. Satisfiability of sets of formulae is defined as usual. For a set of formulae Γ, a formula φ is *satisfiable under the global constraints* Γ if there is a model M such that $M \models_G \Gamma$ and $M \models_L \varphi$.

A model $M = (W, w_0, R_1, \ldots, R_n, \pi)$ is *tree-like* if $\bigcup_{a=1}^{n} R_a$ is a tree, i.e. a directed acyclic graph (with root w_0). When considering local satisfiability, the following holds (see, for instance, [26]):

Theorem 2.7. *Let $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$ be a formula and $M = (W, w_0, R_1, \ldots, R_n, \pi)$ be a model. $M \models_L \varphi$ if, and only if, there is a tree-like model M' such that $M' \models_L \varphi$. Moreover, M' is finite and its depth is bounded by* $\mathsf{mdepth}(\varphi)$.

The proof of Theorem 2.7 shows that given a formula φ and a model M, such that $M \models_L \varphi$, then a tree-like model M' for φ can be built by unravelling up to the modal depth of φ the original model M. Given a tree-like model $M = (W, w_0, R_1, \ldots, R_n, \pi)$, we denote by $\mathsf{depth}(w)$ the length of a path from w_0 to w through the union of the relations in M. A *modal layer* is the equivalence class of worlds at the same distance from the root of a tree-like model. The next result also holds.

Theorem 2.8. *Let φ be a modal formula, and M be a tree (or tree-like) model with root w such that $\langle M, w\rangle \models \varphi$. Let ψ be a subformula of φ which occurs in the modal level l and which has modal depth k. To determine the truth value of ψ we only need to consider nodes at tree depth i, where $l \leq i \leq k + l$.*

Theorem 2.8 is adapted from [2, Proposition 3.2]. The proof is by induction on the structure of a formula and shows that a subformula φ' of φ, with $(\varphi', \lambda, ml) \in \tau(\varphi, \varepsilon, 0)$, is satisfied at a node with distance ml of the root of the tree-like model, i.e. that there is a correspondence between the syntactic notion of the modal level of a subformula and the semantic notion of the modal layer at which the subformula is satisfied. As determining the satisfiability of a formula depends only on its subformulae, only the subtrees of height $\mathsf{mdepth}(\varphi')$ starting at level ml need to be checked. The bound on the height of the subtrees follows from Theorem 2.7.

The global satisfiability problem for a (first-order definable) modal logic can be given in terms of the local satisfiability problem of a logic obtained by adding the universal modality, $\boxed{*}$, to the original language [19, Proposition 2.1]. Let K_n^* be the logic obtained by adding $\boxed{*}$ to K_n. Let $M = (W, w_0, R_1, \ldots, R_n, \pi)$ be a tree-like model for K_n. A model M^* for K_n^* is the pair (M, R_*), where $R_* = W \times W$. A formula $\boxed{*}\,\varphi$ is locally satisfied at the world w in the model M^*, written $\langle M^*, w\rangle \models_L \boxed{*}\,\varphi$, if, and only if, for all $w' \in W$, we have that $\langle M^*, w'\rangle \models \varphi$. Given these definitions, for φ in $\mathsf{WFF}_{\mathsf{K}_n}$, it follows that $M \models_G \varphi$ if, and only if, $M^* \models_L \boxed{*}\,\varphi$. So, instead of deciding $M \models_G \varphi$, we can decide $M^* \models_L \boxed{*}\,\varphi$.

We note that although the full language of K_n^* enjoys the finite model property (it is satisfied in a model which is exponential in the size of the original formula [63]), it does not retain the *finite* tree model property. More precisely, if a formula is satisfied in M^*, then the unravelling of this model might yield an infinite tree-like model. For instance, $\boxed{*}(p \Rightarrow \neg \boxed{*}\, p) \wedge \boxed{*}(\neg p \Rightarrow \neg \boxed{*} \neg p)$ cannot be satisfied in any finite tree-like structure [35].

3 LAYERED NORMAL FORM

A formula to be tested for local or global satisfiability is first translated into a normal form called *Separated Normal Form with Modal Levels*, SNF_{ml}, whose language extends that of K_n with labels for modal levels. Informally, we write $ml : \varphi$ to denote that a formula φ occurs at the modal level $ml \in \mathbb{N} \cup \{*\}$. By $* : \varphi$ we mean that φ occurs at all modal levels. Formally, let $\mathsf{WFF}_{\mathsf{K}_n}^{ml}$ be the set of formulae $ml : \varphi$ such that $ml \in \mathbb{N} \cup \{*\}$ and $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$. Let $M^* = (W, w_0, R_1, \ldots, R_n, R_*, \pi)$ be a model and $\varphi, \varphi' \in \mathsf{WFF}_{\mathsf{K}_n}$. Satisfaction of labelled formulae is defined as follows:

- $M^* \models ml : \varphi$ if, and only if, for all worlds $w \in W$ such that $\mathsf{depth}(w) = ml$, we have that $\langle M^*, w\rangle \models \varphi$;
- $M^* \models * : \varphi$ if, and only if, $M^* \models \boxed{*}\, \varphi$;
- $M^* \models (ml : \varphi) \wedge (ml' : \varphi')$ if, and only if, $M^* \models ml : \varphi$ and $M^* \models ml' : \varphi'$.

The labels in a formula in $\mathsf{WFF}_{\mathsf{K}_n}^{ml}$ work as a kind of *weak* universal operator, allowing us to talk about formulae that are satisfied at all worlds in a given modal layer. We note that $(ml : \varphi) \wedge (ml : \varphi')$ is semantically equivalent to $(ml : \varphi \wedge \varphi')$, a property which we will use later in our proofs.

A formula in SNF_{ml} is a conjunction of clauses labelled by the modal level in which they occur. Clauses in SNF_{ml} are in one of the following forms:

• Literal clause	$ml : \bigvee_{b=1}^{r} l_b$
• Positive a-clause	$ml : l' \Rightarrow \boxed{a}\, l$
• Negative a-clause	$ml : l' \Rightarrow \diamondsuit_a\, l$

where $ml \in \mathbb{N} \cup \{*\}$ and $l, l', l_b \in L$, $1 \leq b \leq r$, $r \in \mathbb{N}$. Positive and negative a-clauses are together known as *modal a-clauses*; the index a may be omitted if it is clear from the context. We require that clauses are kept in simplified form, that is, the simplification rules given in Table 2, with the obvious adaptations for labelled clauses, are applied. In particular, if $ml \in \mathbb{N} \cup \{*\}$, D is a (possibly empty) disjunction of literals, and $l \in L$, then: $ml : D \vee l \vee l$ simplifies to $ml : D \vee l$; $ml : D \vee \mathbf{false}$ simplifies to $ml : D$; and $ml : D \vee l \vee \neg l$ and $ml : D \vee \mathbf{true}$ simplify to $ml : \mathbf{true}$. We regard literal clauses as set of literals, that is, two clauses are the same if they contain the same set of literals modulo ordering. A clause in SNF_{ml} is a *tautology* if it simplifies to $ml : \mathbf{true}$. The clause $ml : D$, where $ml \in \{0, *\}$ and D is the empty disjunction (i.e. when $r = 0$ in the definition of literal clauses), is called an *empty clause* and denoted by $ml : \mathbf{false}$. Finally, as formulae in SNF_{ml} are required to be in simplified form, note that for a conjunction of clauses C, the formula $ml : \mathbf{true} \wedge C$ simplifies to $ml : C$.

Let φ be a formula in the language of K_n. In the following, we assume φ is in Negation Normal Form (NNF), that is, a formula where the operators are restricted to $\wedge$, $\vee$, $\boxed{a}$, $\diamondsuit_a$ and $\neg$; also, only propositional symbols are allowed in the scope of negations. The NNF of a formula can be obtained by exhaustively applying the rewriting rules given in Table 1. The simplification rules given in Table 2 are applied at any step of the transformation into NNF as well as during the transformation into the normal form.

$$\begin{array}{rcl@{\qquad}rcl}
\varphi \Rightarrow \varphi' & \longrightarrow & \neg\varphi \vee \varphi' & \neg\neg\varphi & \longrightarrow & \varphi \\
\neg(\varphi \wedge \varphi') & \longrightarrow & \neg\varphi \vee \neg\varphi' & \neg \boxed{a}\varphi & \longrightarrow & \langle a\rangle \neg\varphi \\
\neg(\varphi \vee \varphi') & \longrightarrow & \neg\varphi \wedge \neg\varphi' & \neg \langle a\rangle \varphi & \longrightarrow & \boxed{a}\neg\varphi
\end{array}$$

Table 1. Rewriting rules: NNF

$$\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl}
\varphi \wedge \varphi & \longrightarrow & \varphi & \varphi \wedge \neg\varphi & \longrightarrow & \mathbf{false} & \boxed{a}\mathbf{true} & \longrightarrow & \mathbf{true} \\
\varphi \vee \varphi & \longrightarrow & \varphi & \varphi \vee \neg\varphi & \longrightarrow & \mathbf{true} & \langle a\rangle\mathbf{false} & \longrightarrow & \mathbf{false} \\
\varphi \wedge \mathbf{true} & \longrightarrow & \varphi & \varphi \wedge \mathbf{false} & \longrightarrow & \mathbf{false} & \neg\mathbf{true} & \longrightarrow & \mathbf{false} \\
\varphi \vee \mathbf{false} & \longrightarrow & \varphi & \varphi \vee \mathbf{true} & \longrightarrow & \mathbf{true} & \neg\mathbf{false} & \longrightarrow & \mathbf{true}
\end{array}$$

Table 2. Rewriting Rules for Simplification

The transformation of the NNF of a formula φ into SNF_{ml} is achieved by recursively applying rewriting and renaming [51]. Let φ be a formula and t a propositional symbol not occurring in φ. For local satisfiability, the translation of φ is given by $(0 : t) \wedge \rho(0 : t \Rightarrow \varphi)$. We refer to clauses of the form $0 : D$, for a disjunction of literals D, as *initial clauses*. For global satisfiability, the translation of φ is given by $(* : t) \wedge \rho(* : t \Rightarrow \varphi)$ where t is a new propositional symbol. For the satisfiability of φ under the global constraints $\Gamma = \{\gamma_1, \dots, \gamma_m\}$, the translation is given by $(* : t) \wedge \rho(* : t \Rightarrow \bigwedge_{i=1}^{m} \gamma_i) \wedge \rho(0 : t \Rightarrow \varphi)$, where t is a new propositional symbol. The translation function $\rho : \mathsf{WFF}^{ml}_{\mathsf{K}_n} \longrightarrow \mathsf{WFF}^{ml}_{\mathsf{K}_n}$ is defined as follows (with $\varphi, \varphi' \in \mathsf{WFF}_{\mathsf{K}_n}$, t' is a new propositional symbol, and $* + 1 = *$):

$$\begin{array}{rcl}
\rho(ml : t \Rightarrow \varphi \wedge \varphi') & = & \rho(ml : t \Rightarrow \varphi) \wedge \rho(ml : t \Rightarrow \varphi') \\
\rho(ml : t \Rightarrow \boxed{a}\varphi) & = & (ml : t \Rightarrow \boxed{a}\varphi), \text{ if } \varphi \text{ is a literal} \\
 & = & (ml : t \Rightarrow \boxed{a}t') \wedge \rho(ml + 1 : t' \Rightarrow \varphi), \text{ otherwise} \\
\rho(ml : t \Rightarrow \langle a\rangle\varphi) & = & (ml : t \Rightarrow \langle a\rangle\varphi), \text{ if } \varphi \text{ is a literal} \\
 & = & (ml : t \Rightarrow \langle a\rangle t') \wedge \rho(ml + 1 : t' \Rightarrow \varphi), \text{ otherwise} \\
\rho(ml : t \Rightarrow \varphi \vee \varphi') & = & (ml : \neg t \vee \varphi \vee \varphi'), \\
 & & \text{if } \varphi \text{ and } \varphi' \text{ are disjunctions of literals or constants} \\
 & & \text{where } \varphi \text{ is possibly empty.} \\
 & = & \rho(ml : t \Rightarrow \varphi \vee t') \wedge \rho(ml : t' \Rightarrow \varphi'), \\
 & & \text{if } \varphi' \text{ is not a disjunction of literals or constants}
\end{array}$$

The definition of ρ introduces a new propositional symbol for each formula being renamed. This simplifies our presentation, but in order to reduce the number of variables and clauses introduced by the transformation function, propositional symbols can be reused, instead of introducing new ones, when the transformation is applied to formulae which have already been renamed. As the conjunction operator is commutative, associative, and idempotent, in the following we often refer to a formula in SNF_{ml} as a set of clauses. The next lemmas state that the transformation into SNF_{ml} is satisfiability preserving. The results are analogous to those for clause form transformation of first-order formulae using renaming (see, for instance, [7, 45] and references therein).

LEMMA 3.1. *Let $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$ be a formula, and $M = (W, w_0, R_1, \dots, R_n, \pi)$ and $M' = (W, w_0, R_1, \dots, R_n, \pi')$ be models. If, for all propositional symbols p occurring in φ and worlds $w \in W$, $\pi(w)(p) = \pi'(w)(p)$, then $\langle M, w\rangle \models \varphi$ if, and only if, $\langle M', w\rangle \models \varphi$.*

Proof. The proof is by induction on the structure of φ. The base case, where φ is a propositional symbol $p \in P$, is immediate, as $\pi(w)(p) = \pi'(w)(p)$. For the induction hypothesis, assume that for any proper subformula φ' of φ, we have that $\langle M, w\rangle \models \varphi'$ if, and only if, $\langle M', w\rangle \models \varphi'$, for all $w \in W$. The proof proceeds by case analysis (where $\varphi', \varphi'' \in \mathsf{WFF}_{\mathsf{K}_n}$; $a \in A_n$):

- Assume φ is of the form $\neg\varphi'$. By the definition of satisfiability, $\langle M, w\rangle \models \varphi$ if, and only if, $\langle M, w\rangle \not\models \varphi'$. By induction hypothesis, we have that $\langle M, w\rangle \not\models \varphi'$ if, and only if, $\langle M', w\rangle \not\models \varphi'$. By the definition of satisfiability, $\langle M', w\rangle \not\models \varphi'$ if, and only if, $\langle M', w\rangle \models \neg\varphi'$, that is, $\langle M', w\rangle \models \varphi$.
- Assume φ is of the form $\varphi' \wedge \varphi''$. By the definition of satisfiability, $\langle M, w\rangle \models \varphi$ if, and only if, $\langle M, w\rangle \models \varphi'$ and $\langle M, w\rangle \models \varphi''$. By induction hypothesis, we have that $\langle M, w\rangle \models \varphi'$ if, and only if, (1) $\langle M', w\rangle \models \varphi'$. Similarly, we obtain that $\langle M, w\rangle \models \varphi''$ if, and only if, (2) $\langle M', w\rangle \models \varphi''$. From (1) and (2), by the semantics of conjunction, we have that $\langle M', w\rangle \models \varphi' \wedge \varphi''$.
- Assume φ is of the form $\boxed{a}\varphi'$. By the definition of satisfiability, $\langle M, w\rangle \models \varphi$ if, and only if, for all worlds w' such that $(w, w') \in R_a$ we have that $\langle M, w'\rangle \models \varphi'$. By induction hypothesis, we have that $\langle M', w'\rangle \models \varphi'$, for all worlds w' such that $(w, w') \in R_a$. By the semantics of $\boxed{a}$, we obtain that $\langle M', w\rangle \models \boxed{a}\varphi'$.

The proof for the case where φ is of the form $\Diamond_a \varphi'$ is similar to the last case above. The proofs for constants, disjunctions, and implications follow easily from the above and well-known semantic equivalences, i.e. **true** $= p \vee \neg p$ and **false** $= p \wedge \neg p$, for some $p \in P$; $(\varphi' \vee \varphi'') = \neg(\neg\varphi' \wedge \neg\varphi'')$, and $(\varphi' \Rightarrow \varphi'') = (\neg\varphi' \vee \varphi'')$. From the definition of satisfiability of a formula and from the cases above, it follows that $\langle M, w\rangle \models \varphi$ if, and only if, $\langle M', w\rangle \models \varphi$. □

Corollary 3.2. *Let $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$ be a formula, and $M = (W, w_0, R_1, \ldots, R_n, \pi)$ and $M' = (W, w_0, R_1, \ldots, R_n, \pi')$ be models. If, for all propositional symbols p occurring in φ and worlds w in W, $\pi(w)(p) = \pi'(w)(p)$, then $M \models_L \varphi$ if, and only if, $M' \models_L \varphi$.*

Proof. By Lemma 3.1, by taking $w = w_0$. □

Corollary 3.3. *Let $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$ be a formula, and $M = (W, w_0, R_1, \ldots, R_n, \pi)$ and $M' = (W, w_0, R_1, \ldots, R_n, \pi')$ be models. If, for all propositional symbols p occurring in φ and worlds w in W, $\pi(w)(p) = \pi'(w)(p)$, then $M \models_G \varphi$ if, and only if, $M' \models_G \varphi$.*

Proof. By Lemma 3.1 applied to all worlds w in W. □

Corollary 3.4. *Let $\Gamma \subseteq \mathsf{WFF}_{\mathsf{K}_n}$ be a set of formulae, and $M = (W, w_0, R_1, \ldots, R_n, \pi)$ and $M' = (W, w_0, R_1, \ldots, R_n, \pi')$ be models. If, for all propositional symbols p occurring in φ and worlds w in W, $\pi(w)(p) = \pi'(w)(p)$, then $M \models_G \Gamma$ if, and only if, $M' \models_G \Gamma$.*

Proof. By Corollary 3.3 applied to all formulae in Γ. □

Corollary 3.5. *Let $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$ be a formula, $\Gamma \subseteq \mathsf{WFF}_{\mathsf{K}_n}$ be a set of formulae, and $M = (W, w_0, R_1, \ldots, R_n, \pi)$ and $M' = (W, w_0, R_1, \ldots, R_n, \pi')$ be models. If, for all propositional symbols p occurring in φ and worlds w in W, $\pi(w)(p) = \pi'(w)(p)$, φ is satisfied under the global constraints Γ in M if, and only if, φ is satisfied under the global constraints Γ in M'.*

Proof. By Corollary 3.2 applied to φ and Corollary 3.4 applied to Γ. □

Lemma 3.6. *Let $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$ be a formula and $t \in P$ be a propositional symbol not occurring in φ. Then, φ is locally satisfiable if, and only if, $(0 : t) \wedge (0 : t \Rightarrow \varphi)$ is satisfiable.*

Proof. If φ is locally satisfiable then there is a model $M = (W, w_0, R_1, \ldots, R_n, \pi)$ for K_n such that $\langle M, w_0\rangle \models \varphi$. Let M^* be the model obtained from M by only adding the universal relation R_* to M, that is, $M^* = (W, w_0, R_1, \ldots, R_n, R_*, \pi)$. As the evaluation of φ does not depend on the relation R_*, it is easy to show that $\langle M^*, w_0\rangle \models \varphi$. We now build a model $M'^* = (W, w_0, R_1, \ldots, R_n, R_*, \pi')$ where $\pi'(w)(p) = \pi(w)(p)$, for all worlds $w \in W$ and propositional symbols $p \neq t$, $\pi(w_0)(t) = \mathit{true}$, and $\pi(w)(t) = \mathit{false}$ for all worlds $w \neq w_0$. By construction, (1) $\langle M'^*, w_0\rangle \models t$. By Corollary 3.2, we also have that (2) $\langle M'^*, w_0\rangle \models \varphi$. By the semantics of implication and conjunction, it follows that $\langle M'^*, w_0\rangle \models t \wedge (t \Rightarrow \varphi)$. From Theorem 2.8, a formula is satisfied at the modal layer it occurs. Hence, as $\mathrm{depth}(w_0) = 0$, $M^* \models 0 : (t \wedge (t \Rightarrow \varphi))$. By the semantics of conjunction, we obtain that $M^* \models (0 : t) \wedge (0 : t \Rightarrow \varphi)$. □

Lemma 3.7. *Let $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$ be a formula and t be a propositional symbol not occurring in φ. Then φ is globally satisfiable if, and only if, $(* : t) \wedge (* : t \Rightarrow \varphi)$ is satisfiable.*

Proof. If φ is globally satisfiable then there is a model $M = (W, w_0, R_1, \ldots, R_n, \pi)$ for K_n such that for all $w \in W$ we have that $\langle M, w\rangle \models \varphi$. Let M^* be the model obtained from M by only adding the universal relation R_* to M, that is, $M^* = (W, w_0, R_1, \ldots, R_n, R_*, \pi)$. By [19, Proposition 2.1], $\langle M^*, w_0\rangle \models \boxed{*}\, \varphi$. We now build a model $M'^* = (W, w_0, R_1, \ldots, R_n, R_*, \pi')$ where, for all worlds $w \in W$, $\pi'(w)(p) = \pi(w)(p)$, for all $p \neq t$, and $\pi(w)(t) = \mathit{true}$. By construction, (1) $\langle M'^*, w_0\rangle \models \boxed{*}\, t$. By Corollary 3.3, we also have that (2) $\langle M'^*, w_0\rangle \models \boxed{*}\, \varphi$. By additivity, from (1) and (2), we obtain that $\langle M'^*, w_0\rangle \models \boxed{*}(t \wedge \varphi)$. By the semantics of conjunction and implication, we have that $\langle M'^*, w_0\rangle \models \boxed{*}(t \wedge (t \Rightarrow \varphi))$. By the semantics of labelled formulae, it follows that $\langle M'^*, w_0\rangle \models * : (t \wedge (t \Rightarrow \varphi))$. By the semantics of conjunction, we have that $\langle M'^*, w_0\rangle \models (* : t) \wedge (* : t \Rightarrow \varphi)$. □

Corollary 3.8. *Let $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$ be a formula, $\Gamma = \{\gamma_1, \ldots, \gamma_m\} \subseteq \mathsf{WFF}_{\mathsf{k}_n}$ a set of formulae, and t be a propositional symbol not occurring in φ nor in Γ. Then φ is satisfiable under the global constraints Γ if, and only if, $(* : t) \wedge (* : t \Rightarrow \bigwedge_{i=1}^{m} \gamma_i) \wedge (0 : t \Rightarrow \varphi)$ is satisfiable.*

Proof. From the definition of satisfiability of sets, if Γ is globally satisfiable, by Lemma 3.7, we have that $(* : t) \wedge (* : t \Rightarrow \bigwedge_{i=1}^{m} \gamma_i)$ is also globally satisfiable. If φ is locally satisfiable, then, by Lemma 3.6, we have that $(0 : t) \wedge (0 : t \Rightarrow \varphi)$ is also satisfiable. By the semantics of conjunction, we obtain that $(0 : t) \wedge (* : t) \wedge (* : t \Rightarrow \bigwedge_{i=1}^{m} \gamma_i) \wedge (0 : t \Rightarrow \varphi)$, which simplifies to $(* : t) \wedge (* : t \Rightarrow \bigwedge_{i=1}^{m} \gamma_i) \wedge (0 : t \Rightarrow \varphi)$, which is satisfiable. □

The next five lemmas show that every step of the transformation into the normal form is correct, that is, the application of the transformation function to a formula produces an equisatisfiable formula.

Lemma 3.9 ($\rho_\wedge$). *Let $\varphi, \varphi' \in \mathsf{WFF}_{\mathsf{K}_n}$ be formulae and $t \in P$ be a propositional symbol. Then $(ml : t \Rightarrow \varphi \wedge \varphi')$ is satisfiable if, and only if, $(ml : t \Rightarrow \varphi) \wedge (ml : t \Rightarrow \varphi')$ is satisfiable.*

Proof. ($\Rightarrow$) If $(ml : t \Rightarrow \varphi \wedge \varphi')$ is satisfiable, then there is a model $M^* = (W, w_0, R_1, \ldots, R_n, R_*, \pi)$ such that $M^* \models (ml : t \Rightarrow \varphi \wedge \varphi')$. Let $w \in W$ be any world such that $\mathrm{depth}(w) = ml$. By the definition of satisfiability of labelled formulae, $\langle M^*, w\rangle \models (t \Rightarrow \varphi \wedge \varphi')$. There are two cases.

- If $\langle M^*, w\rangle \not\models t$, then by the semantics of the implication we have that both $\langle M^*, w\rangle \models t \Rightarrow \varphi$ and $\langle M^*, w\rangle \models t \Rightarrow \varphi'$. By the semantics of the conjunction, we obtain that $\langle M^*, w\rangle \models (t \Rightarrow \varphi) \wedge (t \Rightarrow \varphi')$.
- If (1) $\langle M^*, w\rangle \models t$, then, by the semantics of the implication, we have that $\langle M^*, w\rangle \models (\varphi \wedge \varphi')$. By the semantics of the conjunction, we obtain that (2) $\langle M^*, w\rangle \models \varphi$ and (3) $\langle M^*, w\rangle \models \varphi'$.

By the semantics of the implication, from (1) and (2), we have that $\langle M^*, w\rangle \models t \Rightarrow \varphi$; similarly, from (1) and (3), we obtain that $\langle M^*, w\rangle \models t \Rightarrow \varphi'$. By the semantics of the conjunction, from (4) and (5), $\langle M^*, w\rangle \models (t \Rightarrow \varphi) \wedge (t \Rightarrow \varphi')$.

From the above, in both cases we have that $\langle M^*, w\rangle \models (t \Rightarrow \varphi) \wedge (t \Rightarrow \varphi')$. As w is arbitrarily chosen, we have that for all w, with $\mathrm{depth}(w) = ml$, $\langle M^*, w\rangle \models (t \Rightarrow \varphi) \wedge (t \Rightarrow \varphi')$. By the semantics of labelled formulae, we then have that $M^* \models ml : ((t \Rightarrow \varphi) \wedge (t \Rightarrow \varphi'))$. By the semantics of conjunction, it follows that $M^* \models (ml : t \Rightarrow \varphi) \wedge (ml : t \Rightarrow \varphi')$.

($\Leftarrow$) If $(ml : t \Rightarrow \varphi) \wedge (ml : t \Rightarrow \varphi')$ is satisfiable, then there is a model $M^* = (W, w_0, R_1, \ldots, R_n, R_*, \pi)$ such that $M^* \models (ml : t \Rightarrow \varphi) \wedge (ml : t \Rightarrow \varphi')$. By the semantics of conjunction, we have that $M^* \models ml : ((t \Rightarrow \varphi) \wedge (t \Rightarrow \varphi'))$. Let $w \in W$ be any world such that $\mathrm{depth}(w) = ml$. By the definition of satisfiability of labelled formulae, then $\langle M^*, w\rangle \models ((t \Rightarrow \varphi) \wedge (t \Rightarrow \varphi'))$. By the semantics of conjunction and implication, we have that $\langle M^*, w\rangle \models t \Rightarrow \varphi \wedge \varphi'$. By the semantics of labelled formulae, we then have that $M^* \models ml : t \Rightarrow \varphi \wedge \varphi'$. □

Lemma 3.10 ($\rho_\Box$). *Let $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$ be a formula and $t \in P$ be a propositional symbol. Then $(ml : t \Rightarrow \boxed{a}\varphi)$ is satisfiable if, and only if, $(ml : t \Rightarrow \boxed{a}t') \wedge (ml + 1 : t' \Rightarrow \varphi)$, where t' is a propositional symbol not occurring in φ, is satisfiable.*

Proof. ($\Rightarrow$) If $(ml : t \Rightarrow \boxed{a}\varphi)$ is satisfiable, then there is a model $M^* = (W, w_0, R_1, \ldots, R_n, R_*, \pi)$ such that $M^* \models ml : t \Rightarrow \boxed{a}\varphi$. Let $w \in W$ be any world such that $\mathrm{depth}(w) = ml$. By the definition of satisfiability of labelled formulae, then $\langle M^*, w\rangle \models t \Rightarrow \boxed{a}\varphi$. Construct $M'^* = (W, w_0, R_1, \ldots, R_n, R_*, \pi')$ such that $\pi'(w)(p) = \pi(w)(p)$ for all $p \neq t'$; and let $\pi'(w)(t') = true$ if, and only if, $\langle M^*, w\rangle \models \varphi$. As $M^* \models (ml : t \Rightarrow \boxed{a}\varphi)$, by Lemma 3.1, we have that $\langle M'^*, w\rangle \models (t \Rightarrow \boxed{a}\varphi)$. There are two cases:

- If $\langle M^*, w\rangle \not\models t$, then $\langle M'^*, w\rangle \not\models t$ (by Lemma 3.1). From the semantics of implication, it follows that $\langle M'^*, w\rangle \models t \Rightarrow \boxed{a}t'$.
- If $\langle M^*, w\rangle \models t$, then, by the semantics of implication and the semantics of the modal operator, for all w' such that $(w, w') \in R_a$, we have that $\langle M'^*, w'\rangle \models \varphi$. By construction, as t' holds at exactly the same worlds at which φ holds, for all w' such that $(w, w') \in R_a$, we have that $\langle M'^*, w'\rangle \models t'$, that is, $\langle M'^*, w\rangle \models \boxed{a}t'$. From the semantics of implication, we also have that $\langle M'^*, w\rangle \models t \Rightarrow \boxed{a}t'$. As $\mathrm{depth}(w) = ml$, we obtain (1) $M'^* \models_L (ml : t \Rightarrow \boxed{a}t')$. Also, by construction and by the semantics of implication, we have that $\langle M'^*, w'\rangle \models t' \Rightarrow \varphi$, for all such w'. As $\mathrm{depth}(w') = ml + 1$, by the semantics of labelled formulae, we have that (2) $M'^* \models (ml + 1 : t' \Rightarrow \varphi)$.

From (1) and (2), by the semantics of conjunction, $M'^* \models (ml : t \Rightarrow \boxed{a}t') \wedge (ml + 1 : t' \Rightarrow \varphi)$.

($\Leftarrow$) If $(ml : t \Rightarrow \boxed{a}t') \wedge (ml + 1 : t' \Rightarrow \varphi)$ is satisfiable, then there is a model $M^* = (W, w_0, R_1, \ldots, R_n, R_*, \pi)$ such that $M^* \models (ml : t \Rightarrow \boxed{a}t') \wedge (ml + 1 : t' \Rightarrow \varphi)$. By the semantics of conjunction, we have that $M^* \models (ml : t \Rightarrow \boxed{a}t')$ and $M^* \models (ml + 1 : t' \Rightarrow \varphi)$. Let $w \in W$ be any world such that $\mathrm{depth}(w) = ml$. By the definition of satisfiability of labelled formulae, then $\langle M^*, w\rangle \models t \Rightarrow \boxed{a}t'$. If $\langle M^*, w\rangle \not\models t$, then it easily follows that $\langle M^*, w\rangle \models t \Rightarrow \boxed{a}\varphi$. Assume $\langle M^*, w\rangle \models t$. Then, by the semantics of implication, we have that $\langle M^*, w\rangle \models \boxed{a}t'$. By the semantics of the modal operator, for all w' such that $(w, w') \in R_a$, then (1) $\langle M^*, w'\rangle \models t'$. As $M^* \models (ml + 1 : t' \Rightarrow \varphi)$, because $\mathrm{depth}(w') = ml + 1$, by the definition of satisfiability of labelled formulae, we also have that (2) $\langle M^*, w'\rangle \models t' \Rightarrow \varphi$. From (1) and (2), by chaining, we obtain that $\langle M^*, w'\rangle \models \varphi$. Hence, $\langle M^*, w\rangle \models \boxed{a}\varphi$. By the semantics of the implication, $\langle M^*, w\rangle \models t \Rightarrow \boxed{a}\varphi$, for all w such that $\mathrm{depth}(w) = ml$. By the semantics of labelled formulae, we have that $M^* \models (ml : t \Rightarrow \boxed{a}\varphi)$. □

The proofs of the next lemmas are similar to that of Lemmas 3.9 and 3.10 and are omitted here.

Lemma 3.11 ($\rho_{\diamond}$). *Let $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$ be a formula and $t \in P$ be a propositional symbol. Then $(ml : t \Rightarrow \diamondsuit\varphi)$ is satisfiable if, and only if, $(ml : t \Rightarrow \diamondsuit t') \land (ml + 1 : t' \Rightarrow \varphi)$, where t' is a propositional symbol not occurring in φ, is satisfiable.*

Lemma 3.12 ($\rho_{\lor D}$). *Let $\varphi, \varphi' \in \mathsf{WFF}_{\mathsf{K}_n}$ be formulae, where φ and φ' are disjunctions of literals or constants and $t \in P$ be a propositional symbol. Then $(ml : t \Rightarrow \varphi \lor \varphi')$ is satisfiable if, and only if, $(ml : \neg t \lor \varphi \lor \varphi')$ is satisfiable.*

Lemma 3.13 ($\rho_{\lor}$). *Let $\varphi, \varphi' \in \mathsf{WFF}_{\mathsf{K}_n}$ be formulae, where φ' is not a disjunction of literals or constants, and $t \in P$ be a propositional symbol. Then $(ml : t \Rightarrow \varphi \lor \varphi')$ is satisfiable if, and only if, $(ml : t \Rightarrow \varphi \lor t') \land \rho(ml : t' \Rightarrow \varphi')$, where t' is a propositional symbol not occurring in φ, is satisfiable.*

The next theorems show that the transformation into the normal form is correct for each of the reasoning tasks being considered here.

Theorem 3.14. *Let $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$ be a formula and $t \in P$ a propositional symbol not occurring in φ. Then φ is locally satisfiable if, and only if, $(0 : t) \land \rho(0 : t \Rightarrow \varphi)$ is satisfiable.*

Proof. By Lemma 3.6 and by Lemmas 3.9 to 3.13, which show that every step in the transformation is satisfiability preserving. □

Theorem 3.15. *Let $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$ be a formula and $t \in P$ a propositional symbol not occurring in φ. Then φ is globally satisfiable if, and only if, $(* : t) \land \rho(* : t \Rightarrow \varphi)$ is satisfiable.*

Proof. By Lemma 3.7 and by Lemmas 3.9 to 3.13, which show that every step in the transformation is satisfiability preserving. □

Theorem 3.16. *Let $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$ be a formula, $\Gamma \subseteq \mathsf{WFF}_{\mathsf{K}_n}$ be a set of formulae, and $t \in P$ a propositional symbol not occurring in φ nor in Γ. Then φ is locally satisfiable under the global constraints Γ if, and only if, $(* : t) \land \rho(* : t \Rightarrow \bigwedge_{\gamma \in \Gamma} \gamma) \land \rho(0 : t \Rightarrow \varphi)$ is satisfiable.*

Proof. By Corollary 3.8 and by Lemmas 3.9 to 3.13, which show that every step in the transformation is satisfiability preserving. □

In order to show that the translation procedure is terminating, note that recursive calls of ρ to formulae of the form $ml : t \Rightarrow \varphi$ apply to formulae whose number of operators is strictly smaller than those in φ. Thus, the application of ρ eventually stops.

4 INFERENCE RULES

The layered resolution-based calculus for K_n, named $\mathsf{RES}_{\mathsf{ml}}$, comprises a set of inference rules for dealing with propositional and modal reasoning. In the following, we denote by σ the result of unifying the labels in the premises for each rule. Formally, unification is given by a function $\sigma : \mathcal{P}(\mathbb{N} \cup \{*\}) \longrightarrow \mathbb{N} \cup \{*\}$, where $\sigma(\{ml, *\}) = ml$; and $\sigma(\{ml\}) = ml$; otherwise, σ is undefined. The inference rules given in Table 3 can only be applied if the unification of their labels is defined (where $* - 1 = *$). Note that for GEN1 and GEN3, if the modal clauses in the premises occur at the modal level ml, then the literal clause in the premises occurs at the next modal level, $ml + 1$. Also note that in both GEN1 and GEN3, the set of positive a-clauses in the premises can be empty, that is, $m = 0$. The conclusion of any of the inference rules is called a *resolvent*.

$$\begin{array}{rlcl}
\text{[LRES]} & & & \\
ml: & D & \vee & l \\
ml': & D' & \vee & \neg l \\
\hline
\sigma(\{ml, ml'\}): & D & \vee & D'
\end{array}
\qquad
\begin{array}{rlcl}
\text{[MRES]} & & & \\
ml: & l_1 & \Rightarrow & \boxed{a} l \\
ml': & l_2 & \Rightarrow & \langle a \rangle \neg l \\
\hline
\sigma(\{ml, ml'\}): & \neg l_1 & \vee & \neg l_2
\end{array}
\qquad
\begin{array}{rlcl}
\text{[GEN2]} & & & \\
ml_1: & l'_1 & \Rightarrow & \boxed{a} l_1 \\
ml_2: & l'_2 & \Rightarrow & \boxed{a} \neg l_1 \\
ml_3: & l'_3 & \Rightarrow & \langle a \rangle l_2 \\
\hline
\sigma(\{ml_1, ml_2, ml_3\}): & \multicolumn{3}{l}{\neg l'_1 \vee \neg l'_2 \vee \neg l'_3}
\end{array}$$

$$\begin{array}{rlcl}
\text{[GEN1]} & & & \\
ml_1: & l'_1 & \Rightarrow & \boxed{a} \neg l_1 \\
 & \vdots & & \\
ml_m: & l'_m & \Rightarrow & \boxed{a} \neg l_m \\
ml_{m+1}: & l' & \Rightarrow & \langle a \rangle \neg l \\
ml_{m+2}: & \multicolumn{3}{l}{l_1 \vee \ldots \vee l_m \vee l} \\
\hline
ml: & \multicolumn{3}{l}{\neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l'} \\
\multicolumn{4}{c}{\text{where } ml = \sigma(\{ml_1, \ldots, ml_{m+1}, ml_{m+2} - 1\})}
\end{array}
\qquad
\begin{array}{rlcl}
\text{[GEN3]} & & & \\
ml_1: & l'_1 & \Rightarrow & \boxed{a} \neg l_1 \\
 & \vdots & & \\
ml_m: & l'_m & \Rightarrow & \boxed{a} \neg l_m \\
ml_{m+1}: & l' & \Rightarrow & \langle a \rangle l \\
ml_{m+2}: & \multicolumn{3}{l}{l_1 \vee \ldots \vee l_m} \\
\hline
ml: & \multicolumn{3}{l}{\neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l'} \\
\multicolumn{4}{c}{\text{where } ml = \sigma(\{ml_1, \ldots, ml_{m+1}, ml_{m+2} - 1\})}
\end{array}$$

Table 3. Inference rules

Definition 4.1. Let Φ be a set of clauses in SNF_{ml}. A *derivation by* $\mathsf{RES}_{\mathsf{ml}}$ *from* Φ is a sequence of sets $\Phi_0, \Phi_1, \ldots$ where $\Phi_0 = \Phi$ and, for each $i > 0$, $\Phi_{i+1} = \Phi_i \cup \{D\}$, where D is the resolvent obtained from Φ_i by an application of either LRES, MRES, GEN1, GEN2, or GEN3 to premises in Φ_i. We also require that D is in simplified form, $D \notin \Phi_i$, and that D is not a tautology.

Note that all inference rules of $\mathsf{RES}_{\mathsf{ml}}$ generate literal clauses. Thus, in the previous definition we require that the resolvent does not simplify to $ml : \mathbf{true}$, for any modal level ml. A refutation is a finite derivation where the last derived clause represents a contradiction.

Definition 4.2. Let Φ be a set of clauses in SNF_{ml}. A *local refutation by* $\mathsf{RES}_{\mathsf{ml}}$ *from* Φ is a derivation $\Phi_0, \ldots, \Phi_k$ from Φ, $k \in \mathbb{N}$, where $0 : \mathbf{false} \in \Phi_k$. A *global refutation by* $\mathsf{RES}_{\mathsf{ml}}$ *from* Φ is a derivation $\Phi_0, \ldots, \Phi_k$ from Φ, $k \in \mathbb{N}$, where $* : \mathbf{false} \in \Phi_k$. A *local refutation under global constraints by* $\mathsf{RES}_{\mathsf{ml}}$ *from* Φ is a derivation $\Phi_0, \ldots, \Phi_k$ from Φ, $k \in \mathbb{N}$, where $0 : \mathbf{false} \in \Phi_k$ or $* : \mathbf{false} \in \Phi_k$.

The definition of a derivation already establishes two criteria for redundancy elimination, namely that the added resolvent is not repeated nor a tautology. Subsumption is another redundancy criterion which is used together with deletion for eliminating clauses that are already implied by another clause in the clause set. Although subsumption is not needed for termination, the procedure is generally used in the implementation of resolution-based provers, as it can be efficiently implemented and it often helps to reduce the number of clauses in the clause set.

Definition 4.3. Let C and D be disjunctions of literals. A clause $(ml_1 : C)$ *subsumes* a clause $(ml_2 : D)$, denoted by $(ml_1 : C) \leq_s (ml_2 : D)$, if, and only if, $\sigma(\{ml_1, ml_2\}) = ml_2$ and D is of the form $C \vee D'$, where D' is a (possibly empty) disjunction of literals.

Definition 4.4. Let Φ be a set of clauses in SNF_{ml}. We say that Φ is *saturated* if any further application of the inference rules LRES, MRES, GEN1, GEN2, or GEN3 generates a clause already in Φ or subsumed by a clause in Φ.

Definition 4.5. Let Φ be a set of clauses in SNF_{ml}. A derivation $\Phi_0, \Phi_1, \ldots$ from Φ *terminates* if either there is $k \in \mathbb{N}$ such that Φ_k is saturated or $\Phi_0, \Phi_1, \ldots, \Phi_k$ is a refutation from Φ.

Before presenting correctness results, we show an example of a local refutation from global assumptions.

Example 4.6. We assume a theory about the descendant relation, where this relation is transitive. The modality $\langle c \rangle$ is used for representing the relation of having a child. The modality $\langle d \rangle$ is used for representing the relation of having descendants. We assume that having grandchildren (i.e. children of children) hold so that having a descendant of a descendant also must hold. Globally, having a child implies having descendants. This is expressed by the following formula:

$$\langle c \rangle \mathbf{true} \Rightarrow \langle d \rangle \mathbf{true}$$

whose NNF is $\boxed{c}\mathbf{false} \vee \langle d \rangle \mathbf{true}$. The application of the transformation function to this formula introduces the new labelled clause $* : t_0$ together with the clauses resulting from $\rho(* : t_0 \Rightarrow \boxed{c}\mathbf{false} \vee \langle d \rangle \mathbf{true})$. That is, we obtain the following set of clauses:

$$\begin{array}{lll}
1. & * : & t_0 \\
2. & * : & \neg t_0 \vee t_1 \vee t_2 \\
3. & * : & t_1 \Rightarrow \boxed{c} t_f \\
4. & * : & t_2 \Rightarrow \langle d \rangle t_t \\
5. & * : & \neg t_f
\end{array}$$

where the propositional symbols t_f and t_t are introduced as the definitions of **false** and **true**, respectively. Note that the formula $* : t_f \Rightarrow \mathbf{false}$ simplifies to $* : \neg t_f$, as given in Clause 5. Also, the formula $* : t_t \Rightarrow \mathbf{true}$ simplifies to $* : \mathbf{true}$, which is a tautology. As formulae in SNF_{ml} are in simplified form, the tautology is not included in the clause set. In the following, we will reuse the propositional symbols t_t and t_f whenever the definitions of **true** and **false** are needed. The initial situation says that the grandchildren relation is not empty, that is, the local constraint is given by $\rho(0 : t_0 \Rightarrow \langle c \rangle \langle c \rangle \mathbf{true})$. In the normal form, we have the following two clauses:

$$\begin{array}{lll}
6. & 0 : & t_0 \Rightarrow \langle c \rangle t_3 \\
7. & 1 : & t_3 \Rightarrow \langle c \rangle t_t
\end{array}$$

We want to prove that the descendant relation has at least two generations, that is, we want to prove that the local constraint $\langle d \rangle \langle d \rangle \mathbf{true}$ holds. In order to obtain a contradiction, we apply the transformation function to the negation of this formula. From $\rho(0 : t_0 \Rightarrow \boxed{d}\,\boxed{d}\mathbf{false})$, the following two clauses are obtained:

$$\begin{array}{lll}
8. & 0 : & t_0 \Rightarrow \boxed{d} t_4 \\
9. & 1 : & t_4 \Rightarrow \boxed{d} t_f
\end{array}$$

Transitivity of a relation R_a is characterised by the Axiom 4, that is, $\boxed{a}\varphi \Rightarrow \boxed{a}\,\boxed{a}\varphi$, for a formula φ and a modal operator $\boxed{a}$ [14]. In order to properly characterise the fact that the descendant relation is transitive, we follow the approach in [39], adapted to the normal form given here: For each positive modal clause of the form $ml : l' \Rightarrow \boxed{a} l$ where R_a is a transitive relation, we add the global clauses $ml : \neg l' \vee nec_{a,l}$, $* : nec_{a,l} \Rightarrow \boxed{a} l$, $* : \neg nec_{a,l} \Rightarrow \langle a \rangle \neg nec_{a,l}$, and $* : nec_{a,l} \Rightarrow \boxed{a} nec_{a,l}$. That is, we introduce the propositional symbol $nec_{a,l}$ as the definition of $\boxed{a} l$. Hence, the clause $* : nec_{a,l} \Rightarrow \boxed{d} nec_{a,l}$ says that $* : \boxed{a} l \Rightarrow \boxed{a}\,\boxed{a} l$ holds. For this particular example, we add the following clauses to the clause set:

$$\begin{array}{llll}
10. & 1 : & \neg t_4 \vee nec_{d,t_f} & \text{[Axiom 4, 9]} \\
11. & * : & nec_{d,t_f} \Rightarrow \boxed{d} t_f & \text{[Axiom 4, 9]} \\
12. & * : & \neg nec_{d,t_f} \Rightarrow \langle d \rangle \neg nec_{d,t_f} & \text{[Axiom 4, 9]} \\
13. & * : & nec_{d,t_f} \Rightarrow \boxed{d} nec_{d,t_f} & \text{[Axiom 4, 9]}
\end{array}$$

We do not show the clauses which result from applying the same encoding to Clause 8, as they are not needed in the refutation, which proceeds as follows. The justification, shown on the right of each derived clause, says what rule was applied, the clauses in the premises, and the literals being resolved.

14.	$0:$	$\neg t_1 \vee \neg t_0$	$[\mathsf{GEN3}, 6, 3, 5, t_3, t_f]$
15.	$*:$	$\neg nec_{d,t_f} \vee \neg t_2$	$[\mathsf{GEN3}, 4, 11, 5, t_t, t_f]$
16.	$0:$	$nec_{d,t_f} \vee \neg t_0$	$[\mathsf{GEN1}, 12, 8, 10, nec_{d,t_f}, t_4]$
17.	$0:$	$\neg t_2 \vee \neg t_0$	$[\mathsf{LRES}, 15, 16, nec_{d,t_f}]$
18.	$0:$	$t_1 \vee \neg t_0$	$[\mathsf{LRES}, 2, 17, t_2]$
19.	$0:$	$\neg t_0$	$[\mathsf{LRES}, 18, 14, t_1]$
20.	$0:$	**false**	$[\mathsf{LRES}, 19, 1, t_0]$

5 CORRECTNESS RESULTS

In this section, we provide proofs for termination, soundness, and completeness of the layered resolution-based calculus for K_n, $\mathsf{RES}_{\mathsf{ml}}$.

THEOREM 5.1 (TERMINATION). *Let Φ be a set of clauses in SNF_{ml}. Then, any derivation by $\mathsf{RES}_{\mathsf{ml}}$ from Φ terminates.*

PROOF. Here we regard a clause as a set of literals or modal literals. Let P_Φ be the set of propositional symbols occurring in Φ. We define $\overline{P_\Phi} = \{\neg p \mid p \in P_\Phi\}$, $L_\Phi = P_\Phi \cup \overline{P_\Phi}$, and $L_\Phi^{A_n} = \{\boxed{a} l, \Diamond_a l \mid l \in L_\Phi \text{ and } a \in A_n\}$. As P_Φ and A_n are both finite, we have that $\mathcal{P}(L_\Phi \cup L_\Phi^{A_n})$ is finite. Let C_Φ be the largest set of clauses that can be constructed from P_Φ and A_n. From Definition 4.1, for all derivations $\Phi_0, \Phi_1, \ldots$ from Φ, we have that $\Phi_i \subset C_\Phi$ and also that $\Phi_i \subset \Phi_{i+1}$, for all $i > 0$. Thus, every derivation must terminate. □

Next, we show that the inference rules are sound.

LEMMA 5.2 (σ). *Let $(ml : C)$ and $(ml' : C')$ be clauses in SNF_{ml} and M be a model such that $M \models (ml : C) \wedge (ml' : C')$. If $\sigma(\{ml, ml'\}) = ml''$ is defined, then $M \models (ml'' : C) \wedge (ml'' : C')$.*

PROOF. By definition, the unification function is defined in only two cases:

- The function is applied to $\{ml, ml'\} = \{ml''\}$, that is, the labels are identical, then the lemma holds trivially.
- The function is applied to $\{ml, ml'\}$, where $ml \neq *$ and $ml' = *$. By definition, $ml'' = ml$ and, clearly, we have that (1) $M \models (ml'' : C)$. As $M \models (ml' : C')$ and $ml' = *$, then we have that for all worlds w, $\langle M, w\rangle \models C'$. In particular, for all worlds w' with $\mathrm{depth}(w') = ml''$, we have that $\langle M, w'\rangle \models C'$. Hence, by the definition of satisfiability of labelled clauses, (2) $M \models ml'' : C'$. From (1) and (2), by the semantics of conjunction, we have that $M \models (ml'' : C) \wedge (ml'' : C')$.

It follows that if $\sigma(\{ml, ml'\}) = ml''$ is defined, then $M \models (ml'' : C) \wedge (ml'' : C')$. □

LEMMA 5.3 (LRES). *Let Φ be a set of clauses in SNF_{ml} with $\{ml : D \vee l, ml' : D' \vee \neg l\} \subseteq \Phi$, where D and D' are disjunctions of literals. If Φ is satisfiable and $\sigma(\{ml, ml'\})$ is defined, then $\Phi \cup \{\sigma(\{ml, ml'\}) : D \vee D'\}$ is satisfiable.*

PROOF. Let $M^* = (W, w_0, R_1, \ldots, R_n, R_*, \pi)$ be a model such that $M^* \models \Phi$. As $ml : D \vee l, ml' : D' \vee \neg l \in \Phi$, then $M^* \models ml : D \vee l$ and $M^* \models ml' : D' \vee \neg l$. Let $\sigma(\{ml, ml'\}) = ml''$. By Lemma 5.2, we have that $M^* \models ml'' : D \vee l$ and $M^* \models ml'' : D' \vee \neg l$. As $M^* \models ml'' : D' \vee \neg l$, for all $w \in W$ with $\mathrm{depth}(w) = ml''$, then $\langle M^*, w\rangle \models D' \vee \neg l$. Similarly, from $M^* \models ml'' : D \vee l$, for all $w \in W$

with $\mathrm{depth}(w) = ml''$, we obtain that $\langle M^*, w\rangle \models D \vee l$. It follows that $\langle M^*, w\rangle \models (D \vee l) \wedge (D' \vee \neg l)$, for all $w \in W$ with $\mathrm{depth}(w) = ml''$. By soundness of resolution, we have that $\langle M^*, w\rangle \models D \vee D'$. As $\mathrm{depth}(w) = ml'' = \sigma(\{ml, ml'\})$, we conclude that $M^* \models \sigma(\{ml, ml'\}) : D \vee D'$. □

Lemma 5.4 (MRES). *Let Φ be a set of clauses in SNF_{ml} with $\{ml : l_1 \Rightarrow \boxed{a} l, ml' : l_2 \Rightarrow \Diamond_a \neg l\} \subseteq \Phi$. If Φ is satisfiable and $\sigma(\{ml, ml'\})$ is defined, then $\Phi \cup \{\sigma(\{ml, ml'\}) : \neg l_1 \vee \neg l_2\}$ is satisfiable.*

Proof. The proof is similar to that of Lemma 5.3, as implications can be rewritten as disjunctions and $\Diamond_a \neg l$ is semantically equivalent to $\neg \boxed{a} l$. □

Lemma 5.5 (GEN1). *Let Φ be a set of clauses in SNF_{ml} with $\{ml_1 : l'_1 \Rightarrow \boxed{a}\neg l_1, \ldots, ml_m : l'_m \Rightarrow \boxed{a}\neg l_m, ml_{m+1} : l' \Rightarrow \Diamond_a \neg l, ml_{m+2} : l_1 \vee \ldots \vee l_m \vee l\} \subseteq \Phi$. If Φ is satisfiable and $\sigma(\{ml_1, \ldots, ml_m, ml_{m+1}, ml_{m+2} - 1\})$ is defined, then $\Phi \cup \{\sigma(\{ml_1, \ldots, ml_m, ml_{m+1}, ml_{m+2} - 1\}) : \neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l'\}$ is satisfiable.*

Proof. Let $M^* = (W, w_0, R_1, \ldots, R_n, R_*, \pi)$ be a model such that $M^* \models \Phi$. Let $\sigma(\{ml_1, \ldots, ml_m, ml_{m+1}, ml_{m+2} - 1\}) = ml$, for a particular modal level ml. By Lemma 5.2, if Φ is satisfiable, then $M^* \models (ml : (l'_1 \Rightarrow \boxed{a}\neg l_1) \wedge \ldots \wedge (l'_m \Rightarrow \boxed{a}\neg l_m) \wedge (l' \Rightarrow \Diamond_a \neg l)) \wedge (ml + 1 : (l_1 \vee \ldots \vee l_m \vee l))$ and so for all worlds $w \in W$, with $\mathrm{depth}(w) = ml$, we have that **(1)** $\langle M^*, w\rangle \models (l'_1 \Rightarrow \boxed{a}\neg l_1) \wedge \ldots \wedge (l'_m \Rightarrow \boxed{a}\neg l_m) \wedge (l' \Rightarrow \Diamond_a \neg l)$. If $\langle M^*, w\rangle \not\models l'$, it follows easily that $\langle M^*, w\rangle \models \neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l'$ and, therefore, $M^* \models \sigma(\{ml_1, \ldots, ml_m, ml_{m+1}, ml_{m+2} - 1\}) : \neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l'$. The same occurs if any of the literals l'_i, $1 \leq i \leq m$, is not satisfied at w. We show, by contradiction, that this must be the case. Suppose $\langle M^*, w\rangle \models l'_1 \wedge \ldots \wedge l'_m \wedge l'$. From this and from (1), by the semantics of implication, the semantics of the modal operator $\Diamond_a$, and the semantics of the modal operator $\boxed{a}$, we have that there is a world w', with $\mathrm{depth}(w') = \mathrm{depth}(w) + 1$, where $\neg l_1 \wedge \ldots \wedge \neg l_m \wedge \neg l$ holds. Now, as $ml_{m+2} - 1$ is unifiable with $\{ml_1, \ldots, ml_m, ml_{m+1}\}$, for all worlds w'' with $\mathrm{depth}(w'') = \mathrm{depth}(w) + 1$, we obtain that $\langle M^*, w''\rangle \models l_1 \vee \ldots \vee l_m \vee l$. In particular, because $\mathrm{depth}(w') = \mathrm{depth}(w'')$, we obtain that $\langle M^*, w'\rangle \models (\neg l_1 \wedge \ldots \wedge \neg l_m \wedge \neg l) \wedge (l_1 \vee \ldots \vee l_m \vee l)$. By several applications of the classical propositional resolution rule, we have that $\langle M^*, w'\rangle \models$ **false**. This contradicts the fact that $ml : (l' \Rightarrow \Diamond_a \neg l)$ is satisfiable. Thus, $\langle M^*, w\rangle \models \neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l'$. As $\mathrm{depth}(w) = ml$, we conclude that $M^* \models \sigma(\{ml_1, \ldots, ml_m, ml_{m+1}, ml_{m+2} - 1\}) : \neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l'$. □

Lemma 5.6 (GEN2). *Let Φ be a set of clauses in SNF_{ml} with $\{ml_1 : l'_1 \Rightarrow \boxed{a} l_1, ml_2 : l'_2 \Rightarrow \boxed{a}\neg l_1, ml_3 : l'_3 \Rightarrow \Diamond_a l_2\} \subseteq \Phi$. If Φ is satisfiable and $\sigma(\{ml_1, ml_2, ml_3\})$ is defined, then $\Phi \cup \{\sigma(\{ml_1, ml_2, ml_3\}) : \neg l'_1 \vee \neg l'_2 \vee \neg l'_3\}$ is satisfiable.*

Proof. From Lemma 5.5 by taking Φ such that $\{ml_1 : l'_1 \Rightarrow \boxed{a} l_1, ml_2 : l'_2 \Rightarrow \boxed{a}\neg l_1, ml_3 : l'_3 \Rightarrow \Diamond_a l_2, * : l_1 \vee \neg l_1 \vee \neg l_2\} \subseteq \Phi$, as $l_1 \vee \neg l_1 \vee \neg l_2$ is a tautology. □

Lemma 5.7 (GEN3). *Let Φ be a set of clauses in SNF_{ml} with $\{ml_1 : l'_1 \Rightarrow \boxed{a}\neg l_1, \ldots, ml_m : l'_m \Rightarrow \boxed{a}\neg l_m, ml_{m+1} : l' \Rightarrow \Diamond_a l, ml_{m+2} : l_1 \vee \ldots \vee l_m\} \subseteq \Phi$. If Φ is satisfiable and $\sigma(\{ml_1, \ldots, ml_m, ml_{m+1}, ml_{m+2} - 1\})$ is defined, then $\Phi \cup \{\sigma(\{ml_1, \ldots, ml_m, ml_{m+1}, ml_{m+2} - 1\}) : \neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l'\}$ is satisfiable.*

Proof. The formula $ml_{m+2} : l_1 \vee \ldots \vee l_m$ is semantically equivalent to $(ml_{m+2} : l_1 \vee \ldots \vee l_m \vee l) \wedge (ml_{m+2} : l_1 \vee \ldots \vee l_m \vee \neg l)$. The proof follows from Lemma 5.5 by taking Φ such that $\{ml_1 : l'_1 \Rightarrow \boxed{a}\neg l_1, \ldots, ml_m : l'_m \Rightarrow \boxed{a}\neg l_m, ml_{m+1} : l' \Rightarrow \Diamond_a l, ml_{m+2} : l_1 \vee \ldots \vee l_m \vee \neg l\} \subseteq \Phi$. □

The next theorem shows that $\mathsf{RES}_{\mathsf{ml}}$ is sound.

Theorem 5.8 (Soundness). *Let Φ be a set of clauses in SNF_{ml} and $\Phi_0, \ldots, \Phi_k$, $k \in \mathbb{N}$, be a derivation by $\mathsf{RES}_{\mathsf{ml}}$ from Φ. If Φ is satisfiable, then every Φ_i, $0 \leq i \leq k$, is satisfiable.*

Proof. From Lemmas 5.3 to 5.7, by induction on the number of sets in a derivation. □

Completeness is proved by showing that if a set Φ of clauses in SNF_{ml} is unsatisfiable, then there is a refutation produced by the method presented here. The proof is by induction on the number of nodes of a graph, known as *behaviour graph*, built from Φ. Intuitively, nodes in the graph correspond to worlds and the set of edges correspond to the agents' accessibility relations in a model. The graph construction is similar to the construction of a canonical model, followed by filtrations based on the set of clauses, often used to prove completeness for proof methods in modal logics [14]. The main difference from the usual construction is that we take advantage of the fact that clauses are in a simple normal form. Thus, nodes contain sets of literals, instead of the usual set of formulae. As we are dealing with both global and local satisfiability, we first construct a graph $\mathcal{G}_G$ that satisfies the clauses labelled by $*$ and then complete the construction by unfolding $\mathcal{G}_G$ into a graph $\mathcal{G}$ which satisfies all clauses in Φ. We prove that an unsatisfiable set of clauses has an empty behaviour graph. We then show that deletions of either edges or nodes from the graph correspond to applications of the inference rules of $\mathsf{RES}_{\mathsf{ml}}$. It follows that if the behaviour graph is empty, then there is a refutation by $\mathsf{RES}_{\mathsf{ml}}$ for Φ using the inference rules given in Section 4.

Formally, let P_Φ and $\{0, \dots, m\} \cup \{*\}$ be the set of propositional symbols and the set of labels occurring in Φ, respectively. We define L_Φ as the set of literals given by $\{p, \neg p \mid \text{ for all } p \in P_\Phi\}$. A set of literals η is *maximally consistent* if for all $l \in L_\Phi$ either l or $\neg l$ is in η (but not both).

The behaviour graph $\mathcal{G}$ for n agents is a tuple $\mathcal{G} = \langle N_0, \dots, N_{m+1}, E_1, \dots, E_n \rangle$, built from the set of SNF_{ml} clauses Φ, where N_i is a set of nodes for each modal level $0 \leq i \leq m+1$ occurring in Φ and each E_a is a set of edges labelled by $a \in A_n$. Every element of N_i is a maximally consistent set of literals occurring in Φ. Note that we could have defined a node in N_i as containing literals occurring in any clauses labelled by $*$, literals occurring in literal clauses labelled by i, literals occurring on the left-hand side of modal clauses labelled i, and literals occurring in the scope of modal operators of modal clauses labelled by $i-1$. Using all literals in Φ simplifies the construction.

Let η be a node in $\mathcal{G}$. Let φ, ψ, and ψ' be Boolean combinations of literals and modal literals. We say that η satisfies φ in $\mathcal{G}$ (written $\mathcal{G}, \eta \models \varphi$, or $\eta \models \varphi$ if $\mathcal{G}$ is clear from the context), if, and only if:

- φ is a propositional literal and $\varphi \in \eta$;
- φ is of the form $\neg\psi$ and η does not satisfy ψ (written $\eta \not\models \psi$);
- φ is of the form $\psi \wedge \psi'$ and $\eta \models \psi$ and $\eta \models \psi'$;
- φ is of the form $\boxed{a} l$, where $l \in L_\Phi$ and $a \in A_n$, and for every η', if $(\eta, \eta') \in E_a$ then η' satisfies l;
- φ is of the form $\Diamond_a l$, where $l \in L_\Phi$ and $a \in A_n$, and there exists η' such that $(\eta, \eta') \in E_a$ and η' satisfies l.

The construction of the behaviour graph starts by partitioning a set of clauses Φ into two components corresponding to the set of global clauses and the set of local clauses. Let Φ_G be $\{* : \varphi \mid * : \varphi \in \Phi\}$ and $\Phi_L = \Phi \setminus \Phi_G$. First we construct a graph $\mathcal{G}_G = \langle N, E'_1, \dots, E'_n \rangle$, where N is the set of all maximally consistent sets of literals occurring in Φ. Delete from N any nodes that do not satisfy D such that the literal clause $* : D$ is in Φ_G. This ensures that all literal clauses in Φ_G are satisfied at all nodes. If the set of nodes is empty, then the graph is empty and the literal clauses in Φ_G are unsatisfiable. Otherwise, the construction proceeds as follows.

We now construct the sets of edges related to each agent and ensure that any global modal clause in Φ is also satisfied. We first set E'_a as $N \times N$, for all $a \in A_n$, and then remove elements of E'_a that do not satisfy the a-clauses. For all clauses $* : l' \Rightarrow \boxed{a} l$ occurring in Φ_G, delete from E'_a the edges (η, η') where $\eta \models l'$ but $\eta' \not\models l$. This ensures that all clauses of the form $* : l' \Rightarrow \boxed{a} l$ are now satisfied in $\mathcal{G}_G$. For all clauses $* : l' \Rightarrow \Diamond_a l$, repeatedly delete from $\mathcal{G}_G$ any nodes η such that

$\eta \models l'$ and there is no η' such that $(\eta, \eta') \in E'_a$ and $\eta' \models l$. This ensures that all clauses of the form $ml : l' \Rightarrow \langle a \rangle l$ are now satisfied in $\mathcal{G}_G$. If the set of clauses Φ_L is empty, the construction finishes and the behaviour graph $\mathcal{G}$ is given by $\mathcal{G}_G$. Else, if Φ_L and the $\mathcal{G}_G$ are both not empty, in order to satisfy the local constraints, given by clauses in Φ_L, we construct the graph $\mathcal{G}$ for Φ as follows.

Let $\mathcal{G}_G = \langle N, E'_1, ..., E'_n \rangle$ be the non-empty graph for Φ_G constructed as above. Recall that $\{0, \dots, m\}$ is the set of modal levels occurring as labels in Φ. The graph $\mathcal{G} = \langle N_0, ..., N_{m+1}, E_1, ..., E_n \rangle$ for Φ is constructed by the unfolding of $\mathcal{G}_G$ as follows. Note that we need to construct the nodes at the level $m+1$ in order to satisfy the literals in the scope of modal operators at the level m. First, we construct the set of nodes N_{ml} for each modal level ml, $0 \leq ml \leq m+1$. Define $N_0 = N$, $N_{ml} = \emptyset$, for $0 < ml \leq m+1$, and $E_a = \emptyset$, for $0 \leq a \leq n$. For each $\eta, \eta' \in N$, for all $ml \in \{0, \dots, m+1\}$, if $\eta \in N_{ml}$ and $(\eta, \eta') \in E'_a$ for any $a \in A_n$, then add a copy of η', named η'_{ml+1}, to N_{ml+1} and make $E_a = E_a \cup \{(\eta, \eta'_{ml+1})\}$. For the highest modal level, $ml = m+1$, we also need to make sure that the global constraints are still satisfied: if $\eta \in N_{m+1}$ then we also add copies of all nodes reachable from η, by any relation E'_a, and add the corresponding relations to each E_a. That is, for all $\eta \in N_{m+1}$, for all $a \in A_n$, if $(\eta, \eta') \in E'_a$, then let η'_{ml+1} be a copy of η' and make $N_{ml+1} = N_{ml+1} \cup \{\eta'_{ml+1}\}$ and $E_a = E_a \cup \{(\eta, \eta'_{ml+1})\}$.

Once the construction has finished, we delete nodes and edges in the following order to ensure that clauses in Φ are satisfied. First, delete from N_{ml} any nodes that do not satisfy D for all literal clauses of the form $ml : D$ in Φ_L. Then, for all clauses $ml : l' \Rightarrow \boxed{a} l$ in Φ_L, delete from E_a the edges $(\eta_{ml}, \eta'_{ml+1})$ where $\eta_{ml} \models l'$ but $\eta' \not\models l$, which ensures that the positive a-clauses at the modal level ml are satisfied at η. Next, consider any nodes that do not satisfy the negative a-clauses in Φ_L or in Φ_G. For all clauses $ml : l' \Rightarrow \langle a \rangle l$ in Φ_L (resp. for all clauses $* : l''' \Rightarrow \langle a \rangle l''$ in Φ_G), repeatedly delete from N_{ml} any nodes η_{ml} that satisfy l' (resp. l'''), but there is no node η'_{ml+1} such that $(\eta_{ml}, \eta'_{ml+1}) \in E_a$ and $\eta'_{ml+1} \models l$ (resp. $\eta'_{ml+1} \models l''$). This ensures that the negative a-clauses are satisfied at every node η_{ml} in N_{ml}.

Let $\mathcal{G} = \langle N_0, \dots, N_{m+1}, E_1, \dots, E_n \rangle$ be the behaviour graph for Φ. A node $\eta \in N_0$ is called an *initial node*. We define the *reduced behaviour graph* $\mathcal{G}'$ for Φ as the reachable component subgraphs of $\mathcal{G}$ which contain an initial node. It follows that if N_0 is empty, then the reduced behaviour graph is empty. We show that a set of SNF_{ml} clauses is satisfiable if, and only if, the reduced behaviour graph for this set of clauses is non-empty. In the only if part of the proof, we rely on the fact that if a set of clauses is satisfiable, then it is satisfiable in a finite structure obtained by filtration. The following definitions are needed.

Definition 5.9. Let Φ be a set of clauses. The set of labelled subformulae of Φ, denoted by $\mathsf{subf}(\Phi)$, is the union of the set of subformulae for each clause C occurring in Φ, where the set of subformulae of a clause C is given as follows:

- if C is of the form $ml : p$, then $\mathsf{subf}(C) = \{ml : p\}$, for $p \in P$;
- if C is of the form $ml : \neg p$, then $\mathsf{subf}(C) = \{ml : \neg p, ml : p\}$, for $p \in P$;
- if C is of the form $ml : l_1 \vee \dots \vee l_k$, then $\mathsf{subf}(C) = \{ml : l_1 \vee \dots \vee l_k\} \bigcup_{i=1}^{k} \mathsf{subf}(ml : l_i)$, for literals l_i, $1 \leq i \leq k$, $k \in \mathbb{N}$;
- if C is of the form $ml : l \Rightarrow \boxed{a} l'$, then $\mathsf{subf}(C) = \{ml : l \Rightarrow \boxed{a} l'\} \cup \mathsf{subf}(ml : \neg l) \cup \mathsf{subf}(ml + 1 : l')$, for literals l, l';
- if C is of the form $ml : l \Rightarrow \langle a \rangle l'$, then $\mathsf{subf}(C) = \{ml : l \Rightarrow \langle a \rangle l'\} \cup \mathsf{subf}(ml : \neg l) \cup \mathsf{subf}(ml + 1 : l')$, for literals l, l'.

Definition 5.10. Let Φ be a set of clauses and $\mathsf{subf}(\Phi)$ be the set of all labelled subformulae of Φ. Let $M = (W, w_0, R_1, \dots, R_n, R_*, \pi)$ be a model. Then $M^\Phi = (W^\Phi, w_0^\Phi, R_1^\Phi, \dots, R_n^\Phi, \pi^\Phi)$ is the *filtration* of M with respect to Φ, where for every $w \in W$, $w^\Phi = \{ml : \varphi \mid ml : \varphi \in \mathsf{subf}(\Phi), \mathsf{depth}(w) =$

ml or $ml = *$, and $\langle M, w\rangle \models ml : \varphi\}$, $W^{\Phi} = \{w^{\Phi} \mid w \in W\}$, $R^{\Phi}_a = \{(w^{\Phi}, w'^{\Phi}) \mid (w, w') \in R_a\}$, for all $w, w' \in W$ and $a \in A_n \cup \{*\}$; and $\pi^{\Phi}(w^{\Phi})(p) = \pi(w)(p)$, for all propositional symbols p such that $ml : p \in \text{subf}(\Phi)$ and worlds $w \in W$.

It is easy to show that the following two properties hold for this construction of M^{Φ}: (i) for all propositional symbols p, $\pi^{\Phi}(w^{\Phi})(p) = \mathit{true}$ if, and only if, $ml : p \in w^{\Phi}$; and (ii) for all worlds w^{Φ} in W^{Φ}, for all $ml : \Diamond_a l \in \text{subf}(\Phi)$, where l is a literal, if $ml : \Diamond_a l \in w^{\Phi}$, then there is a world $w'^{\Phi} \in W^{\Phi}$ with $(w^{\Phi}, w'^{\Phi}) \in R^{\Phi}_a$ such that $ml + 1 : l \in w'^{\Phi}$. It is also easy to show that no propositional inconsistencies can arise from the construction of M^{Φ}, that is, the following properties hold: (iii) $ml : p \in w^{\Phi}$ if, and only if, $ml : \neg p \notin w^{\Phi}$; (iv) $(ml : l_1 \vee \ldots \vee l_k) \in w^{\Phi}$, for literals l_j, $1 \leq j \leq k$, $k \in \mathbb{N}$, $k > 0$, if, and only if, there is $ml : l_j$ in w^{Φ}, with $1 \leq j \leq k$; (iv) $(ml : l' \Rightarrow \boxed{a} l) \in w^{\Phi}$ (resp. $(ml : l' \Rightarrow \Diamond_a l) \in w^{\Phi}$) if, and only if, $ml : l' \notin w^{\Phi}$ or $ml : \boxed{a} l \in w^{\Phi}$ (resp. $ml : \Diamond_a l \in w^{\Phi}$). A truth lemma shows that for a formula $ml : \varphi \in \text{subf}(\Phi)$ we have that $\langle M, w\rangle \models ml : \varphi$ if, and only if, $\langle M^{\Phi}, w^{\Phi}\rangle \models \varphi$. The proof is by induction on the structure of $ml : \varphi$ and it follows the standard proofs on filtrations (see [34], for filtrations for modal logics under global constraints).

Lemma 5.11. *Let Φ be a set of clauses. Φ is satisfiable if, and only if, the reduced behaviour graph $\mathcal{G}$ constructed from Φ is non-empty.*

Proof. ($\Rightarrow$) Assume Φ is satisfiable and let $M = (W, w_0, R_1, \ldots, R_n, R_*, \pi)$ be the filtration of a model that satisfies Φ. Let $\mathcal{G} = \langle N_0, \ldots, N_{m+1}, E_1, \ldots, E_n\rangle$ be the reduced graph for Φ. We show that for every world w in W, there is a node η in $\mathcal{G}$ such that, for every subformula φ in $\text{subf}(\Phi)$, if $\langle M, w\rangle \models \varphi$, then $\eta \models \varphi$. Let N denote the union of N_0, …, N_{m+1} in $\mathcal{G}$. We define a function $f : W \longrightarrow \mathcal{P}(N)$ such that $f(w) = \{\eta \mid \eta \in N_{\text{depth}(w)}$ and for all propositional symbols p occurring in Φ, $p \in \eta$ if, and only if, $\pi(w)(p) = \mathit{true}\}$. In other words, the function f associates to every world in M occurring at a depth ml a set of nodes in the graph at the corresponding modal level which agree on the satisfiability of the propositional symbols p occurring in Φ . By the maximal consistency property of nodes, it follows that if $\eta \in f(w)$ and $\pi(w)(p) = \mathit{false}$, then $\neg p \in \eta$. Assume that $\eta \in f(w)$, that is, η is a node in N_{ml} such that η agrees on the satisfiability of the propositional symbols occurring in w.

- For any literal clause $ml : l_1 \vee \ldots \vee l_k$, if $\langle M, w\rangle \models ml : l_1 \vee \ldots \vee l_k$, then there exists a literal l_j, $1 \leq j \leq k$, such that $\langle M, w\rangle \models ml : l_j$. If $\eta \in f(w)$, then, by definition of f, $l_j \in \eta$. Thus $\eta \models l_1 \vee \ldots \vee l_k$.
- For any positive a-clause $ml : l' \Rightarrow \boxed{a} l$, if $\langle M, w\rangle \models ml : l' \Rightarrow \boxed{a} l$, then (i) $\langle M, w\rangle \models ml : \neg l'$ or (ii) $\langle M, w\rangle \models ml : l' \wedge \boxed{a} l$. Let $\eta \in f(w)$. For (i), by the definition of f, we have that (iii) $\eta \models \neg l'$; thus, $\eta \models l' \Rightarrow \boxed{a} l$. For (ii), if $\langle M, w\rangle \models ml : l' \wedge \boxed{a} l$, then, by the definition of f, we have that $\eta \models l'$. By construction of $\mathcal{G}$, there is no node $\eta' \in N_{ml+1}$ such that $(\eta, \eta') \in E_a$ and $\eta' \models \neg l$. It follows that (iv) $\eta \models \boxed{a} l$. From (iii) and (iv), we have that $\eta \models l' \Rightarrow \boxed{a} l$.
- For any negative a-clause $ml : l' \Rightarrow \Diamond_a l$, the proof is similar: if $\langle M, w\rangle \models ml : l' \Rightarrow \Diamond_a l$, then (i) $\langle M, w\rangle \models ml : \neg l'$ or (ii) $\langle M, w\rangle \models ml : l' \wedge \Diamond_a l$. Let $\eta \in f(w)$. For (i), by the definition of f, we have that (iii) $\eta \models \neg l'$; thus, $\eta \models l' \Rightarrow \Diamond_a l$. For (ii), if $\langle M, w\rangle \models ml : l' \wedge \Diamond_a l$, then, by the definition of f, we have that $\eta \models l'$. By construction of $\mathcal{G}$, there is a node $\eta' \in N_{ml+1}$ such that $(\eta, \eta') \in E_a$ and $\eta' \models \neg l$. It follows that (iv) $\eta \models \Diamond_a l$. From (iii) and (iv), we have that $\eta \models l' \Rightarrow \Diamond_a l$.

From the above, for any world in a filtrated model M there is a node in the behaviour graph $\mathcal{G}$ that satisfies the same set of clauses. As the set of nodes of $\mathcal{G}$ are, by construction, propositionally consistent, $f(w)$ in the non-reduced graph is not empty. The deletion process will not remove a node of the behaviour graph that is included in some $f(w)$. Therefore, the reduced behaviour graph is non-empty.

($\Leftarrow$) Assume that the reduced graph $\mathcal{G} = \langle N_0, \ldots, N_{m+1}, E_1, \ldots, E_n \rangle$ constructed from Φ is non-empty. To show that Φ is satisfiable we construct a model M from $\mathcal{G}$. Let $ord : N_i \longrightarrow \mathbb{N}$ be a total order on the nodes in N_i. Let $w_{ml,ord(\eta)}$ be the world named by $(ml, ord(\eta))$ for $\eta \in N_{ml}$ and let $W_{ml} = \bigcup_{\eta \in N_{ml}} \{w_{ml,ord(\eta)}\}$. The set of worlds W is given by $\bigcup_{i=0}^{m+1} W_i$. Let w_0 be any of the worlds in W_0. The pair $(w_{ml,ord(\eta)}, w_{ml',ord(\eta')})$ is in R_a if and only if $(\eta, \eta') \in E_a$. Also, take $R_* = W \times W$. Finally, set $\pi(w_{ml,ord(\eta)})(p) = \mathit{true}$ if and only if $p \in \eta$, for all $p \in P$. This completes the construction of the model $M = (W, w_0, R_1, \ldots, R_n, R_*, \pi)$. That M is indeed a model for Φ follows from the construction of the graph $\mathcal{G}$: if $ml : C$ is a clause in Φ and $w_{ml,ord(\eta)}$ is a world in M, then $\langle M, w_{ml,ord(\eta)} \rangle \models C$, for otherwise η would have been removed during the reduction phase in the construction of the graph. By an easy induction, it follows that M is a model for Φ. □

The completeness proof uses the fact that binary propositional resolution is consequence complete [33]. Given a set of clauses Φ, a clause C is a *prime consequence* of Φ if, and only if, C is implied by Φ and there exists no other clause D implied by Φ such that C is implied by D [62]. Given a set of clauses Φ, a calculus is *consequence complete* if any prime consequence C of Φ is derivable.

We now show that the calculus for global and local reasoning in K_n is complete.

THEOREM 5.12. *Let Φ be an unsatisfiable set of clauses in* SNF_{ml}. *Then there is a refutation by* $\mathsf{RES}_{\mathsf{ml}}$ *for* Φ.

PROOF. Given a set of clauses Φ, construct the reduced behaviour graph as described above. First assume that the set of literal clauses is unsatisfiable. Thus all initial nodes will be removed from the reduced graph and the graph becomes empty. From the completeness of classical resolution there is a series of resolution steps which can be applied to these clauses which lead to the derivation of **false**. The same applies within any modal level. We can mimic these steps by applying the rule LRES to literal clauses and derive $ml :$ **false**, for some modal level ml.

If the non-reduced graph is not empty and we have that both (1) $ml : l' \Rightarrow \boxed{a} l$ and (2) $ml' : l'' \Rightarrow \Diamond_a \neg l$ are in Φ, then, by construction of the graph, if $\{ml, ml'\}$ are unifiable, then any node in $N_{\sigma(\{ml,ml'\})}$ containing both l' and l'' is removed from the graph. The resolution rule MRES applied to (1) and (2) results in $\sigma(\{ml, ml'\}) : \neg l' \vee \neg l''$, simulating the deletion of nodes at the same modal level that satisfy both l' and l''.

Next, if the non-reduced graph is not empty, consider any nodes that do not satisfy the negative a-clauses in Φ. Recall that for each node $\eta_{ml} \in N_{ml}$ and for each agent $a \in A_n$, if $ml : l \Rightarrow \Diamond_a \neg l'$ is in Φ, $\eta_{ml} \models l$ and there is no a-edge between η and a node that satisfies $\neg l'$, then η_{ml} is deleted. We show next what inference rules or what inference steps correspond to the deletion of η_{ml}.

Let $\mathbb{C}_a^{\eta_{ml}}$ in Φ be the set of positive a-clauses corresponding to agent a, that is, the clauses of the form $ml : l_j \Rightarrow \boxed{a} l'_j$, where l_j and l'_j are literals, whose left-hand side are satisfied by η_{ml}. Let $\mathbb{R}_a^{\eta_{ml}}$ be the set of literals in the scope of $\boxed{a}$ on the right-hand side from the clauses in $\mathbb{C}_a^{\eta_{ml}}$, that is, if $ml : l_j \Rightarrow \boxed{a} l'_j \in \mathbb{C}_a^{\eta_{ml}}$, then $l'_j \in \mathbb{R}_a^{\eta_{ml}}$. From the construction of the graph, for a clause $ml : l \Rightarrow \Diamond_a l'$, if $\eta_{ml} \models l$ but there is no a-edge to a node containing l', it means that l', $\mathbb{R}_a^{\eta_{ml}}$, and the literal clauses at the level $ml + 1$ must be contradictory. As l' alone is not contradictory and because the case where the literal clauses are contradictory by themselves has been covered above (by applications of LRES), there are five cases:

(1) Assume that $\mathbb{R}_a^{\eta_{ml}}$ itself is contradictory. This means there must be clauses of the form $ml : l_1 \Rightarrow \boxed{a} l'', ml : l_2 \Rightarrow \boxed{a} \neg l'' \in \mathbb{C}_a^{\eta}$, where $\eta_{ml} \models l_1$ and $\eta_{ml} \models l_2$. Thus we can apply GEN2 to these clauses and the negative modal clause $ml : l \Rightarrow \Diamond_a l'$ deriving

$ml : \neg l_1 \vee \neg l_2 \vee \neg l$. Hence the addition of this resolvent means that η_{ml} will be deleted as required.

(2) Assume that l' and $\mathbb{R}_a^{\eta_{ml}}$ is contradictory. Then, $\mathbb{C}_a^{\eta_{ml}}$ in Φ contains a clause as $ml : l_1 \Rightarrow \boxed{a}\neg l'$ where, from the definition of $\mathbb{C}_a^{\eta_{ml}}$, $\eta_{ml} \models l_1$. Thus, by an application of MRES to this clause and $ml : l \Rightarrow \Diamond_a l'$, we derive $ml : \neg l_1 \vee \neg l$ and η_{ml} is removed as required.

(3) Assume that l' and the literal clauses at the modal level $ml + 1$ are contradictory. By consequence completeness of binary resolution [33], applications of LRES to the set of literal clauses generates $ml + 1 : \neg l'$, which can be used together with $ml : l \Rightarrow \Diamond_a l'$ to apply GEN1 and generate $ml : \neg l$. This resolvent deletes η_{ml} as required. Note that this is a special case where the set of positive a-clauses in the premise of GEN1 is empty.

(4) Assume that $\mathbb{R}_a^{\eta_{ml}}$ and the literal clauses at the modal level $ml + 1$ all contribute to the contradiction (but not l'), by the results in [33], applications of LRES will generate the relevant clause to which we can apply GEN3 and delete η_{ml} as required. Note that this is also the case in the special case where the set of positive clauses in the premise of GEN3 is empty. In this case, the literal clause in the premises is of the form $ml : C$ where C is the empty disjunction, which by consequence completeness will also be produced during a derivation.

(5) Assume that l', $\mathbb{R}_a^{\eta_{ml}}$ and the literal clauses all contribute to the contradiction. Thus, similarly to the above, applying LRES generates the relevant literal clause to which GEN1 can be applied. This deletes η_{ml} as required.

Summarising, LRES corresponds to deletions from the graph of nodes related to contradictions in the set of literal clauses at a particular modal level. The rule MRES also simulates classical resolution and corresponds to removing from the graph those nodes related to contradiction within the set of modal literals occurring at the same modal level. The inference rule GEN1 corresponds to deleting parts of the graph related to contradictions between the literal in the scope of $\Diamond_a$, the set of literal clauses, and the literals in the scope of $\boxed{a}$. The resolution rule GEN2 corresponds to deleting parts of the graph related to contradictions between the literals in the scope of $\boxed{a}$. Finally, GEN3 corresponds to deleting parts of the graph related to contradictions between the literals in the scope of $\boxed{a}$ and the set of literal clauses. These are all possible combinations of contradicting sets within a clause set.

If the resulting graph is empty, the set of clauses Φ is not satisfiable and there is a resolution proof corresponding to the deletion procedure, as described above. If the graph is not empty, by Lemma 5.11, a model for Φ can be built. □

Theorem 5.12 shows that if a set of clauses is unsatisfiable, then there is a refutation by RES_{ml}. The next lemma shows that the deletion of a subsumed clause preserves the inconsistency of a clause set. Note that we only consider the case of subsumption of literal clauses (see Definition 4.3).

Lemma 5.13. *Let Φ be an unsatisfiable set of SNF_{ml} clauses, and $ml_1 : C$ and $ml_2 : D$ in Φ be literal clauses, such that $ml_1 : C$ subsumes $ml_2 : D$. Then, $\Phi \setminus \{ml_2 : D\}$ is unsatisfiable.*

Proof. We prove the contrapositive, i.e. if $\Phi \setminus \{ml_2 : D\}$ is satisfiable, then Φ is satisfiable. If $\Phi \setminus \{ml_2 : D\}$ is satisfiable, then there is a model $M^* = (W, w_0, R_1, \ldots, R_n, R_*, \pi)$ such that $M^* \models \Phi \setminus \{ml_2 : D\}$. By the definition of satisfiability of sets, (1) all clauses in $\Phi \setminus \{ml_2 : D\}$ are satisfied in M^*. In particular, $M^* \models ml_1 : C$. By the definition of satisfiability of labelled clauses, we have that (2) $\langle M^*, w\rangle \models C$, for all $w \in W$ such that $\text{depth}(w) = ml_1$. As $ml_1 : C$ subsumes $ml_2 : D$, by Definition 4.3, $\sigma(\{ml_1, ml_2\}) = ml_2$ and D is of the form $C \vee D'$, for a disjunction of literals D'. From this and from (2), by the semantics of disjunction, we obtain that $\langle M^*, w\rangle \models C \vee D'$, for all

$w \in W$ such that $\mathrm{depth}(w) = ml_2$. Hence, (3) $M^* \models ml_2 : C \vee D'$. It follows, from the definition of satisfiability of sets, from (1) and (3), that $M^* \models \Phi$. □

In the following, we show that our calculus remains complete if in the construction of a refutation we remove subsumed clauses, more precisely, for a derivation $\Phi_0, \Phi_1, \ldots$, we can require that a resolvent D, obtained by the application of one of the inference rules to clauses in Φ_{i-1}, is in simplified form, D is not a tautology, and D is not subsumed by any clause in Φ_{i-1}. We first consider the inference rules LRES, GEN1, and GEN3 before stating the main result.

Lemma 5.14. *Let $ml_1 : C_1$, $ml_2 : C_2$, $ml'_1 : C'_1$, $ml'_2 : C'_2$ be literal clauses such that $ml'_1 : C'_1 \leq_s ml_1 : C_1$ and $ml'_2 : C'_2 \leq_s ml_2 : C_2$. Let $ml : C$ with $ml = \sigma(\{ml_1, ml_2\})$ be a resolvent of $ml_1 : C_1$ and $ml_2 : C_2$ by* LRES. *Then $ml'_1 : C'_1 \leq_s ml : C$ or $ml'_2 : C'_2 \leq_s ml : C$ or there is a resolvent $ml' : C'$ of $ml'_1 : C'_1$ and $ml'_2 : C'_2$ by* LRES *such that $ml' : C' \leq_s ml : C$.*

Proof. Let $ml_1 : C_1$, $ml_2 : C_2$ and $ml : C$ be of the form $ml_1 : D_1 \vee p$, $ml_2 : D_2 \vee \neg p$, and $\sigma(\{ml_1, ml_2\}) : D_1 \vee D_2$, respectively. As $ml'_1 : C'_1$ subsumes $ml_1 : C_1$ it either has the form $ml'_1 : D'_1 \vee p$ or $ml'_1 : D'_1$, where in both cases $\sigma(\{ml_1, ml'_1\}) = ml_1$ and D'_1 only contains literals in D_1. Analogously, as $ml'_2 : C'_2$ subsumes $ml_2 : C_2$ it either has the form $ml'_2 : D'_2 \vee p$ or $ml'_2 : D'_2$, where in both cases $\sigma(\{ml_2, ml'_2\}) = ml_2$ and D'_2 only contains literals in D_2. There are three possibilities to consider:

- If $ml'_1 : C'_1$ has the form $ml'_1 : D'_1$ then it subsumes the resolvent $ml : D_1 \vee D_2$ as (i) $\sigma(\{ml'_1, ml_1\}) = ml_1$ and $\sigma(\{ml_1, ml_2\}) = ml$ imply $\sigma(\{ml'_1, ml\}) = ml$ and (ii) all literals in D'_1 occur in D_1.
- If $ml'_2 : C'_2$ has the form $ml'_2 : D'_2$ then it subsumes the resolvent $ml : D_1 \vee D_2$ as (i) $\sigma(\{ml'_2, ml_2\}) = ml_2$ and $\sigma(\{ml_2, ml_2\}) = ml$ imply $\sigma(\{ml'_2, ml\}) = ml$ and (ii) all literals in D'_2 occur in D_2.
- If $ml'_1 : C'_1$ has the form $ml'_1 : D'_1 \vee p$ and $ml'_2 : C'_2$ has the form $ml'_2 : D'_2 \vee \neg p$, then we first of all observe that $\sigma(\{ml'_1, ml_1\}) = ml_1$, $\sigma(\{ml'_2, ml_2\}) = ml_2$, and $\sigma(\{ml_1, ml_2\}) = ml$. Therefore, $\sigma(\{ml'_1, ml'_2\})$ is defined and there is a resolvent $ml' : D'_1 \vee D'_2$ of the two clauses by LRES, where $ml' = \sigma(\{ml'_1, ml'_2\})$. Furthermore, $\sigma(\{ml', ml\}) = ml$ and all literals in D'_1 occur in D_1 and all literals in D'_2 occur in D_2. Thus, this resolvent subsumes $ml : D_1 \vee D_2$.

It follows that $ml'_1 : C'_1 \leq_s ml : C$ or $ml'_2 : C'_2 \leq_s ml : C$ or there is a resolvent $ml' : C'$ of $ml'_1 : C'_1$ and $ml'_2 : C'_2$ by LRES such that $ml' : C' \leq_s ml : C$. □

Lemma 5.15. *Let $ml_1 : l'_1 \Rightarrow \boxed{a}\neg l_1$, ..., $ml_m : l'_m \Rightarrow \boxed{a}\neg l_m$, and $ml_{m+1} : l'_{m+1} \Rightarrow \langle a \rangle \neg l_{m+1}$ be modal a-clauses, and let $ml : \neg l'_1 \vee \cdots \vee \neg l'_{m+1}$ with $\sigma(\{ml_1, \ldots, ml_{m+1}, ml_{m+2} - 1\}) = ml$ be the literal clause obtained as resolvent by* GEN1 *of a literal clause $ml_{m+2} : l_1 \vee \cdots \vee l_{m+1}$ with these modal a-clauses. Let $ml'_{m+2} : C'_1$ be a literal clause that subsumes $ml_{m+2} : l_1 \vee \cdots \vee l_{m+1}$. Then we can derive a clause $ml' : C'$ that subsumes $ml : \neg l'_1 \vee \cdots \vee \neg l'_{m+1}$.*

Proof. As $ml'_{m+2} : C'_1$ subsumes $ml_{m+2} : l_1 \vee \cdots \vee l_{m+1}$, $\sigma(\{ml'_{m+2}, ml_{m+2}\}) = ml_{m+2}$ and all literals in $ml_{m+2} : C'_1$ are among $\{l_1, \ldots, l_{m+1}\}$. We need to distinguish two cases, namely, whether l_{m+1} occurs in $ml'_{m+2} : C'_1$ or not. First, assume that l_{m+1} occurs in $ml'_{m+2} : C'_1$ and without loss of generality assume that $ml'_{m+2} : C'_1$ has the form $ml'_{m+2} : l_k \vee \cdots \vee l_{m+1}$ for $1 \leq k \leq m+1$. Since $\sigma(\{ml'_{m+2}, ml_{m+2}\}) = ml_{m+2}$ and $\sigma(\{ml_1, \ldots, ml_{m+1}, ml_{m+2} - 1\})$ is defined, $\sigma(\{ml_k, \ldots, ml_{m+1}, ml'_{m+2} - 1\})$ is also defined. Then there is a resolvent $ml' : \neg l'_k \vee \cdots \vee \neg l'_{m+1}$ by GEN1 with $ml' = \sigma(\{ml_k, \ldots, ml_{m+1}, ml'_{m+2} - 1\})$ of $ml'_{m+2} : l_k \vee \cdots \vee l_{m+1}$ with $ml_k : l'_k \Rightarrow \boxed{a} l_k$, ..., $ml_m : l'_m \Rightarrow \boxed{a} l_m$ and $ml_{m+1} : l'_{m+1} \Rightarrow \langle a \rangle l_{m+1}$. We have $\sigma(\{ml', ml\}) = ml$ and every literal in $ml' : \neg l'_k \vee \cdots \vee \neg l'_{m+1}$ occurs in $ml : \neg l'_1 \vee \cdots \vee \neg l'_{m+1}$. Thus, $ml' : \neg l'_k \vee \cdots \vee \neg l'_{m+1}$ subsumes $ml : \neg l'_1 \vee \cdots \vee \neg l'_{m+1}$.

Second, assume that l_{m+1} does not occur in $ml'_{m+2} : C'_1$ and without loss of generality assume that $ml'_{m+2} : C'_1$ has the form $ml'_{m+2} : l_1 \vee \cdots \vee l_k$ for $0 \leq k \leq m$. Since $\sigma(\{ml'_{m+2}, ml_{m+2}\}) = ml_{m+2}$ and $\sigma(\{ml_1, \ldots, ml_k, ml_{m+2} - 1\})$ is defined, $\sigma(\{ml_1, \ldots, ml_k, ml'_{m+2} - 1\})$ is also defined. Then there is a resolvent $ml' : \neg l'_1 \vee \cdots \vee \neg l'_k \vee \neg l'_{m+1}$ by GEN3 with $ml' = \sigma(\{ml_1, \ldots, m_k, ml_{m+1}, ml'_{m+2} - 1\})$ of $ml'_{m+2} : l_1 \vee \cdots \vee l_k$ with $ml_1 : l'_1 \Rightarrow \boxed{a} l_1, \ldots, ml_k : l'_k \Rightarrow \boxed{a} l_k$ and $ml_{m+1} : l'_{m+1} \Rightarrow \langle a \rangle l_{m+1}$. We have $\sigma(\{ml', ml\}) = ml$ and every literal in $ml' : \neg l'_1 \vee \cdots \vee \neg l'_k \vee \neg l'_{m+1}$ occurs in $ml : \neg l'_1 \vee \cdots \vee \neg l'_{m+1}$. Thus, $ml' : \neg l'_1 \vee \cdots \vee \neg l'_k \vee \neg l'_{m+1}$ subsumes $ml : \neg l'_1 \vee \cdots \vee \neg l'_{m+1}$. □

Lemma 5.16. *Let $ml_1 : l'_1 \Rightarrow \boxed{a} l_1, \ldots, ml_m : l'_m \Rightarrow \boxed{a} l_m$, and $ml_{m+1} : l'_{m+1} \Rightarrow \langle a \rangle l_{m+1}$ be modal a-clauses, and let $ml_{m+2} : \neg l'_1 \vee \cdots \vee \neg l'_{m+1}$ with $\sigma(\{ml_1, \ldots, ml_{m+1}, ml_{m+2} - 1\}) = ml$ be the literal clause obtained as resolvent by* GEN3 *of a literal clause $ml_{m+2} : l_1 \vee \cdots \vee l_m$ with these modal a-clauses. Let $ml'_{m+2} : C'_1$ be a literal clause that subsumes $ml_{m+2} : l_1 \vee \cdots \vee l_m$. Then we can derive a clause $ml' : C'$ that subsumes $ml : \neg l'_1 \vee \cdots \vee \neg l'_{m+1}$.*

Proof. The proof proceeds in analogy to the second case in the proof of Lemma 5.15. As $ml'_{m+2} : C'_1$ subsumes $ml_{m+2} : l_1 \vee \cdots \vee l_m$ we have $\sigma(\{ml'_{m+2}, ml_{m+2}\}) = ml_{m+2}$ and all literals in C'_1 are in $\{l_1, \ldots, l_m\}$. Without loss of generality assume that $ml'_{m+2} : C'_1$ has the form $ml'_{m+2} : l_1 \vee \cdots \vee l_k$ for $0 \leq k \leq m$. Since $\sigma(\{ml'_{m+2}, ml_{m+2}\}) = ml_{m+2}$ and $\sigma(\{ml_1, \ldots, ml_k, ml_{m+2} - 1\})$ is defined, $\sigma(\{ml_1, \ldots, ml_k, ml'_{m+2} - 1\})$ is also defined. Then there is a resolvent $ml' : \neg l'_1 \vee \cdots \vee \neg l'_k \vee \neg l'_{m+1}$ by GEN3 with $ml' = \sigma(\{ml_1, \ldots, m_k, ml_{m+1}, ml'_{m+2} - 1\})$ of $ml'_{m+2} : l_1 \vee \cdots \vee l_k$ with $ml_1 : l'_1 \Rightarrow \boxed{a} l_1$, $\ldots$, $ml_k : l'_k \Rightarrow \boxed{a} l_k$ and $ml_{m+1} : l'_{m+1} \Rightarrow \langle a \rangle l_{m+1}$. We have $\sigma(\{ml', ml\}) = ml$ and every literal in $ml' : \neg l'_1 \vee \cdots \vee \neg l'_k \vee \neg l'_{m+1}$ occurs in $ml : \neg l'_1 \vee \cdots \vee \neg l'_{m+1}$. Thus, $ml' : \neg l'_1 \vee \cdots \vee \neg l'_k \vee \neg l'_{m+1}$ subsumes $ml : \neg l'_1 \vee \cdots \vee \neg l'_{m+1}$. □

Theorem 5.17. *Let Φ be an unsatisfiable set of* SNF_{ml} *clauses. Then there is a refutation of Φ by* $\mathsf{RES}_{\mathsf{ml}}$ *of the form $\Phi = \Phi_0, \ldots, \Phi_n$ where for every i, $0 \leq i \leq n-1$, $\Phi_{i+1} = \Phi_i \cup \{ml_i : C_i\}$, $ml_i : C_i$ is not subsumed by a clause in Φ_i, and no premise $ml'_i : C'_i$ used in the derivation of $ml_i : C_i$ is subsumed by a clause distinct to $ml'_i : C'_i$ in Φ_i.*

Proof. Let Φ' be obtained from Φ by removing all subsumed clauses, that is, all clauses of the form $ml : D$ such that there is $ml' : C \in \Phi$ and $ml' : C \leq_s ml : D$. By Lemma 5.13, Φ' is unsatisfiable. By Theorem 5.12, there is a refutation of $\Phi' = \Phi'_0, \ldots, \Phi'_k$ of Φ' by $\mathsf{RES}_{\mathsf{ml}}$.

By induction over $\Phi'_0, \ldots, \Phi'_k$ we construct a refutation $\Phi_0, \Phi_1, \ldots, \Phi_n$, $n \leq k$, and a monotonically increasing partial function $\lambda : \mathbb{N} \longrightarrow \mathbb{N}$ such that for all i, $0 \leq i \leq k$, (1) $\lambda(i) \leq i \leq n$, (2) if $i > 0$ and $\Phi_{\lambda(i)} = \Phi_{\lambda(i)-1} \cup \{ml : C\}$ then $ml : C$ is not subsumed by a clause in $\Phi_{\lambda(i)-1}$, (3) for all clauses $ml'_1 : C'_1 \in \Phi'_i$ there exists a clause $ml_1 : C_1 \in \Phi_{\lambda(i)}$ such that $ml_1 : C_1$ subsumes $ml'_1 : C'_1$.

For the base case, we define $\Phi_0 = \Phi'_0$ and $\lambda(0) = 0$. Obviously, Property (1), $\lambda(0) \leq 0 \leq n$, holds. Property (3) holds as $\Phi_{\lambda(0)}$ and Φ_0 are identical. Property (2) holds as $i = 0$.

Now assume that we have proceeded to Φ'_i in $\Phi'_0, \ldots, \Phi'_k$ and have constructed a derivation $\Phi_0, \ldots, \Phi_i$, $0 \leq i < k$, and defined $\lambda(j)$, for all j, $0 \leq j \leq i$, with the desired properties.

In the induction step we consider Φ'_{i+1}. In the refutation $\Phi'_0, \Phi'_1, \ldots, \Phi'_k$, $\Phi'_{i+1} = \Phi'_i \cup \{ml' : D'\}$ where $ml' : D'$ is a literal clause derived by an application of one of the inference rules of $\mathsf{RES}_{\mathsf{ml}}$ to premises in Φ'_i. By Property (3), for each such premise $ml'' : D''$, $\Phi_{\lambda(i)}$ contains a clause subsuming $ml'' : D''$.

It is also possible that $\Phi_{\lambda(i)}$ already contains a clause $ml : D$ that subsumes $ml' : D'$. We then define $\lambda(i+1) = \lambda(i)$. Regarding Properties (1), as by induction hypothesis $\lambda(i) \leq i$ we have $\lambda(i+1) \leq i+1$. Properties (2) and (3) hold for $i+1$ as, by induction hypothesis, they hold for i.

Assume that $ml' : D'$ is not subsumed by a clause in $\Phi_{\lambda(i)}$. We distinguish the following cases depending on which inference rule was used to derive $ml : D$:

- If $ml' : D'$ was derived by an application of one of the inference rules LRES, GEN1 and GEN3, then by Lemmas 5.14, 5.15 and 5.16, respectively, either $\Phi_{\lambda(i)}$ contains a clause $ml : D$ that subsumes $ml' : D'$, or we can derive a clause $ml : D$ from $\Phi_{\lambda(i)}$ that subsumes $ml' : D'$. We only need to consider the second case. Here we define $\lambda(i + 1) = \lambda(i) + 1$ and $\Phi_{\lambda(i+1)} = \Phi_{\lambda(i)} \cup \{ml : D\}$. Properties (1) to (3) hold by construction.
- If $ml' : D'$ was derived by an application of one of the inference rules MRES and GEN2 from premises in Φ'_i, then these premises were already present in Φ'_0 as our calculus does not derive new modal clauses. By construction, these premises are then also in $\Phi_{\lambda(0)} = \Phi_0$ and in $\Phi_{\lambda(i)}$. Thus, $ml' : D'$ can be derived from $\Phi_{\lambda(i)}$. We define $\lambda(i + 1) = \lambda(i) + 1$ and $\Phi'_{\lambda(i+1)} = \Phi'_{\lambda(i)} \cup \{ml : D\}$. Properties (1) and (3) hold by construction. Regarding Property (2) we have assumed that $ml' : D'$ is not subsumed by a clause in $\Phi_{\lambda(i)}$, therefore Property (2) holds for $i + 1$.

This completes the induction step of our proof. Once the construction has proceeded to Φ'_k, we have obtained a derivation $\Phi_0, \Phi_1, \ldots, \Phi_n$, with $n = \lambda(k)$, in which the clause $ml : C$ does not occur. By Property (3), since Φ_k contains an empty clause $ml'' : \mathbf{false}$ with $ml'' \in \{0, *\}$, Φ'_n contains a clause subsuming $ml'' : \mathbf{false}$. Thus, $\Phi'_0, \Phi'_1, \ldots, \Phi'_n$ is a refutation. □

6 REFINEMENTS

In this section we propose two refinements for the calculus given in Section 4, namely negative and ordered resolution. Both refinements restrict the clauses that can be selected for applying the inference rules. Negative resolution [54] is a special case of semantic resolution [61], which restricts clause selection by using an interpretation as a guide. For the classical case, given an interpretation π, the (binary) semantic resolution rule is given by:

$$[\text{S-RES}]\quad \begin{array}{ccc} D & \vee & l \\ D' & \vee & \neg l \\ \hline D & \vee & D' \end{array}$$

where resolution can be applied only if one of the clauses in the premises is a *nucleus*, that is, a clause which evaluates to *true* under π. By taking $\pi(p) = \mathit{false}$, for all propositional symbols p, semantic resolution corresponds to *negative* resolution. Semantic resolution is complete irrespective of the interpretation chosen to guide the search for a proof [61]. Moreover, semantic resolution is also *consequence complete* [62]. The following theorem, which follows directly from the consequence completeness of semantic resolution, holds:

Theorem 6.1 ([62, Theorem 8]). *If a clause C is a prime consequence of a finite set Φ of clauses and contains no negative (positive) literals, then there is a positive (negative) resolution derivation of C from Φ.*

Theorem 6.1 ensures that all clauses containing only negated propositional symbols and which are consequences of a set of clauses are generated by applications of negative resolution to the clause set. We also note that the choice of literals to resolve during the application of the semantic resolution rule can also be restricted to those which are maximal with respect to a given ordering over the literals. However, for ordered semantic resolution, consequence completeness does not hold for all orderings [38].

In the next sections, we show how restrictions can be applied to the modal case whilst retaining completeness. As motivated by examples and discussed below, few adaptions to the normal form are required. As the calculus operates on a set of clauses, we do not consider improvements that could be applied to the transformation of a formula into the normal form, but the transformations that should be modularly applied to any given clause set in such a way that the calculus together with either of the proposed refinements is complete.

6.1 Negative Resolution

Negative resolution was introduced in [54] as a refinement for the hyper-resolution method, which restricts the clauses that are candidates to being resolved. The following definitions are needed.

Definition 6.2. A literal is said to be *negative* if it is the negation of a propositional symbol. A clause is said to be *negative* if it contains only negative literals.

In the classical case, the negative binary resolution inference rule is exactly as S-RES, where the resolution rule can only be applied if one of the clauses being resolved is negative. For the refinement of the resolution calculus given here, we define a literal clause $ml : D$ to be negative if, and only if, D is a negative clause. Restricting the modal calculus to negative resolution means that at least one of the literal clauses in the premises of inference rules is a negative literal clause, that is, the restriction takes place in the application of the inference rules LRES, GEN1, and GEN3. As it is, the calculus is not complete for negative resolution, as shown in Example 6.3.

Example 6.3. Consider the following set of clauses:

$$\begin{array}{ll}
1. & 0 : t_0 \\
2. & 0 : t_0 \Rightarrow \boxed{a} t_1 \\
3. & 0 : t_0 \Rightarrow \langle a \rangle \neg p \\
4. & 1 : \neg t_1 \vee p \vee \neg q \\
5. & 1 : \neg t_1 \vee p \vee q
\end{array}$$

Resolving Clauses (4) and (5), by an application of LRES, results in:

$$6. \quad 1 : \neg t_1 \vee p \quad [\mathsf{LRES}, 4, 5, q]$$

which can be resolved with Clauses (2) and (3), by an application of GEN1, generating:

$$7. \quad 0 : \neg t_0 \quad [\mathsf{GEN1}, 6, 2, 3, t_1, p]$$

Applying LRES to Clauses (1) and (7) produces the empty clause. However, if the application of LRES is restricted to negative resolution, Clause (6) would not have been generated and no proof would have been found.

The calculus can, however, be restricted to negative resolution with a small change in the normal form by allowing only positive literals in the scope of modal operators. Given a set of clauses in SNF_{ml}, we apply the following transformation to all modal clauses (where $ml \in \mathbb{N} \cup \{*\}$, $t \in L$, $p \in P$, and t' is a new propositional symbol):

$$\begin{array}{lcl}
\rho(ml : t \Rightarrow \boxed{a} \neg p) & = & (ml : t \Rightarrow \boxed{a} t') \wedge \rho(ml + 1 : t' \Rightarrow \neg p) \\
\rho(ml : t \Rightarrow \langle a \rangle \neg p) & = & (ml : t \Rightarrow \langle a \rangle t') \wedge \rho(ml + 1 : t' \Rightarrow \neg p)
\end{array}$$

It can be shown that the resulting set of clauses is satisfiable if, and only if, the original set of clauses is satisfiable. We call the resulting normal form SNF^+_{ml}. As the resulting set of clauses is still in SNF_{ml}, it follows immediately that the original calculus $\mathsf{RES}_{\mathsf{ml}}$ is terminating, sound, and complete for SNF^+_{ml}. We denote by $\mathsf{RES}^{\mathsf{neg}}_{\mathsf{ml}}$ the resolution calculus resulting by restricting $\mathsf{RES}_{\mathsf{ml}}$ to negative resolution. Obviously, clause selection does not have any impact on soundness and termination. It rests to prove that restricting the application of the resolution rules to the case where at least one of the clauses is negative is complete.

Theorem 6.4. *Let Φ be a set of clauses in* SNF^{+}_{ml}. *If Φ is unsatisfiable, then there is a refutation by* $\mathsf{RES}^{\mathsf{neg}}_{\mathsf{ml}}$ *from Φ.*

Proof. We examine all cases of Theorem 5.12 which show that the applications of the inference rules correspond to deletions of nodes in the behaviour graph. For sets of literal clauses, negative resolution is a complete refutation strategy [54]. Hence, if the set of literal clauses at the modal level ml is contradictory, then ml : **false** is generated by applications of the negative version of LRES. Under the new normal form, the inference rules MRES and GEN2 can never be applied. Thus, Cases 1 and 2 of Theorem 5.12 hold trivially. For the case where the literals in the scope of modal operators contradict with the set of literal clauses (Cases 3, 4, and 5 in the proof of Theorem 5.12), the proof follows from the fact that negative resolution is also consequence complete [62]: as all negative clauses which are consequences of the set of literal clauses are generated, the rules GEN1 and GEN3 can be applied as follows. For Case 3, if $ml : l \Rightarrow \diamondsuit l'$ occurs in the clause set, and l' and the literal clauses at the next modal level are contradictory, because negative resolution is consequence complete and $ml + 1 : \neg l'$ is a negative clause, then the application of the negative version of LRES to the set of literal clauses generates $ml + 1 : \neg l'$, to which GEN1 can be applied. For Case 4, if $ml : l \Rightarrow \diamondsuit l'$, $ml : l_1 \Rightarrow \boxed{a} l'_1, \ldots, ml : l_m \Rightarrow \boxed{a} l'_m$ occur in the clause set and the literals $l'_1, \ldots, l'_m$ and the literal clauses at the modal level $m + 1$ are contradictory, then $\neg l'_1 \vee \ldots \vee \neg l'_m$, which is a negative clause, is a consequence of the set of literal clauses. By consequence completeness of negative resolution, applications of the negative version of LRES to the set of literal clauses generates $ml + 1 : \neg l'_1 \vee \ldots \vee \neg l'_m$, to which GEN3 can be applied. For Case 5, if $ml : l \Rightarrow \diamondsuit l'$, $ml : l_1 \Rightarrow \boxed{a} l'_1, \ldots, ml : l_m \Rightarrow \boxed{a} l'_m$ occur in the clause set and the literals $l'_1, \ldots, l'_m, l'$ and the literal clauses at the modal level $m + 1$ are contradictory, then $\neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l'$, which is a negative clause, is a consequence of the set of literal clauses. By consequence completeness of negative resolution, applications of the negative version of LRES to the set of literal clauses generates $ml + 1 : \neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l'$, to which GEN1 can be applied. As in all cases the negative version of LRES produces the needed clauses corresponding to deletions on the behaviour graph, if the set of clauses Φ is not satisfiable, then there is a negative resolution proof corresponding to the deletion procedure described in Section 5. □

Example 6.5. We show a negative refutation for the set of clauses given in Example 6.3. Clauses (3') and (6') are introduced in order to obtain a set of clauses in SNF^{+}_{ml}.

1.	$0 : t_0$
2.	$0 : t_0 \Rightarrow \boxed{a} t_1$
3′.	$0 : t_0 \Rightarrow \diamondsuit t_2$
4.	$1 : \neg t_1 \vee p \vee \neg q$
5.	$1 : \neg t_1 \vee p \vee q$
6′	$1 : \neg t_2 \vee \neg p$

The refutation proceeds as follows, where the negative clauses are underlined in the justification for each of the obtained resolvents:

7′	$1 : \neg t_1 \vee \neg t_2 \vee \neg q$	$[\mathsf{LRES}, \underline{6'}, 4, p]$
8′	$1 : \neg t_1 \vee \neg t_2 \vee q$	$[\mathsf{LRES}, \underline{6'}, 5, p]$
9′	$1 : \neg t_1 \vee \neg t_2$	$[\mathsf{LRES}, \underline{7'}, 8', q]$
10′	$0 : \neg t_0$	$[\mathsf{GEN1}, 2, 3', \underline{9'}, t_1, t_2]$
11′	$0 :$ **false**	$[\mathsf{LRES}, \underline{10'}, 1, t_0]$

6.2 Ordered Resolution

Ordered resolution is a refinement of resolution where inferences are restricted to maximal literals in a clause, with respect to a well-founded ordering on literals. Formally, let Φ be a set of clauses and P_Φ be the set of propositional symbols occurring in Φ. Let $\succ$ be a well-founded and total ordering on P_Φ. This ordering can be extended to literals L_Φ occurring in Φ by setting $\neg p \succ p$ and $p \succ \neg q$ whenever $p \succ q$, for all $p, q \in P_\Phi$. A literal l is said to be *maximal* with respect to a clause $C \vee l$ if, and only if, there is no l' occurring in C such that $l' \succ l$. In the case of classical binary resolution, the ordering refinement restricts the application to clauses $C \vee l$ and $D \vee \neg l$ where l is maximal with respect to C and $\neg l$ is maximal with respect to D. Ordered resolution is refutation complete [27] and it has been successfully applied as the core strategy for many automated theorem proving tools for both classical and modal logics [4, 29, 58, 64, 66]. It has also been shown that classical hyper-resolution is complete under ordering refinements for any ordering on the set of literals [10]. Restricting resolution by orderings has been proved complete for hybrid logics in [3].

We show that the restriction given by ordered resolution cannot be easily applied to the calculus given in Section 4. From the results in [10], any ordering over literals can be used to find contradictions at the propositional fragment of the language by restricting the application of LRES. However, the application of the hyper-resolution rules (GEN1 and GEN3) requires that the relevant literal clauses for applying those inference rules are generated. As ordered resolution lacks consequence completeness [38], such clauses cannot be derived for all orderings, as we show in the next two examples.

Example 6.6. Consider the following set of clauses:

1.	$0:$	t_0
2.	$0:$	$t_0 \Rightarrow \Box t_1$
3.	$1:$	$\neg t_1 \vee p$
4.	$1:$	$\neg t_1 \vee q$
5.	$0:$	$t_0 \Rightarrow \Diamond t_2$
6.	$1:$	$\neg t_2 \vee \neg p \vee \neg q$

which is unsatisfiable: applying LRES to Clauses (3), (4), and (6), we obtain $1 : \neg t_1 \vee \neg t_2$; this clause can then be resolved, by an application of GEN1, with Clauses (2) and (5), generating $0 : \neg t_0$, which contradicts with Clause (1) at the modal level 0.

The ordering given by $t_0 \succ t_1 \succ t_2 \succ p \succ q$ does not allow any inference rule to be applied if LRES is restricted to ordered resolution. Reversing the ordering allows a refutation to be found for this particular example. In the next example, we show a refutation for $\Diamond_{1}(p \wedge \Diamond_{2} q) \wedge \boxed{1}(\neg p \vee \boxed{2}\neg q)$.

Example 6.7. Consider the following set of clauses, where literals are ordered within each clause, that is, the rightmost literal is the maximal literal with respect to each clause ($p \succ q \succ t_0 \succ t_1 \succ t_2 \succ t_3$).

1.	$0:$	t_0	
2.	$0:$	$t_0 \Rightarrow \Diamond_{1} t_1$	
3.	$0:$	$t_0 \Rightarrow \boxed{1} t_2$	
4.	$1:$	$\neg t_1 \vee p$	
5.	$1:$	$t_3 \vee \neg t_2 \vee \neg p$	
6.	$1:$	$t_1 \Rightarrow \Diamond_{2} q$	
7.	$1:$	$t_3 \Rightarrow \boxed{2} \neg q$	
8.	$1:$	$\neg t_3 \vee \neg t_1$	$[\mathsf{MRES}, 6, 7, q]$

$$
\begin{array}{rlll}
9. & 1: & \neg t_2 \vee \neg t_1 \vee \neg p & [\mathrm{LRES}, 8, 5, t_3] \\
10. & 1: & \neg t_2 \vee \neg t_1 & [\mathrm{LRES}, 9, 4, p] \\
11. & 0: & \neg t_0 & [\mathrm{GEN1}, 10, 2, 3, t_1, t_2] \\
12. & 0: & \mathbf{false} & [\mathrm{LRES}, 11, 1, t_0]
\end{array}
$$

The set of clauses given in Example 6.7 is unsatisfiable, as shown in the refutation above. However, for the given ordering, the literal t_3 in Clause (8) cannot be resolved with its complement in Clause (5), as $\neg t_3$ is not the maximal literal in this clause. Thus, by using the ordered version of LRES, a refutation for this set and this particular ordering does not exist.

The examples show that finding an ordering for which the ordered version of the calculus is complete is not trivial. The key idea for achieving completeness is to introduce new literals in the scope of the modal operators and set their ordering to be "low enough" so that the relevant literal clauses needed for the modal hyper-resolution rules are generated. Given a set of clauses Φ in SNF_{ml} and a well-founded and total ordering $\succ$ on P_Φ, we apply the following transformation to all modal clauses (where $ml \in \mathbb{N} \cup \{*\}$, $t, l \in L$ and t' is a new propositional symbol):

$$
\begin{array}{rcl}
\rho(ml : t \Rightarrow \boxed{a} l) & = & (ml : t \Rightarrow \boxed{a} t') \wedge \rho(ml+1 : t' \Rightarrow l) \\
\rho(ml : t \Rightarrow \Diamond_a l) & = & (ml : t \Rightarrow \Diamond_a t') \wedge \rho(ml+1 : t' \Rightarrow l)
\end{array}
$$

and extend the given ordering by setting $p \succ t'$, for all p occurring in Φ. We call the resulting normal form SNF^{++}_{ml}. Note that we only need to apply the rewriting rule to the clauses in Φ, but not to the generated clauses in SNF^{++}_{ml}. Thus, the rewriting procedure is terminating. Again, it is easy to show that Φ is satisfiable if, and only if, the resulting set of clauses in SNF^{++}_{ml} is satisfiable. For a set of clauses $\Phi = \{\varphi_1, \ldots, \varphi_m\}$, we denote by $\rho(\Phi)$ the set resulting of the application of ρ to each formula in Φ, that is, $\rho(\Phi) = \rho(\varphi_1) \cup \ldots \cup \rho(\varphi_m)$. We denote by $\mathsf{RES}^{\mathsf{ord}}_{\mathsf{ml}}$ the resolution calculus resulting by restricting $\mathsf{RES}_{\mathsf{ml}}$ to ordered resolution.

In order to show completeness, we have to show that given a set of SNF^{++}_{ml} clauses, all the cases of Theorem 5.12 hold. We first note that, for a modal level ml, the fact that there is a refutation for an unsatisfiable set of literal clauses, for any ordering, follows from [10]. For the same reason, for a contradictory set of clauses at the modal level ml, the clause ml : **false** will be derived and the rule GEN3 in the special case where $m = 0$ can be applied if there is any negative modal clause at the level $ml - 1$. Also, as there are only positive literals in the scope of modal operators, the rules GEN2 and MRES cannot be applied. Thus, Cases 1 and 2 of Theorem 5.12 hold trivially. It rests to show that the inference rules GEN1 and GEN3 will be applied whenever a contradiction between literals in the scope of modal operators in modal clauses and the set of literal clauses occur. We recall that GEN1 should be applied whenever the set of literals occurring in the scope of modal literals and the literal clauses are contradictory, which corresponds to Cases 3 and 5 of Theorem 5.12; GEN3 should be applied whenever the set of literals occurring in the scope of modal operators in a-positive clauses and the set of literal clauses is contradictory, which corresponds to Case 4 of Theorem 5.12.

The next lemma shows that the relevant literal clause for the application of GEN1 is generated.

Lemma 6.8. *Let Φ be a set of clauses in* SNF_{ml} *with $\chi = \{ml : l'_1 \Rightarrow \boxed{a}\neg l_1, \ldots, ml : l'_m \Rightarrow \boxed{a}\neg l_m, ml : l' \Rightarrow \Diamond_a \neg l\} \subseteq \Phi$. Let $\rho(\chi) = \chi'_{ml} \cup \chi'_{ml+1}$ with $\chi'_{ml} = \{ml : l'_1 \Rightarrow \boxed{a} l''_1, \ldots, ml : l'_m \Rightarrow \boxed{a} l''_m, ml : l' \Rightarrow \Diamond_a l''\}$ and $\chi'_{ml+1} = \{ml+1 : \neg l''_1 \vee \neg l_1, \ldots, ml+1 : \neg l''_m \vee \neg l_m, ml+1 : \neg l'' \vee \neg l\}$. Let $\Phi_{ml+1} = \{C \mid ml+1 : C \in \Phi$ and C is a disjunction of literals$\}$ be the set of literal clauses at the modal level $ml+1$ and $\Phi'_{ml+1} = \{ml+1 : C \mid C$ is a disjunction of literals$\} \subseteq \Phi$ be the set of labelled modal clauses at this same modal level. If $\Xi = \{\neg l_1, \ldots, \neg l_m, \neg l\} \cup \Phi_{ml+1}$ is unsatisfiable and every strict subset of Ξ is satisfiable, then there is a derivation of $ml+1 : \neg l''_1 \vee \ldots \vee \neg l''_m \vee \neg l''$ from $\Xi' = \chi'_{ml+1} \cup \Phi'_{ml+1}$ by applications of ordered* LRES.

Proof. If Ξ is unsatisfiable then, by the results in [10], there is a refutation for Ξ, for any ordering over the literals occurring in Ξ. Let $\Psi_0, \ldots, \Psi_k$, with $\Psi_0 = \Xi$ and **false** $\in \Psi_k$ be such a refutation. We inductively construct a derivation $\Psi'_0, \ldots, \Psi'_k$ such that for every i, $1 \leq i \leq k$, the following Property (1) holds: for every clause $C \in \Psi_i$ there exists a clause $ml + 1 : C \lor D \in \Psi'_i$, such that all literals in D are in $\{\neg l''_1, \ldots, \neg l''_m, \neg l''\}$ and every literal in D is smaller than any literal in C.

The construction starts with $\Psi'_0 = \Xi' = \chi'_{ml+1} \cup \Phi'_{ml+1}$. By definition of Ψ'_0, every clause in Ψ'_0 is of the form $ml+1 : C \lor D$, where D is either the empty disjunction or is a literal in $\{\neg l''_1, \ldots, \neg l''_m, \neg l''\}$, every literal in D is smaller than any literal in C, and C is either a literal in $\{\neg l_1, \ldots, \neg l_m, \neg l\}$ or is a literal clause in Φ_{ml+1}. Note, in particular, that each literal in $\{\neg l_1, \ldots, \neg l_m, \neg l\}$ corresponds to one of the clauses in $\chi'_{ml+1} = \{ml + 1 : \neg l''_1 \lor \neg l_1, \ldots, ml + 1 : \neg l''_m \lor \neg l_m, ml + 1 : \neg l'' \lor \neg l\}$. Thus, $C \in \Xi = \Psi_0$ and Property (1) holds for Ψ_0 and Ψ'_0.

Assume that $\Psi'_0, \ldots, \Psi'_i$ has been constructed and that Property (1) holds for Ψ_i and Ψ'_i.

Let $\Psi_{i+1} = \Psi_i \cup \{C \lor C'\}$, where $C \lor C'$ is the resolvent of the application of ordered LRES to $\{C \lor l_d, C' \lor \neg l_d\} \in \Psi_i$, for some literal l_d. Clearly, l_d and $\neg l_d$ are maximal with respect to C and C', respectively; otherwise, the clause $C \lor C'$ would have not been derived by ordered resolution and added to Ψ_{i+1}. As Property (1) holds for Ψ_i and Ψ'_i, there exist clauses $ml + 1 : C \lor D \lor l_d$ and $ml + 1 : C' \lor D' \lor \neg l_d$ in Ψ'_i all the literals in D and D' are in $\{\neg l''_1, \ldots, \neg l''_m, \neg l''\}$ and are smaller than every literal in C and C'. Thus, ordered LRES can be applied to $ml + 1 : C \lor D \lor l_d$ and $ml + 1 : C' \lor D' \lor \neg l_d$ in Ψ'_i and we obtain $\Psi'_{i+1} = \Psi'_i \cup \{ml + 1 : C \lor C' \lor D \lor D'\}$. Property (1) holds for Ψ_{i+1} and Ψ'_{i+1}.

Since **false** $\in \Psi_k$ there must then be a clause $ml + 1 : D \in \Psi'_k$ such that all literals in D are in $\{\neg l''_1, \ldots, \neg l''_m, \neg l\}$. Because every strict subset of Ξ is satisfiable, all the literals in $\{\neg l_1, \ldots, \neg l_m, \neg l\}$ will be premises in an inference step in $\Psi_0, \ldots, \Psi_k$. Therefore, *all* clauses in $\chi'_{ml+1} = \{ml + 1 : \neg l''_1 \lor \neg l_1, \ldots, ml + 1 : \neg l''_m \lor \neg l_m, ml + 1 : \neg l'' \lor \neg l\}$ also contribute to the derivation of $ml + 1 : D$. Note that there are no positive occurrences of l''_i, $1 \leq i \leq m$, and l'' in Ξ', so once one of these literal occurs negatively in a clause, it cannot be resolved away. Thus, $ml + 1 : D$ is exactly the clause $ml + 1 : \neg l''_1 \lor \ldots \lor \neg l''_m \lor \neg l''$. □

Lemma 6.8 shows that by applying further transformation to the set of clauses and ensuring that the new literals are minimal with respect to the given ordering, the relevant literal clause is generated by the restricted version of the propositional resolution inference rule. Thus, GEN1 can be applied as given in the proof of Cases 3 and 5 of Theorem 5.12. The proof that GEN3 can also be applied (Case 4 of Theorem 5.12) is very similar and omitted here. This shows the completeness of $\mathsf{RES}^{\mathsf{ord}}_{\mathsf{ml}}$ on sets of clauses in SNF^{++}_{ml}.

Theorem 6.9. *Let Φ be a set of clauses in SNF^{++}_{ml}. If Φ is unsatisfiable, then there is a refutation by $\mathsf{RES}^{\mathsf{ord}}_{\mathsf{ml}}$ from Φ.*

In Examples 6.10 and 6.11 below, we show how the clauses introduced in Examples 6.6 and 6.7 can be transformed into SNF^{++}_{ml} and then refuted using ordered resolution.

Example 6.10. Let Φ be the set of clauses given in Example 6.6 where the maximal literal in each literal clause is underlined, that is,

$$\begin{array}{lll}
1. & 0: & \underline{t_0} \\
2. & 0: & t_0 \Rightarrow \Box t_1 \\
3. & 1: & \underline{\neg t_1} \lor p \\
4. & 1: & \underline{\neg t_1} \lor q \\
5. & 0: & t_0 \Rightarrow \Diamond t_2 \\
6. & 1: & \underline{\neg t_2} \lor \neg p \lor \neg q
\end{array}$$

and $t_0 \succ t_1 \succ t_2 \succ p \succ q$ be the ordering of literals. From Φ, we obtain the following set of SNF^{++}_{ml} clauses, where Clause 2 (resp. Clause 5) from above has been replaced by Clauses 2' and 7 (resp. Clauses 5' and 8), the ordering over the literals is extended such that $q \succ t_3 \succ t_4$, and the maximal literals in each literal clause are underlined:

$$\begin{array}{rlll}
1. & 0: & \underline{t_0} & \\
2'. & 0: & t_0 \Rightarrow \Box t_3 & \\
3. & 1: & \underline{\neg t_1} \lor p & \\
4. & 1: & \underline{\neg t_1} \lor q & \\
5'. & 0: & t_0 \Rightarrow \Diamond t_4 & \\
6. & 1: & \underline{\neg t_2} \lor \neg p \lor \neg q & \\
7. & 1: & \neg t_3 \lor \underline{t_1} & [\mathsf{SNF}^{++}_{ml}, 2] \\
8. & 1: & \neg t_4 \lor \underline{t_2} & [\mathsf{SNF}^{++}_{ml}, 5]
\end{array}$$

The refutation restricted by ordering follows:

$$\begin{array}{rlll}
9. & 1: & \neg t_3 \lor \underline{p} & [\mathrm{LRES}, 7, 3, t_1] \\
10. & 1: & \neg t_4 \lor \underline{\neg p} \lor \neg q & [\mathrm{LRES}, 8, 6, t_2] \\
11. & 1: & \neg t_4 \lor \neg t_3 \lor \underline{\neg q} & [\mathrm{LRES}, 10, 9, p] \\
12. & 1: & \neg t_3 \lor \underline{q} & [\mathrm{LRES}, 7, 4, t_1] \\
13. & 1: & \underline{\neg t_3} \lor \neg t_4 & [\mathrm{LRES}, 12, 11, q] \\
14. & 0: & \underline{\neg t_0} & [\mathrm{GEN1}, 13, 2', 5', t_4, t_3] \\
15. & 1: & \mathbf{false} & [\mathrm{LRES}, 14, 1, t_0]
\end{array}$$

Example 6.11. We start with the following set of SNF_{ml} clauses given in Example 6.7 where literals are ordered within each clause, that is, the rightmost literal, which is underlined, is the maximal literal with respect to each clause ($p \succ q \succ t_0 \succ t_1 \succ t_2 \succ t_3$).

$$\begin{array}{rll}
1. & 0: & \underline{t_0} \\
2. & 0: & t_0 \Rightarrow \langle 1 \rangle t_1 \\
3. & 0: & t_0 \Rightarrow \boxed{1} t_2 \\
4. & 1: & \neg t_1 \lor \underline{p} \\
5. & 1: & t_3 \lor \neg t_2 \lor \underline{\neg p} \\
6. & 1: & t_1 \Rightarrow \langle 2 \rangle q \\
7. & 1: & t_3 \Rightarrow \boxed{2} \neg q
\end{array}$$

The set of SNF^{++}_{ml} clauses is given below, where Clause (2) (resp. Clause (3), Clause (6) and Clause (7)) from above has been replaced by Clauses (2') and (8) (Clauses (3') and (9); Clauses (6') and (10); and Clauses (7') and (11)) and the ordering over the literals is extended such that $t_3 \succ t_4 \succ t_5 \succ t_6 \succ t_7$.

$$\begin{array}{rll}
1. & 0: & \underline{t_0} \\
2'. & 0: & t_0 \Rightarrow \langle 1 \rangle t_4 \\
3'. & 0: & t_0 \Rightarrow \boxed{1} t_5 \\
4. & 1: & \neg t_1 \lor \underline{p} \\
5. & 1: & t_3 \lor \neg t_2 \lor \underline{\neg p} \\
6'. & 1: & t_1 \Rightarrow \langle 2 \rangle t_6 \\
7'. & 1: & t_3 \Rightarrow \boxed{2} t_7 \\
8. & 1: & \neg t_4 \lor \underline{t_1} \\
9. & 1: & \neg t_5 \lor \underline{t_2}
\end{array}$$

10.	2 :	$\neg t_6 \vee \underline{q}$
11.	2 :	$\neg t_7 \vee \underline{\neg q}$

and the refutation proceeds as follows:

12.	2 :	$\neg t_7 \vee \underline{\neg t_6}$	$[\mathrm{LRES}, 10, 11, q]$
13.	1 :	$\neg t_3 \vee \underline{\neg t_1}$	$[\mathrm{GEN1}, 12, 6', 7', t_6, t_7]$
14.	1 :	$t_3 \vee \neg t_2 \vee \underline{\neg t_1}$	$[\mathrm{LRES}, 4, 5, p]$
15.	1 :	$\neg t_4 \vee t_3 \vee \underline{\neg t_2}$	$[\mathrm{LRES}, 14, 8, t_1]$
16.	1 :	$\neg t_5 \vee \neg t_4 \vee \underline{t_3}$	$[\mathrm{LRES}, 15, 9, t_2]$
17.	1 :	$\neg t_4 \vee \underline{\neg t_3}$	$[\mathrm{LRES}, 13, 8, t_1]$
18.	1 :	$\neg t_5 \vee \underline{\neg t_4}$	$[\mathrm{LRES}, 16, 17, t_3]$
19.	0 :	$\underline{\neg t_0}$	$[\mathrm{GEN1}, 18, 2', 3', t_4, t_5]$
20.	0 :	**false**	$[\mathrm{LRES}, 19, 1, t_0]$

7 EXPERIMENTAL EVALUATION

K$_S$P [41, 43, 44] is an implementation, written in C, of the calculus presented in Section 4. The sources as well as instructions for installing and running the prover can be found in [42]. The prover was designed to support experimentation with different combinations of refinements of its basic calculus. Refinements and options for (pre)processing the input are coded as independently as possible in order to allow for the easy addition and testing of new features. The main loop is based on the given-clause algorithm implemented in Otter [37], a variation of the set of support strategy [69], a refinement which restricts the set of choices of clauses participating in a derivation step.

We have evaluated the different refinements implemented in K$_S$P over two collections of modal formulae:

(1) The complete set of TANCS-2000 modalised random QBF (MQBF) formulae [36] complemented by the additional MQBF formulae provided by Kaminski and Tebbi [30]. This collection consists of 1016 formulae in total, of which 617 are known to be satisfiable and 399 are known to be unsatisfiable. The minimum modal depth of formulae in this collection is 19, the maximum 225, average 69.2 with a standard deviation of 47.5.

(2) Selected instances from the 18 classes of parameterised LWB basic modal logic benchmark formulae [12]. The parameter values refer to different characteristics of the formulae being generated (e.g. for the k_branch_n family, the parameter $n \in \mathbb{N}$ generates formulae of $2n + 3$ variables and modal depth $n + 1$; for the k_d4_p family, the parameter refers to the maximum nesting of modal operators; please, refer to [12] for details about the use of parameter values in the construction of formulae for all the families). For each class we have chosen 56 parameter values and corresponding formulae so that only the best current provers, if any at all, can solve every formula in a class within a time limit of 1000 CPU seconds. In total, the collection consists of 1008 formulae of which half are satisfiable and half are unsatisfiable by construction. The minimum modal depth of formulae in this collection is 1, the maximum 30,004, average 1,065.7 with a standard deviation of 2,670.1.

These benchmark formulae have previously been used in [41, 43, 44]. We have excluded a third collection of benchmark formulae that was used in [41, 43, 44], the randomly generated 3CNF$_K$ formulae, as they are of low modal depth and therefore the layered normal form and layered resolution calculus offer little benefit for them.

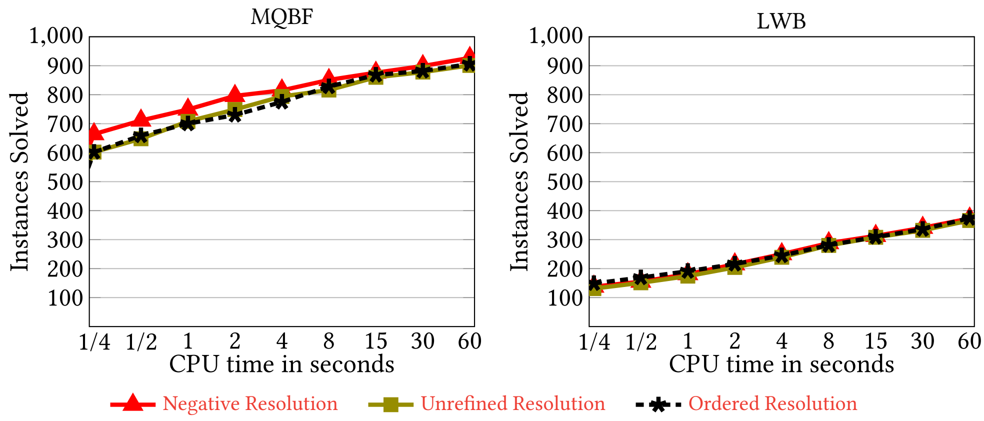

Fig. 1. Influence of Refinements on Performance of Resolution

Benchmarking was performed on PCs with an Intel i7-2600 CPU @3.40GHz and 16GB main memory. For each formula and option of the prover we have determined the median run time over five runs with a time limit of 60 CPU seconds for each run.

Figure 1 compares the impact of different refinements of resolution on the performance of K$_S$P on the MQBF and LWB collections. It is important to remember that the refinements require different normal forms. For unrefined resolution, that is, the calculus given by the rules in Table 3, we use SNF_{ml} clauses. For *negative resolution* and *ordered resolution* we use SNF^{+}_{ml} and SNF^{++}_{ml} clauses, respectively.

Irrespective of the refinement, the shortest clause is selected to perform inferences; both forward and backward subsumption are used, i.e. newly generated or old clauses which are subsumed are deleted; prenex, that is, moving the modal operators outward the formula as far as possible, is set; and no simplification steps are applied.

For the MQBF collection, negative resolution performs better than both ordered and unrefined resolution. Somehow surprisingly, unrefined resolution performs almost as well as ordered resolution for this collection of clauses: whilst ordered resolution produces an output for 906 formulae, the unrefined resolution produces an output for 901 formulae within the same timeout. A possible reason for the good performance of the unrefined resolution is that for the problems in the MQBF benchmark there are few propositional symbols within each modal level, thus the restrictions on the propositional part of the calculus do not have a huge impact on the overall performance of the prover. Also, the normal forms required for applying negative and ordered resolution introduce

	Unrefined Resolution		Negative Resolution		Ordered Resolution	
	#Solved	Parameter	#Solved	Parameter	#Solved	Parameter
k_branch_n	2	2	4	4	12	12
k_poly_n	10	20	11	40	18	180
k_poly_p	11	40	11	40	19	200
k_t4p_n	31	2200	31	2200	19	1000
k_t4p_p	50	9000	50	9000	39	3000

Table 4. Performance of Refinements on Selected LWB Benchmark Classes

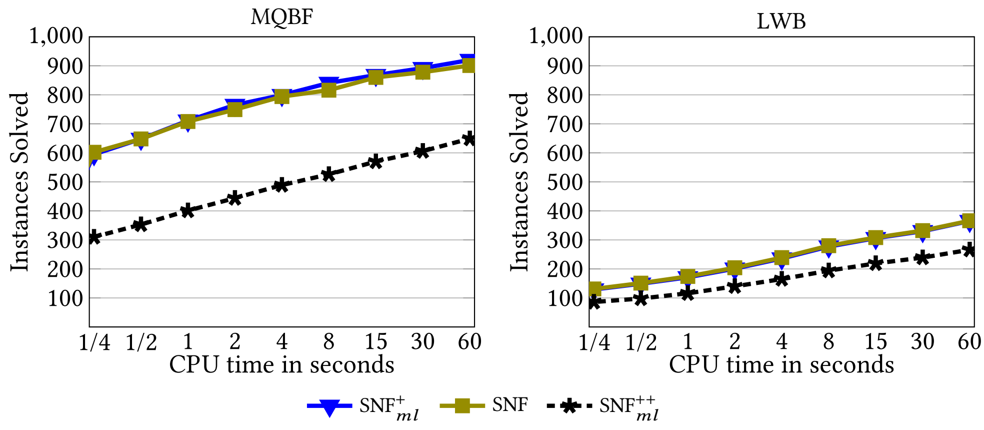

Fig. 2. Influence of Normal Form on Performance of Unrefined Resolution

new propositional symbols, which contribute to the extra time spent on the propositional portion of the problems.

For the LWB collection, there is no discernible difference between unrefined, negative and ordered resolution as far as the total number of solved instances is concerned. However, on the level of individual classes within the collection there are significant differences. Table 4 shows the results for 5 of the 18 classes of the collection, k_poly_p and k_t4p_p contain only provable formulae, k_branch_n, k_poly_n and k_t4p_n only non-provable formulae. For each refinement, the first column in Table 4 shows the number of instances solved and the second column shows the highest value of the parameter solved within the given timeout. On k_t4p_n and k_t4p_p, unrefined and negative resolution perform considerably better than ordered resolution while for the remaining three classes the opposite is true. The poor performance of ordered resolution on k_t4p_n and k_t4p_p is linked to the extra clauses in SNF^{++}_{ml} compared to SNF^{+}_{ml}.

Figure 2 shows the influence of different normal forms on unrefined resolution for the MQBF and LWB collections. There is very little to no difference between the performance on SNF_{ml} and on SNF^{+}_{ml} while performance on SNF^{++}_{ml} is significantly worse. The good performance of unrefined resolution together with SNF^{+}_{ml} on the MQBF collection is not surprising: because of the structure of the problems in this collection, there are not many negated propositional symbols to be renamed in the scope of the modal operators. Considering the 1016 formulae in the MQBF collection, the average number of propositional symbols is 22. After translation into SNF_{ml}, the average number of propositional symbols per problem is 4605. With SNF^{+}_{ml}, this number increases slightly to 4661 propositional symbols. The increase in the average number of clauses is similarly small, from 6410 clauses per problem to 6426 clauses. So, there is no significant difference between the two normal forms on MQBF formulae and therefore no significant difference in the performance of unrefined resolution. However, with SNF^{++}_{ml}, the number of propositional symbols almost doubles, there are 9016 propositional symbols on average per problem, the number of clauses almost doubles to 10781 clauses per problem, which explains the bad performance of ordered resolution.

A similar pattern can also be observed for the LWB benchmark formulae. On the 398 formulae that K_SP can solve within the time limit, the average number of propositional symbols per problem in SNF_{ml}, SNF^{+}_{ml} and SNF^{++}_{ml} is 7496, 8189, and 13377, respectively; only a 9% increase from SNF_{ml} to SNF^{+}_{ml} but a 78% increase from SNF_{ml} to SNF^{++}_{ml}. The average number of clauses per problem in SNF_{ml}, SNF^{+}_{ml} and SNF^{++}_{ml} is 10633, 11309, and 16498; a 6% increase from SNF_{ml} to SNF^{+}_{ml} and a

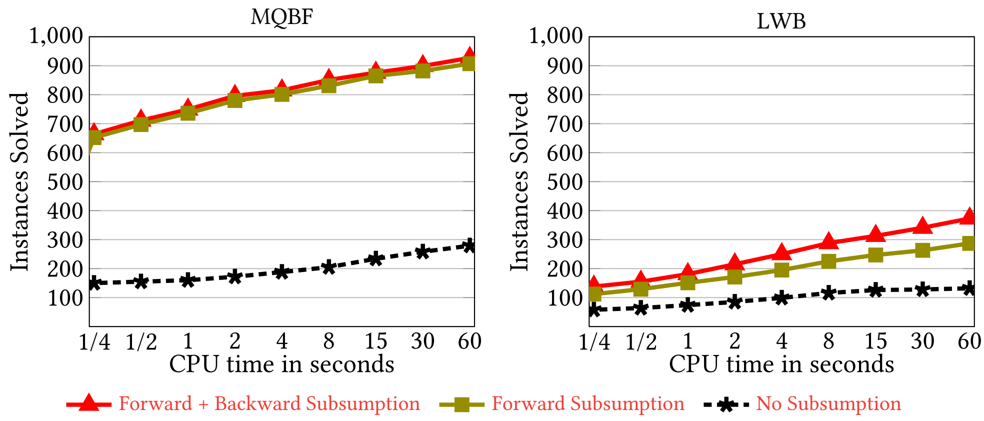

Fig. 3. Influence of Subsumption on Performance of Negative Resolution

55% increase from SNF_{ml} to SNF^{++}_{ml}. This results in a small difference in performance of unrefined resolution on SNF_{ml} clause sets versus SNF^{+}_{ml} clause sets but significant difference between these two normal forms and SNF^{++}_{ml}.

K$_S$P implements both forward subsumption and backward subsumption. With forward subsumption enabled no newly derived literal clause that is subsumed by an existing literal clause will be added to the clause set. With backward subsumption enabled, any newly derived literal clause will be used to remove existing literal clauses subsumed by it. In both cases, subsumption is applied in lazy mode: a clause is tested for subsumption only when it is chosen as a candidate for the resolution and only against clauses in the usable, that is, the set of processed clauses. As pointed out in [59], lazy subsumption avoids expensive checks for clauses that might never be selected during the search of a proof.

Figure 3 shows the impact of forward and backward subsumption on negative resolution. It shows that forward subsumption improves performance significantly. On the MQBF collection, enabling backward subsumption in addition to forward subsumption only has a marginal positive effect: Only a further 20 formulae (2.2%) can be solved within the time limit. On the LWB collection, backward subsumption has a more beneficial effect. Here, an additional 86 formulae (30.0%) can be solved within the time limit.

8 RELATED WORK & DISCUSSION

In [2], a translation-based method for K_1 is given. Formulae are translated into the Guarded Fragment of First-Order logic [1] taking into consideration the modal level where propositional symbols and modal operators occur. The main difference from the usual translation is that relational symbols and propositional symbols are indexed by their modal level at the target language. For instance, $\Diamond(p \Rightarrow \Box\neg p)$ is firstly translated into $\Diamond_1(p_1 \Rightarrow \Box_2\neg p_2)$ and, via standard translation, the formula $\exists y((R_1xy \land P_1) \Rightarrow \forall z(R_2yz \Rightarrow \neg P_2))$ is obtained. Thus, the application of resolution to p and $\neg p$ at the original formula can be avoided in the target language as their translation is not unifiable. The presented translation is suitable for local reasoning. However, for global reasoning, some mechanism for identifying different symbols that are used to translate the same propositional symbol in the original formula would be needed.

The proof method for K_1 given in [5] is also based on translation, but into Hybrid logics (see [15], for a survey on Hybrid logics). Formulae are labelled by either constants a, which correspond to names of worlds in a model, or by pairs (a, b) representing the relation between two worlds named by a and b, respectively. The formula $\Diamond(p \Rightarrow \Diamond\neg p)$ can then be translated into the set of clauses $\{R(a, b), b : \neg p \vee R(b, c), c : \neg p\}$, where a, b, c are constants and R is a new relational symbol. Although this translation does not take into consideration the modal levels where the formulae occur, it avoids applying the resolution rules to occurrences of p at different modal levels as their labels cannot be unified. The presented calculus is also only suitable for local reasoning. Refinements based on selection functions and ordering for the labelled resolution calculus for Hybrid logics are given in [3].

The method most closely related to ours is a combination of the optimised functional translation of K_n to basic path logic [28, 57], which improves and extends previous approaches for translating modal problems into first-order ones [16, 46–48]. Basic path logic is a fragment of sorted first-order logic with a sort S_W for the set of worlds W, a sort S_a for each index $a \in A_n$, binary functions $[\_\_]_a$, $a \in A_n$, of sort $S_W \times S_a \rightarrow S_W$, special unary predicates def_a, $a \in A_n$, of sort S_W representing subsets of W, and unary predicate symbols P_p of sort S_W for each propositional variable $p \in P$. The formula $\Diamond\!\!\!\!{\scriptstyle 1}\,(p \wedge \boxed{1}\neg p)$ is translated to $def_1(x : S_W) \wedge (P_p([x : S_W, y : S_1]) \wedge \forall z : S_1(def_1([x : S_W, y : S_1]) \Rightarrow \neg P_p([[x : S_W, y : S_1], z : S_1])))$ from which the set of clauses $\{def_1(a : S_W), P_p([a : S_W, b : S_1]), \neg def_1([a : S_W, b : S_1]) \vee \neg P_p([[a : S_W, b : S_1], z : S_1])\}$ can be obtained. No inferences by resolution are possible from this set of clauses as, for instance, $P_p([x : S_W, y : S_1])$ and $P_p([[x : S_W, y : S_1], z : S_1])$ do not unify. If one only considers the local satisfiability problem, then it is possible to further improve this translation. In the polyadic optimised functional translation the combination of unary predicate symbols and nested function applications of depth is replaced by n-ary predicate symbols and sorts are encoded into the predicate symbols. The formula $\Diamond\!\!\!\!{\scriptstyle 1}\,(p \wedge \boxed{1}\neg p)$ is then translated to $def_{1,W}(x) \wedge (P_{p,W,1}(x, y) \wedge \forall z(def_{1,W,1}(x, y) \Rightarrow \neg P_{p,W,1,1}(x, y, z)))$ with corresponding set of clauses $\{def_{1,W}(a), P_{p,W,1}(a, b), \neg def_{1,W,1}(a, b) \vee \neg P_{p,W,1,1}(a, b, z)\}$. Again, no resolution inferences are possible from this set of clauses.

The work presented here extends the calculus given in [39], which could only deal with local reasoning. Moreover, two of the inference rules given in [39], namely IRES1 and IRES2, are not required for completeness. Both calculi have been implemented as part of the proof search procedures of K_SP [41, 43]. For local reasoning, the experimental results given in [41] show that the labelling of clauses is effective in avoiding unnecessary applications of inference rules, thus improving performance. From a theoretical point of view, the improvement in efficiency can be explained by the following facts. The resolution procedure for classical logic runs in deterministic exponential time in the size of the number of literals occurring in the clause set [55]. The modal inference rules for the calculus given here do not change the overall complexity, as they can be seen as variations of the classical inference resolution rule. Thus, if m is the number of literals occurring in the clause set, the implementation of the saturation procedure is bound to run in time $T(m) = O(2^m)$. However, if we consider that the modal problem has modal depth $ml > 1$ and that the propositional symbols are uniformly distributed over the modal levels, then the expectation of the number of propositions by modal level is given by m/ml. Thus, the recurrence equation for the running time of the saturation procedure is given by $T(m) = ml \times T(2^{m/ml})$, whose solution is in $O(\sqrt[ml]{2^m})$. It is then clear that the saturation procedure based on levels is asymptotically better than the previous saturation procedure.

In [41] we have also compared the performance of K_SP with state-of-the art provers, namely BDDTab [20], FaCT++ 1.6.3 [65], InKreSAT 1.0 [30], Spartacus 1.0 [21], and a combination of the optimised functional translation [28] with Vampire 3.0 [31]. Besides the MQBF [30, 36] and the LWB collections presented above, one more collection of modal formulae was used in the comparison: the randomly generated $3\mathrm{CNF}_{\mathsf{K}}$ formulae [50] over 3 to 10 propositional symbols with modal depth 1 or 2. When considering the three different benchmarks, K$_S$P performs best on the modalised random QBF formulae. In contrast, K$_S$P performs worse on the randomly generated 3CNF$_K$ formulae as the labelling offers very little advantage for modal formula with a maximal modal depth of 2. Finally, for the LWB benchmark classes, performance very much depends on the characteristics of the individual class, without a clear picture emerging. Overall, the experimental results given in [41] suggest that K$_S$P performs best if propositional variables are evenly spread across a wide range of modal levels. It is worth noting that for the LWB benchmark, ordered resolution outperforms the other implemented refinements. The implementation of the different strategies allows for the user to choose the refinement which is more suitable for the structure of the problem in hand.

9 CONCLUSION

We have presented a novel resolution calculus for K_n that restricts resolution inferences to formulae at the same modal level. The experimental evaluation shows that negative resolution performs better than both ordered resolution and resolution with no restrictions, for the benchmark considered here. The paper provides full proofs that the calculus and its refinements are sound, complete and terminating. Moreover, we show that the calculus remains complete if subsumption is applied. The evaluation shows that applying forward subsumption has a positive impact on the performance of the prover. The evaluation also shows that the gains from the use of ordered resolution are counteracted by the requirement to use a clausal normal form that often contains considerably more clauses than the normal forms we can use with unrefined or negative resolution. In future work we will investigate how this relative increase in size can be minimised.

In the calculus presented here, we have restricted the labelling to modal levels, which makes the unification procedure required for the application of the inference rules very simple. We conjecture that having more structured labels might help achieving more efficient proof search procedures in some cases. For instance, for formulae with a large number of occurrences of the $\langle a \rangle$ operator by modal level, $a \in A_n$, that is, formulae with high *modal branching*, it might help to use the labels to refer to worlds as well. This kind of labelling would mimic labelled tableaux procedures, as that of [18], and possibly minimise the number of inference rules to be applied during the saturation of satisfiable sets of propositional clauses. On the other hand, if both local and global reasoning are applied to such formulae, the labelling of worlds might lead to repeated application of rules to the same set of clauses, as it occurs with propagation of formulae in the scope of $\boxed{a}$ operators in tableaux-based procedures. Another possibility is to label formulae with their modal path, as in the optimised functional translation [28, 57], which makes the unification procedure slightly more complicated, but could help to achieve more efficiency in the case of formulae with high modal depth, but low modal branching. Those and the investigation in the application to other strategies are subject of future work.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their detailed comments that helped improve the paper. C. Dixon was partially supported by the Engineering and Physical Sciences Research Council (EPSRC) funded RAI Hubs FAIR-SPACE (EP/R026092/1) and RAIN (EP/R026084/1), and the EPSRC funded programme grant S4 (EP/N007565/1).

REFERENCES

[1] Hajnal Andréka, Johan van Benthem, and Istvan Németi. 1998. Modal logics and bounded fragments of predicate logic. *Journal of Philosophical Logic* 27, 3 (1998), 217–274.

[2] Carlos Areces, Rosella Gennari, Juan Heguiabehere, and Maarten de Rijke. 2000. Tree-Based Heuristics in Modal Theorem Proving. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2000)*. IOS Press, 199–203.

[3] Carlos Areces and Daniel Gorín. 2011. Resolution with Order and Selection for Hybrid Logics. *Journal of Automated Reasoning* 46, 1 (2011), 1–42.

[4] Carlos Areces and Juan Heguiabehere. 2002. HyLoRes: A Hybrid Logic Prover. (Sept. 18 2002).

[5] Carlos Areces, Hans de Nivelle, and Maarten de Rijke. 1999. Prefixed Resolution: A Resolution Method for Modal and Description Logics. In *Proceedings of the 16th International Conference on Automated Deduction (CADE-16) (Lecture Notes in Artificial Intelligence)*, Vol. 1632. Springer, 187–201.

[6] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press.

[7] Matthias Baaz, Uwe Egly, and Alexander Leitsch. 2001. Normal Form Transformations. In *Handbook of Automated Reasoning*, Alan Robinson and Andrei Voronkov (Eds.). Vol. I. Elsevier and MIT Press, 273–333.

[8] Matthias Baaz and Christian G. Fermüller. 1992. Resolution for Many-Valued Logics. In *Proceedings of the International Conference on Logic Programming and Automated Reasoning (LPAR 1992) (Lecture Notes in Computer Science)*, Andre Voronkov (Ed.), Vol. 624. Springer, 107–118.

[9] Matthias Baaz, Christian G. Fermüller, and Gernot Salzer. 2001. Automated Deduction for Many-Valued Logics. In *Handbook of Automated Reasoning*, Alan Robinson and Andrei Voronkov (Eds.). Vol. II. Elsevier and MIT Press, Chapter 20, 1355–1402.

[10] Leo Bachmair and Harald Ganzinger. 1990. On restrictions of ordered paramodulation with simplification. In *Proceedings of the 10th International Conference on Automated Deduction (CADE-10) (Lecture Notes in Computer Science)*, Vol. 449. Springer, 427–441.

[11] Philippe Balbiani and Stéphane Demri. 1997. Prefixed Tableaux Systems for Modal Logics with Enriched Languages. In *Proceedings of the 15th International Joint Conference on Artificiall Intelligence (JCAI 1997)*. Morgan Kaufmann, 190–195.

[12] Peter Balsiger, Alain Heuerding, and Stefan Schwendimann. 2000. A Benchmark Method for the Propositional Modal Logics K, KT, S4. *Journal of Automated Reasoning* 24, 3 (2000), 297–317.

[13] David Basin, Sean Matthews, and Luca Vigano. 1997. Labelled Propositional Modal Logics: Theory and Practice. *Journal of Logic and Computation* 7, 6 (1997), 685–717.

[14] Patrick Blackburn, Maarten de Rijke, and Yde Venema. 2001. *Modal Logic.* Cambridge University Press.

[15] Torben Braüner. 2014. Hybrid Logic. In *Handbook of Philosophical Logic: Volume 17*, Dov M. Gabbay and Franz Guenthner (Eds.). Springer, 1–77.

[16] Luis Fariñas del Cerro and Andreas Herzig. 1988. Linear Modal Deductions. In *Proceedings of the 9th International Conference on Automated Deduction (CADE-9) (Lecture Notes in Computer Science)*, Vol. 310. Springer, 487–499.

[17] Melvin Fitting. 2012. Prefixed tableaus and nested sequents. *Annals of Pure and Applied Logic* 163, 3 (2012), 291–313.

[18] Melvin Fitting and Richard L. Mendelsohn. 1998. *First-Order Modal Logic.* Kluwer.

[19] Valentin Goranko and Solomon Passy. 1992. Using the universal modality: gains and questions. *Journal of Logic and Computation* 2, 1 (1992), 5–30.

[20] Rajeev Goré, Kerry Olesen, and Jimmy Thomson. 2014. Implementing Tableau Calculi Using BDDs: BDDTab System Description. In *Proceedings of the 7th International Joint Conference on Automated Deduction (IJCAR 2014) (Lecture Notes in Computer Science)*, Vol. 8562. Springer, 337–343.

[21] Daniel Götzmann, Mark Kaminski, and Gert Smolka. 2010. Spartacus: A Tableau Prover for Hybrid Logic. *Electronic Notes in Theoretical Computer Science* 262 (2010), 127–139.

[22] Reiner Hähnle. 1993. *Automated Deduction in Multiple-Valued Logics.* Oxford University Press.

[23] Brent T. Hailpern. 1982. *Verifying Concurrent Processes Using Temporal Logic.* Lecture Notes in Computer Science, Vol. 129. Springer.

[24] Joseph Y. Halpern. 1987. Using Reasoning about Knowledge to Analyze Distributed Systems. *Annual Review of Computer Science* 2 (1987), 37–68.

[25] Joseph Y. Halpern, Zohar Manna, and Ben Moszkowski. 1983. A Hardware Semantics Based on Temporal Intervals, In Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP 1983). *Lecture Notes in Computer Science* 154, 278–291.

[26] Joseph Y. Halpern and Yoram Moses. 1992. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence* 54, 3 (April 1992), 319–379.

[27] Patrick J. Hayes and Robert A. Kowalski. 1969. Semantic Trees in Automatic Theorem Proving. In *Proceedings of the Fourth Annual Machine Intelligence Workshop 1968.* Elsevier, 87–101.

[28] Ian Horrocks, Ullrich Hustadt, Ulrike Sattler, and Renate A. Schmidt. 2006. Computational Modal Logic. In *Handbook of Modal Logic*, Patrick Blackburn, Johan van Benthem, and Frank Wolter (Eds.). Elsevier, 181–245.

[29] Ullrich Hustadt and Boris Konev. 2004. TRP++: A temporal resolution prover. In *Collegium Logicum*. Kurt Gödel Society, 65–79.

[30] Mark Kaminski and Tobias Tebbi. 2013. InKreSAT: Modal Reasoning via Incremental Reduction to SAT. In *Proceedings of the 24th International Conference on Automated Deduction (CADE-24) (Lecture Notes in Artificial Intelligence)*, Vol. 7898. Springer, 436–442.

[31] Laura Kovács and Andrei Voronkov. 2013. First-Order Theorem Proving and Vampire. In *Proceedings of the 25th International Conference on Computer Aided Verification (CAV 2013) (Lecture Notes in Computer Science)*, Vol. 8044. Springer, 1–35.

[32] Richard E. Ladner. 1977. The Computational Complexity of Provability in Systems of Modal Propositional Logic. *SIAM Journal on Computing* 6, 3 (1977), 467–480.

[33] Richard C. T. Lee. 1967. *A completeness theorem and computer program for finding theorems derivable from given axioms*. Ph.D. Dissertation. Berkeley.

[34] Maarten Marx. 2006. Complexity of Modal Logic. In *Handbook of Modal Logic*, Patrick Blackburn, Johan van Benthem, and Frank Wolter (Eds.). Elsevier, New York, 139–179.

[35] Maarten Marx and Yde Venema. 2007. Local Variations on a Loose Theme: Modal Logic and Decidability. In *Finite Model Theory and Its Applications*. Springer, 371–429.

[36] Fabio Massacci and Francesco M. Donini. 2000. Design and Results of TANCS-2000 Non-classical (Modal) Systems Comparison. In *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2000) (Lecture Notes in Computer Science)*, Vol. 1847. Springer, 52–56.

[37] William W. McCune. 2007. OTTER 3.0 reference manual and guide. (May 07 2007).

[38] Eliana Minicozzi and Raymond Reiter. 1972. A note on Linear Resolution Strategies in Consequence-Finding. *Artificial Intelligence* 3, 1–3 (1972), 175–180.

[39] Cláudia Nalon and Clare Dixon. 2007. Clausal resolution for normal modal logics. *Journal of Algorithms* 62 (2007), 117–134. Issue 3-4.

[40] Cláudia Nalon, Ullrich Hustadt, and Clare Dixon. 2015. A Modal-Layered Resolution Calculus for K. In *Proceedings of the 24th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2015) (Lecture Notes in Computer Science)*, Vol. 9323. Springer, 185–200.

[41] Cláudia Nalon, Ullrich Hustadt, and Clare Dixon. 2016. K_SP: A Resolution-Based Prover for Multimodal K. In *Proceedings of the 8th International Joint Conference on Automated Reasoning (IJCAR 2016) (Lecture Notes in Computer Science)*, Vol. 9706. Springer, 406–415.

[42] Cláudia Nalon, Ullrich Hustadt, and Clare Dixon. 2016. K_SP: sources and benchmarks. http://www.cic.unb.br/~nalon/#software. (2016).

[43] Cláudia Nalon, Ullrich Hustadt, and Clare Dixon. 2017. K_SP: A Resolution-based Prover for Multimodal K, Abridged Report. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017)*. ijcai.org, 4919–4923.

[44] Cláudia Nalon, Ullrich Hustadt, and Clare Dixon. 2018. K_SP A Resolution-Based Theorem Prover for K_n: Architecture, Refinements, Strategies and Experiments. *Journal of Automated Reasoning* (Online First: 17 Dec 2018). https://doi.org/10.1007/s10817-018-09503-x

[45] Andreas Nonnengart and Christoph Weidenbach. 2001. Computing Small Clause Normal Forms. In *Handbook of Automated Reasoning*, Alan Robinson and Andrei Voronkov (Eds.). Vol. 1. Elsevier and MIT Press, 335–367.

[46] Hans Jürgen Ohlbach. 1988. A Resolution Calculus for Modal Logics. In *Proceedings of the 9th International Conference on Automated Deduction (CADE-9) (Lecture Notes in Computer Science)*, Vol. 310. Springer, 500–516.

[47] Hans Jürgen Ohlbach. 1991. Semantics-Based Translation Methods for Modal Logics. *Journal of Logic and Computation* 1, 5 (1991), 691–746.

[48] Hans Jürgen Ohlbach and Renate A. Schmidt. 1997. Functional Translation and Second-Order Frame Properties of Modal Logics. *Journal of Logic and Computation* 7, 5 (Oct. 1997), 581–603.

[49] Guoqiang Pan and Moshe Y. Vardi. 2003. Optimizing a BDD-Based Modal Solver. In *Proceedings of the 19th International Conference on Automated Deduction (CADE-19) (Lecture Notes in Computer Science)*, Vol. 2741. Springer, 75–89.

[50] Peter F. Patel-Schneider and Roberto Sebastiani. 2003. A New General Method to Generate Random Modal Formulae for Testing Decision Procedures. *Journal of Artificial Intelligence Research (JAIR)* 18 (2003), 351–389.

[51] David A. Plaisted and Steven A. Greenbaum. 1986. A Structure-Preserving Clause Form Translation. *Journal of Logic and Computation* 2 (1986), 293–304.

[52] Vaughan R. Pratt. 1980. Application of Modal Logic to Programming. *Studia Logica* 39, 2/3 (1980), 257–274.

[53] Anand S. Rao and Michael P. Georgeff. 1991. Modeling Rational Agents within a BDI-Architecture. In *Proceedings of KR&R-91*. Morgan-Kaufmann, 473–484.

[54] John Alan Robinson. 1965. Automatic Deduction with Hyper-resolution. *International Journal of Computer Mathematics* 1 (1965), 227–234.

[55] John Alan Robinson. 1965. A Machine–Oriented Logic Based on the Resolution Principle. *Journal of the ACM* 12, 1 (Jan. 1965), 23–41.

[56] Klaus Schild. 1991. A Correspondence Theory for Terminological Logics. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI 1991)*. Morgan Kaufmann, 466–471.

[57] Renate A. Schmidt. 1999. Decidability by Resolution for Propositional Modal Logics. *Journal of Automated Reasoning* 22, 4 (1999), 379–396.

[58] Stephan Schulz. 2013. The E Theorem Prover. (2013). http://wwwlehre.dhbw-stuttgart.de/~sschulz/E/E.html.

[59] Stephan Schulz. 2013. Simple and Efficient Clause Subsumption with Feature Vector Indexing. In *Automated Reasoning and Mathematics — Essays in Memory of William W. McCune (Lecture Notes in Computer Science)*, Maria Paola Bonacina and Mark E. Stickel (Eds.), Vol. 7788. Springer, 45–67.

[60] Frantisek Simancik, Boris Motik, and Ian Horrocks. 2014. Consequence-based and fixed-parameter tractable reasoning in description logics. *Artificial Intelligence* 209 (2014), 29–77.

[61] James R. Slagle. 1967. Automatic Theorem Proving With Renamable and Semantic Resolution. *Journal of the ACM* 14, 4 (Oct. 1967), 687–697.

[62] James R. Slagle, C. L. Chang, and Richard C. T. Lee. 1969. Completeness Theorems for Semantic Resolution in Consequence-Finding. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence (IJCAI 1969)*. William Kaufmann, 281–286.

[63] Edith Spaan. 1993. *Complexity of Modal Logics*. Ph.D. Dissertation. University of Amsterdam.

[64] The SPASS Team. 2010. Automation of Logic: SPASS. (2010). http://www.spass-prover.org/.

[65] Dmitry Tsarkov and Ian Horrocks. 2006. FaCT++ Description Logic Reasoner: System Description. In *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR 2006) (Lecture Notes in Computer Science)*, Vol. 4130. Springer, 292–297.

[66] Andrei Voronkov. 2013. Vampire. (2013). http://www.vprover.org/index.cgi.

[67] Arild Waaler. 2001. Connections in Nonclassical Logics. In *Handbook of Automated Reasoning*, Alan Robinson and Andrei Voronkov (Eds.). Elsevier and MIT Press, 1487–1578.

[68] Lincoln A. Wallen. 1990. *Automated Deduction in Non-Classical Logics*. MIT Press.

[69] Larry Wos, George A. Robinson, and Daniel Carson. 1965. Efficiency and Completeness of the Set of Support Strategy in Theorem Proving. *Journal of the ACM* 12 (1965), 536–541.

Received 2017