

A Resolution-Based Prover for Normal Modal Logics

Cláudia Nalon¹

George Bezerra Silva¹

Department of Computer Science

University of Brasília, Brazil

nalon@unb.br, georgelione@gmail.com

Abstract: We present a prototype tool for automated reasoning for multimodal normal logics where combinations of the axioms **K**, **T**, **D**, **B**, **4**, and **5** hold. The theorem prover is based on previous work on resolution calculi for such logics. We briefly present the syntax, the semantics, and the calculus for the basic normal logic together with the inference rules for dealing with each specific axiom. We then give details of the implementation of the prover and discuss future work.

1 Introduction

In [1], sound, complete, and terminating resolution-based methods for fifteen families of propositional normal modal logics are presented. These calculi deal with multimodal logics K_n , in which the schemata $\Box(\varphi \Rightarrow \psi) \Rightarrow (\Box\varphi \Rightarrow \Box\psi)$ (**K_a**), where φ and ψ are well-formed formulae, are valid, and the extensions of K_n where the (combination of the) following axioms for reflexive (**T_a**: $\Box\varphi \Rightarrow \varphi$), serial (**D_a**: $\Box\varphi \Rightarrow \Diamond\varphi$), symmetric (**B_a**: $\Diamond\Box\varphi \Rightarrow \varphi$), transitive (**4_a**: $\Box\varphi \Rightarrow \Box\Box\varphi$), and Euclidean systems (**5_a**: $\Diamond\Box\varphi \Rightarrow \Box\Diamond\varphi$) also hold.

Here, we present a prototype theorem-prover for those families of logics using the methods given in [1]. In the next section, we present the language of K_n . In Section 3, we introduce the resolution-based method for K_n and the rules for dealing with the axioms given above. In Section 4, we introduce the theorem-prover for K_n , giving details of its implementation. Conclusions are given in Section 5.

2 The Normal Logic K_n

Formulae in K_n are constructed from a denumerable set of *propositional symbols*, $\mathcal{P} = \{p, q, p', q', p_1, q_1, \dots\}$. Besides classical connectives ($\neg, \wedge$), a set of unary modal operators $\Box, a \in \mathcal{A}$, is introduced, where $\Box\varphi$ is read as “the agent a considers φ necessary” and $\mathcal{A} = \{1, \dots, n\}$ is the set of agents.. The operator $\Diamond\varphi$ is an abbreviation for $\neg\Box\neg\varphi$. The set of well-formed formulae, **WFF**, is defined in the usual way: $p \in \mathcal{P}$ is in **WFF**; **true** is in **WFF**; if φ and ψ are in **WFF**, then so are $\neg\varphi$, $(\varphi \wedge \psi)$, and $\Box\varphi$, for all $a \in \mathcal{A}$. A *literal* is either p or $\neg p$, for $p \in \mathcal{P}$. $\mathcal{L}$ is the set of literals. A *modal literal* is either $\Box l$ or $\neg\Box l$, $l \in \mathcal{L}$.

A *Kripke structure* M over $\mathcal{P}$ is a tuple $M = \langle \mathcal{S}, \pi, \mathcal{R}_1, \dots, \mathcal{R}_n \rangle$, where $\mathcal{S}$ is a set of possible *worlds* (or *states*) with a distinguished world s_0 ; the function $\pi(s) : \mathcal{P} \rightarrow \{\text{true}, \text{false}\}$, $s \in \mathcal{S}$, is an interpretation that associates with each state in $\mathcal{S}$ a truth assignment to propositions; and each $\mathcal{R}_a \subseteq \mathcal{S} \times \mathcal{S}$ is a binary relation on $\mathcal{S}$, where $a \in \mathcal{A}$.

We write $(M, s) \models \varphi$ to say that φ is true at world s in the Kripke structure M . Truth of classical formulae is given as usual; for modal formulae, we have that $(M, s) \models$

$\Box\varphi$ iff for all t , such that $(s, t) \in \mathcal{R}_a$, $(M, t) \models \varphi$. The formulae **false**, $(\varphi \vee \psi)$, and $(\varphi \Rightarrow \psi)$ are introduced as the usual abbreviations for $\neg\text{true}$, $\neg(\neg\varphi \wedge \neg\psi)$, and $(\neg\varphi \vee \psi)$, respectively. Formulae are interpreted with respect to the distinguished world s_0 . Let $M = \langle \mathcal{S}, \pi, \mathcal{R}_1, \dots, \mathcal{R}_n \rangle$ be a Kripke structure with a distinguished world s_0 . A formula φ is said to be *satisfiable* in M if $(M, s_0) \models \varphi$; φ is said to be *satisfiable* if there is a model M such that $(M, s_0) \models \varphi$; and φ is said to be *valid* if for all models M , $(M, s_0) \models \varphi$.

3 The Resolution Method for K

The resolution method is applied to formulae in the Separated Normal Form for Normal Logics (**SNF_K**). A nullary connective **start** is introduced to deal with reasoning within the distinguished world s_0 ; formally, $(M, s) \models \text{start}$ iff $s = s_0$. A formula in **SNF_K** is a conjunction of clauses, in one of the following forms: *initial* (**start** $\Rightarrow \bigvee_{b=1}^r l_b$), *literal* (**true** $\Rightarrow \bigvee_{b=1}^r l_b$), *positive modal* ($l' \Rightarrow \Box l$), or *negative modal* ($l' \Rightarrow \neg\Box l$), where $l, l', l_b \in \mathcal{L}$. Transformation rules and their correctness can be found in [1].

Once a formula has been transformed in its normal form, the inference rules are applied to the set of clauses until either a contradiction is found (in the form of **true** $\Rightarrow$ **false** or **start** $\Rightarrow$ **false**) or no new clauses can be generated. If a contradiction is found when the method is applied to a set of clauses $\mathcal{S}$, we say that there is a *refutation* for $\mathcal{S}$. The inference rules are shown in Figure 1, where $l, l', l_i, l'_i \in \mathcal{L}$ ($i \in \mathbb{N}$) and D, D' are disjunctions of literals. Figure 2 shows the inference rules for dealing with the satisfiability problem in systems where reflexivity ([REF]), seriality ([SER]), symmetry ([SYM]), transitivity ([TRANS]), and Euclideaness ([EUC1] and [EUC2]) hold. The resolvents of the inference rules [TRANS], [EUC1], and [EUC2] introduce literals of the form $\text{pos}_{a,l}$ and $\text{nec}_{a,l}$, which are used to keep the normal form of the resolvents by renaming the formulae $\neg\Box\neg l$ and $\Box l$, respectively,

4 The Prover

The theorem prover for K_n together with rules to deal with systems where the axioms **K**, **T**, **D**, **B**, **4**, and **5** hold can be found in [2]. The prover, which has been implemented

$\begin{array}{l} \text{[IRES1]} \quad \frac{\text{true} \Rightarrow D \vee l}{\text{start} \Rightarrow D' \vee \neg l} \\ \text{start} \Rightarrow D \vee D' \end{array}$			$\begin{array}{l} \text{[IRES2]} \quad \frac{\text{start} \Rightarrow D \vee l}{\text{start} \Rightarrow D' \vee \neg l} \\ \text{start} \Rightarrow D \vee D' \end{array}$		
$\begin{array}{l} \text{[LRES]} \quad \frac{\text{true} \Rightarrow D \vee l}{\text{true} \Rightarrow D' \vee \neg l} \\ \text{true} \Rightarrow D \vee D' \end{array}$			$\begin{array}{l} \text{[MRES]} \quad \frac{l_1 \Rightarrow \Box l}{l_2 \Rightarrow \neg \Box l} \\ \text{true} \Rightarrow \neg l_1 \vee \neg l_2 \end{array}$		
$\begin{array}{l} \text{[GEN1]} \quad \frac{l'_1 \Rightarrow \Box \neg l_1}{\vdots} \\ \frac{l'_m \Rightarrow \Box \neg l_m}{l' \Rightarrow \neg \Box \neg l} \\ \frac{\text{true} \Rightarrow l_1 \vee \dots \vee l_m \vee \neg l}{\text{true} \Rightarrow \neg l'_1 \vee \dots \vee \neg l'_m \vee \neg l'} \end{array}$			$\begin{array}{l} \text{[GEN2]} \quad \frac{l'_1 \Rightarrow \Box l_1}{l'_2 \Rightarrow \Box \neg l_1} \\ \frac{l'_3 \Rightarrow \neg \Box \neg l_2}{\text{true} \Rightarrow \neg l'_1 \vee \neg l'_2 \vee \neg l'_3} \end{array}$		
			$\begin{array}{l} \text{[GEN3]} \quad \frac{l'_1 \Rightarrow \Box \neg l_1}{\vdots} \\ \frac{l'_m \Rightarrow \Box \neg l_m}{l' \Rightarrow \neg \Box \neg l} \\ \frac{\text{true} \Rightarrow l_1 \vee \dots \vee l_m}{\text{true} \Rightarrow \neg l'_1 \vee \dots \vee \neg l'_m \vee \neg l'} \end{array}$		

Figure 1: Inference Rules for K

$\text{[REF]} \quad \frac{l_1 \Rightarrow \Box l}{\text{true} \Rightarrow \neg l_1 \vee l}$			$\text{[SER]} \quad \frac{l_1 \Rightarrow \Box l}{l_1 \Rightarrow \neg \Box \neg l}$			$\text{[SYM]} \quad \frac{l_1 \Rightarrow \Box \neg l}{l \Rightarrow \Box \neg l_1}$		
$\text{[TRANS]} \quad \frac{l_1 \Rightarrow \Box l}{\text{true} \Rightarrow \neg l_1 \vee \text{neg}_{a,l}} \\ \text{neg}_{a,l} \Rightarrow \Box l \\ \text{neg}_{a,l} \Rightarrow \Box \text{neg}_{a,l}}$			$\text{[EUC1]} \quad \frac{l_1 \Rightarrow \neg \Box \neg l}{\text{true} \Rightarrow \neg l_1 \vee \text{pos}_{a,l}} \\ \text{pos}_{a,l} \Rightarrow \neg \Box \neg l \\ \neg \text{pos}_{a,l} \Rightarrow \Box \neg l \\ \text{pos}_{a,l} \Rightarrow \Box \text{pos}_{a,l}}$			$\text{[EUC2]} \quad \frac{l_1 \Rightarrow \Box l_2}{\text{pos}_{a,l_1} \Rightarrow \Box l_2} \\ \text{pos}_{a,l_1} \Rightarrow \neg \Box \neg l_1 \\ \neg \text{pos}_{a,l_1} \Rightarrow \Box \neg l_1 \\ \text{pos}_{a,l_1} \Rightarrow \Box \text{pos}_{a,l_1}}$		

Figure 2: Inference Rules for Other Systems

in C++, takes as input a file with formulae in the language of K_n (where several different notations are allowed). For each formula, the prover carries out the transformation into the normal form and applies the resolution method, taking into consideration all inference rules specified as arguments at a command line. For instance, with the argument `-kt 4`, the prover applies all the inference rules for K_n and the rules [REF] and [TRANS]. The prover has several levels of verbosity: only the result (satisfiable/unsatisfiable), the proof (or the last attempt of a proof), or all the steps in the attempt of finding a proof.

The prover cycles over the set of clauses applying, firstly, the inference rules [SER], [SYM], [TRANS], [EUC1], and [EUC2], until no new clauses can be found. Note that these rules are applied to modal clauses only, so they can all be applied before the cycle corresponding to the resolution method for $K_{(n)}$, as the resolution rules for this system (i.e. [IRES1], [IRES2], [MRES], [LRES], [GEN1], [GEN2], and [GEN3]) do not generate modal clauses. For the same reason, the rules [MRES] and [GEN2] are also applied before the other rules. The prover for K_n then tries to employ linear resolution to the propositional part of the language as far as possible, but does not backtrack. If the last generated clause cannot be resolved with any other clause, then modal resolution is exhaustively applied and a new cycle of linear resolution over the literal clauses begins. Backtracking only takes place if literal and modal resolution cannot be further applied. Initial resolution consists on checking if the literal, which renames the original formula and is introduced during translation, appears negated in the set

of literal clauses. Forward subsumption is implemented for the set of literal clauses.

5 Conclusion

The theorem-prover for K_n performs well and the implementation of the linear strategy helps the prover to make better use of space. Nevertheless, the implemented prover is not robust enough to deal with large formulae and is currently under revision. The architecture of the prover is also to be changed to allow for an easier inclusion of new inference rules. Moreover, all the features of the present version are hard-coded in the prover. We therefore intend to provide more options for testing some features of the calculi as, for instance, giving priorities for inference rules, better selection of clauses and literals (e.g. by using ordered resolution), and the use of pure set of support instead of the combination with linear resolution. Another desirable feature is to allow for modal operators under different logics (e.g. $\Box$ in KD_n and $\Box$ in $S5_n$).

References

- [1] C. Nalon and C. Dixon. Clausal resolution for normal modal logics. *J. Algorithms*, 62:117–134, July 2007.
- [2] G. B. Silva. Implementação de um provador de teoremas por resolução para lógicas modais normais. Monografia de Conclusão de Curso, Universidade de Brasília, 2013. Prover available at <http://www.cic.unb.br/~nalon/#software>.