On the Combination of Resolution and SAT Procedures for Modal Theorem-Proving

Cláudia Nalon
University of Brasília
Brasília, DF, Brazil
nalon@unb.br

Daniella Angelos
University of Brasília
Brasília, DF, Brazil
angelos@aluno.unb.br

Keywords Modal Logics, Resolution, SAT

Modal logics are extensions of classical logic with a unary operator $\Box$. The operator $\Diamond$ is the dual of $\Box$: given a formula φ, $\Diamond\varphi$ is defined as $\neg\Box\neg\varphi$. The usual reading of $\Box$ (resp. $\Diamond$) is that of *necessity* (resp. *possibility*). Formulae are interpreted over Kripke Structures, that is, a tuple (W, R, π), where W is a non-empty set; $R \subseteq W \times W$ is a binary relation; and $\pi : W \to (P \to \{true, false\})$ is a function which associates to each world $w \in W$ a truth-assignment to propositional symbols in a denumerable set $P = \{p, q, r, \ldots, p_1, q_1, r_1, \ldots\}$.

In recent work [2], we have proposed a resolution-based method for the basic normal multimodal logic K_n, where clauses are annotated with the modal level in which they occur. The calculus was implemented in the form of KSP [3]. The prover performs well if the set of propositional symbols is uniformly distributed over the modal levels. However, when there is a high number of propositional symbols in a particular level, the performance deteriorates. One reason is that the specific normal form we use always generates satisfiable sets of literal clauses (clauses without modal operators). Saturation, i.e. computing the closure of such a set of clauses under the resolution rule, can then be very time consuming. In order to try to ameliorate the performance of KSP, we are currently investigating the use of the combination of the resolution procedure and *Boolean Satisfiability Solvers*, or SAT solvers, for short. SAT solvers can often solve hard structured problems with over a million variables and several million constraints in reasonable time [1].

Our implementation, a work in progress, uses MiniSat [5], a SAT solver based on the well-known Conflict-Driven Clause Learning (CDCL) algorithm [4]. Very briefly, in the attempt of finding an assignment which satisfies the input, a CDCL-based SAT prover analyses the clauses and the partial assignments which have generated a conflict (i.e. a contradiction) by applications of unit resolution. From such analysis, a new clause may be learnt and is added to the clause set, often abbreviating the time spent in the search for further satisfiable assignments. As we already know that the set of clauses we will input to MiniSat is satisfiable, we are not interested in the model provided by the SAT prover, but in the learnt clauses themselves. To illustrate how we make use of the SAT prover, consider the following set of labelled clauses:

1.	$1 : l_1 \Rightarrow \Box\neg p_1$	6.	$2 : p_4 \vee p_5 \vee p_6$
2.	$1 : l_2 \Rightarrow \Box\neg p_2$	7.	$2 : \neg p_6 \vee \neg p_7$
3.	$1 : l_3 \Rightarrow \Diamond\neg p_3$	8.	$2 : p_3 \vee p_8$
4.	$2 : p_1 \vee p_2 \vee \neg p_4$	9.	$2 : p_4 \vee p_7 \vee \neg p_8$
5.	$2 : p_1 \vee \neg p_5$		

Applying resolution to the set of literal clauses generates 66 new clauses. However, by feeding the SAT prover with Clauses 4 to 9 and assuming the initial partial assignment to be $V(p_1) = V(p_2) = V(p_3) = false$, a contradiction is generated. The clause $2 : (p_1 \vee p_2 \vee p_3)$ is then learnt and added to the clause set. Not only saturation by the modal prover is improved (15% less clauses are generated), but modal resolution can be immediately applied: the learnt clause is resolved with Clauses 1–3, generating the clause $1 : \neg l_1 \vee \neg l_2 \vee \neg l_3$. Moreover, and very importantly, from the soundness of the CDCL we can also obtain the corresponding resolution proof that would have generated the learnt clause, which allows to provide full proofs if a contradiction from the whole set of clauses is found. Although the combination might be costly (e.g. besides system calls, there is some overhead in translating the problems to MiniSat and back to KSP), we believe that by carefully choosing the set of clauses to be used as input for MiniSat we may be able to reduce the time KSP spends during the search for a proof.

References

[1] Carla P. Gomes, Henry Kautz, Ashish Sabharwal, and Bart Selman. 2008. Satisfiability Solvers. In *Handbook of Knowledge Representation*, Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter (Eds.). Elsevier, Amsterdam, 89–134.

[2] Cláudia Nalon, Ullrich Hustadt, and Clare Dixon. 2015. A Modal-Layered Resolution Calculus for K. In *TABLEAUX 2015, Proceedings (LNCS)*, Hans de Nivelle (Ed.), Vol. 9323. Springer, 185–200.

[3] Cláudia Nalon, Ullrich Hustadt, and Clare Dixon. 2016. KSP: A Resolution-Based Prover for Multimodal K. In *IJCAR 2016, Proceedings*, Nicola Olivetti and Ashish Tiwari (Eds.). Springer, 406–415.

[4] João P. Marques Silva, Inês Lynce, and Sharad Malik. 2009. Conflict-Driven Clause Learning SAT Solvers. In *Handbook of Satisfiability*, Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh (Eds.). Frontiers in Artificial Intelligence and Applications, Vol. 185. IOS Press, 131–153.

[5] Niklas Sörensson and Niklas Eén. 2013. Minisat-2.2.0. http://minisat.se. (2013).

WiL 2018, 8th of July, 2018, Oxford, UK
2018.