Recursive resolution for modal logic

Yevgeny Kazakov

Max-Planck-Institut für Informatik
Stuhlsatzenhausweg, 85, 66123 Saarbrücken, Germany
phone:+49 681-9325-223, e-mail:ykazakov@mpi-sb.mpg.de

Abstract. Resolution for the first order logic can be considered as a practical tool for obtaining a decision procedures for some theories (cf. [2]). For modal logics, however, there is no uniform formulation of the resolution principle, yet the normal modal logics are the most probable candidates to be decidable theories. The translational methods for modal logic, treated for instance in [5], yet possess some uniformness property, but does not let one to extract proofs from the refutations. On the other hand, direct methods (cf. [4], [1]) are local which gives not much practical use of them. This paper presents some arguments on generalization of the classical propositional resolution method to the language of modal logic. We give a resolution calculus for modal logic **K** that inherits some features of classical resolution and propose some suggestions of how can it be used for other modal logics.

1 Preliminaries

Here we start with the basic definitions for the propositional modal logic **K**, note some properties, define the normal forms for modal formulas and show the algorithms for transformations to these forms.

1.1 Modal logic K

We assume a usual notation for propositional modal logic: modal language consists of (countable) set of *propositional variables*: X_i, *boolean connectivities*: $\neg$ $\vee$ $\wedge$ $\rightarrow$, *boolean constants*: *truth* $\bot$, *false* $\top$, and *modal operators*: $\Box$ $\Diamond$. *Modal formulas* can be constructed using the following grammar:

$$F ::= \bot \mid \top \mid X_i \mid (\neg F) \mid (F \wedge F) \mid (F \vee F) \mid (F \rightarrow F) \mid (\Box F) \mid (\Diamond F)$$

which means that $\bot$, $\top$ and propositional variables are formulas and new formulas can be formed by using boolean connectivities and modal operators.

Propositional modal logic **K**, a minimal *normal* propositional modal logic, can be axiomatically defined as on the fig.1.1. One could give a deductive formulation for **K** using only the rule *modus ponens*, but for this purpose one needs to add an arbitrary prefix of $\Box$ in front of each axiom. We will call this system $\Box^{\mathbf{n}}\mathbf{K}$ (see the fig.1.1). Here $\Box^n A$ stands for $\underbrace{\Box \ldots \Box}_{n} A$. To distinguish the derivations in $\Box^{\mathbf{n}}\mathbf{K}$ from the derivations in **K** we will use the respective notations:

$\mathbf{K}$	$\Box^{\mathbf{n}}\mathbf{K}$
Axioms:	
PC Tautologies	$\Box^n\mathbf{PC}$ $\Box^n A$, were A is a tautology
N $\Box(A \to B) \to (\Box A \to \Box B)$	$\Box^n\mathbf{N}$ $\Box^n(\Box(A \to B) \to (\Box A \to \Box B))$
Rules of inference:	$n \geq 0$
MP $(A \to B),\ A \vdash B$	**MP** $(A \to B),\ A \vdash B$
Nec $A \vdash \Box A$	

Fig. 1. Deductive formulations for $\mathbf{K}$

$A_1, \ldots, A_n \vdash_{\mathbf{K}} A$ and $A_1, \ldots, A_n \vdash_{\Box^{\mathbf{n}}\mathbf{K}} A$. These formulations are equivalent: $\Box^{\mathbf{n}}\mathbf{K} \vdash A$ *iff* $\mathbf{K} \vdash A$. In addition, $\Box^{\mathbf{n}}\mathbf{K}$ satisfies the *deduction theorem*:

$$A_1, \ldots, A_n, B \vdash_{\Box^{\mathbf{n}}\mathbf{K}} A \quad \textit{iff} \quad A_1, \ldots, A_n \vdash_{\Box^{\mathbf{n}}\mathbf{K}} B \to A$$

The idea of using resolution for some calculus can be shown as follows. While usual deductive calculus tries to find a proof for A, resolution tries to derive a contradiction from $\neg A$, so resolution for $\mathbf{K}$ should satisfy a principle:

$$\vdash_{\Box^{\mathbf{n}}\mathbf{K}} A \quad \textit{iff} \quad \neg A \vdash_R \bot$$

In the scope of the deduction theorem this means that resolution calculus should be deductively equivalent to the original calculus.

1.2 Normal forms

The simple deduction algorithm needs to have a formulas in a simple form. Classical resolution for propositional logic was operated on the *conjunctions of clauses*, were a *clause* is a disjunction of propositional variables or constants or their negations. We will try to generalize this normal forms to the set of modal formulas.

We say that a modal formula is in *negation normal form* (NNF) if there are only propositional variables in scope of negation. Every modal formula can be transformed to an equivalent negation normal form by a simple procedure (see the fig.1.2). For example, $\mathrm{NNF}(\neg\Box(p \vee \Diamond q)) = \Diamond(\neg p \wedge \Box\neg q)$. However we can perform a further simplifications for formulas in $\mathbf{K}$ to form a *clausal form* (CF). The definition of clausal form uses a concept of *modal literal* and *modal clause*, which we recursively define as follows:

- *Atoms* are either propositional variables X_i, false $\bot$ or truth $\top$.
- *Modal literals* are formulas of the form either a, $\neg a$, were a is an atom, or $\Box C$, were C is a *modal clause*, or $\Diamond S$, were S is a conjunction of *modal clauses*.
- *Modal clause* is a disjunction of *modal literals*.

Modal formula is in a *clausal form* if it is a conjunction of clauses. The algorithms of translating the modal formulas to their negation normal forms and clausal forms are given in the table on fig.1.2 (rules should be applied to every subformula of the formula whenever is possible). From now on we will use the following notation. The letters a, b, c, p, q, r, s, t will denote atoms, letters L, C,

$\underline{\Rightarrow NNF}$	$\underline{NNF \Rightarrow CF}$
$\neg(A \wedge B) \Rightarrow \neg A \vee \neg B$	$(A \wedge B) \vee C \Rightarrow (A \vee C) \wedge (B \vee C)$
$\neg(A \vee B) \Rightarrow \neg A \wedge \neg B$	$\Box(A \wedge B) \Rightarrow \Box A \wedge \Box B$
$\neg\Box A \Rightarrow \Diamond\neg A$	
$\neg\Diamond A \Rightarrow \Box\neg A$	

Fig. 2. Transformation rules

S perhaps with subscripts and primes will denote modal literals, modal clauses and conjunctions of modal clauses respectively (sometimes we will omit "modal" and call them just literals or clauses). We may also omit the conjunctions between clauses, assuming that $S = C_1, \ldots, C_n$ stands for $S = C_1 \wedge \ldots \wedge C_n$. In this case we may identify S with a *set of clauses*. We will implicitly use a "set notation" for clauses, namely we will not distinguish between different orders of literals in the clauses and we may speak about the conjunctions or disjunctions of formulas in one-element sets or empty sets, assuming that $\wedge\{A\} = \vee\{A\} = A$, $\wedge\{\} = \top$, $\vee\{\} = \bot$.

Example 1. The following are clauses:
$\Box(p \vee q \vee \Diamond(r \wedge t))$, $\Diamond((p \vee q) \wedge t) \wedge \neg p$, $\neg p \vee \Box(\Box p \vee \Diamond(q \wedge \Box r))$;
the following are not:
$(p \vee (q \wedge r)) \wedge s$, $(p \vee \Box(q \wedge r)) \wedge s$, $\Box(p \vee q) \wedge \Diamond\Box((s \vee t) \wedge q)$.
Transformation to a clausal form:

$$\begin{aligned} \neg\Box(\Diamond p \vee \neg\Diamond(q \vee (r \wedge \Diamond p))) &\Rightarrow \Diamond(\Box\neg p \wedge \Diamond(q \vee (r \wedge \Diamond p))) && \text{(NNF)} \\ &\Rightarrow \Diamond(\Box\neg p \wedge \Diamond((q \vee r) \wedge (q \vee \Diamond p))) && \text{(CF)} \end{aligned}$$

The last formula can be also written as $\Diamond(\Box\neg p, \Diamond(q \vee r, q \vee \Diamond p))$.

2 Resolution calculus

As was mentioned before, given a candidate formula A the resolution procedure should try to infer a contradiction from the negative assumption about A. If the contradiction in some form will be obtained, it is said that the negative assumption is *refutable*, from what follows that A is derivable. The resolution calculus for **K** that will be given here, *saturates* the clausal normal form of $\neg A$ in order to obtain a set of clauses, containing $\bot$. The variant of this resolution calculus was initially proposed in [4]. It was formulated by means of the inference system for statements of the form $\Sigma(A, B) \to C$, or $\Gamma(A) \to C$, which have a meaning of *resolution rules*. As soon as the appropriate resolution rule is inferred in this calculus, it applies to the set of clauses. This resolution system was proved to be *refutationaly complete* w.r.t. **K** which means in this context that A is derivable in **K** *iff* the empty clause can be obtained from $CF(\neg A)$ by applying rules derivable in this calculus. In contrast to this system, in our formulation we will try to perform saturation another way round: by applying resolution rule without knowing it explicitly matching the input to some number of cases and recursively executing other resolution rules.

Deduction rules:

L1. $a, \neg a \underset{D}{\rightarrow} \bot$;

L2. $\Box C_1,\ \Box C_2 \underset{D}{\rightarrow} \Box C$ if $C_1,\ C_2 \underset{D}{\rightarrow} C$;

L3. $\Box C_1,\ \Diamond(C_2, S) \underset{D}{\rightarrow} \Diamond(C, C_2, S)$ if $C_1\ , C_2 \underset{D}{\rightarrow} C$;

C1. $C_1 \vee L_1,\ C_2 \vee L_2 \underset{D}{\rightarrow} C_1 \vee C_2 \vee L$ if $L_1,\ L_2 \underset{D}{\rightarrow} L$;

Simplification rules:

1. $C \vee \bot \underset{S}{\rightarrow} C$;
2. $C \vee L \vee L \underset{S}{\rightarrow} C \vee L$;
3. $C \vee \Diamond(\bot) \underset{S}{\rightarrow} C$;
4. $C \vee \Box C_1 \underset{S}{\rightarrow} C \vee \Box C_1'$ if $C_1 \overset{*}{\underset{S}{\rightarrow}} C_1'$;
5. $C_1 \vee \Diamond S \underset{S}{\rightarrow} C_1 \vee \Diamond S'$ if $S \overset{*}{\Rightarrow} S'$;

Saturation rules:

1. $S,\ \bot \Rightarrow \bot$;
2. $S,\ C_1 \Rightarrow S,\ C$ if $C_1 \overset{*}{\underset{S}{\rightarrow}} C$;
3. $S,\ C_1,\ C_2 \Rightarrow S\ , C_1,\ C_2, C$ if $C_1,\ C_2 \underset{D}{\rightarrow} C$;

Fig. 3. Resolution calculus **RK**

Formally, resolution system **RK** consists of three types of rules: *deduction rules*, *simplification rules* and *saturation rules*. *Deduction rules* have a form $L_1, L_2 \underset{D}{\rightarrow} L$ or $C_1, C_2 \underset{D}{\rightarrow} C$, which means that they are applied either to literals or to clauses. *Simplification rules* $C_1 \underset{S}{\rightarrow} C_2$ work on clauses and *saturation rules* $S_1 \Rightarrow S_2$ on the set of clauses. $C_1 \overset{*}{\underset{S}{\rightarrow}} C_2$ and $S_1 \overset{*}{\Rightarrow} S_2$ denote the transitive closures of respective relations. Most of the resolution rules has conditions which mean that the rule can be applied whenever the conditional rule can be applied.

Given a set of clauses S one tries to find a saturation of S – a set that cannot be saturated any further. To saturate S means try to perform one of the saturation rules consequently (in a given priority) which will recursively call some other rules. In each recursion step the objects can become only "simpler", so the process of saturation will finally terminate. Thus, **RK** gives a decision procedure for **K**.

Example 2. Suppose we want to check whether $K \vdash \Box(\neg\Box p \vee \Box(p \wedge q))$. The negation of this formula has a clausal form:
$S = \Diamond(\Box p,\ \Diamond(\neg p \vee q))$
Try to saturate S. Rule 1 cannot be applied since there is no $\bot$ in S, rule 2 gives us a task to simplify the clause
$C = \Diamond(\Box p,\ \Diamond(\neg p \vee q))$
There is only simplification rule 5 possible to apply, by which we should saturate
$S_1 = \Box p,\ \Diamond(\neg p \vee q)$
The saturation of the set $\{S_3\} = \neg p \vee q$ is no longer possible, so we should apply the saturation rule 3 for S_1:

$C_1 = \Box p$, $C_2 = \Diamond(\neg p \vee q)$
Again only one deduction rule is possible to apply, by which we resolve the literals
$L_1 = \Box p$, $L_2 = \Diamond(\neg p \vee q)$
Applying L3 we get $C_1' = p$, $C_2' = \neg p \vee q$
which can be resolved by C_1 and L_1 and will produce the clause $C' = q$. Getting all this back, we have performed one recursive resolution step with S and saturated it to
$S' = \Diamond(\Box p, \Diamond(\neg p \vee q), \Diamond(q, \neg p \vee q))$
Now it can be shown that all saturations of this set finally will produce it again, so our test formula is not derivable in **K**.

3 Conclusions and further work

The resolution procedure **RK** can be seen as a generalization of the classical propositional resolution procedure. For that reason, one could try to extend the results from the propositional resolution to the modal case, namely :

1. Adapt the rules to cope with non-ground formulas and unification;
2. Perform the refinement of resolution, using *well-founded orderings* of modal literals and *hyperresolution* (cf. [3]).

We do not seek for the extension of this procedure for the other modal logics (however in [4] it was extended to modal logics **Q**, **T**, **S4** and **S5**), since the resolution procedure for **K** may have enough power to cope with the other modal logics. For instance, **S4** can be formulated as $\Box^{\mathbf{n}}\mathbf{S4} = \Box^{\mathbf{n}}\mathbf{K} + \Box^n T + \Box^n 4$, were T is a reflexivity and 4 is a transitivity axioms. Deduction theorem for $\Box^{\mathbf{n}}\mathbf{K}$ will give us $\mathbf{S4} \vdash A$ *iff* $\Box^n K \vdash A_1 \wedge A_2 \wedge \ldots \wedge A_k \rightarrow A$, for some A_i to be instances of the axioms $\Box^n T$ and $\Box^n 4$. The last can be represented by a finite set of *non-ground* formulas. So, having the formulation of **RK** for non-ground formulas, we can treat it as a universal decision procedure for normal modal logics.

References

1. M. Abadi and Z. Manna. "Modal theorem proving." In *8th Int. Conf. on Automated Deduction*, pages 172-186. LNCS, Springer-Verlag, 1986.
2. A. Armando, S. Ranise and M. Rusinowitch. Uniform Derivation of Decision Procedures by Superposition. In *Proceedings 15th Workshop on Computer Science Logic.*, LNCS vol. 2142, 513-527, 2001.
3. L. Bachmair and H. Ganzinger. "Resolution theorem proving." In J. A. Robinson and A. Voronkov, editors, "Handbook of Automated Reasoning". Elsevier, 2000.
4. P. Enjalbert and L. Farinas del Cerro, "Modal resolution in clausal form." Theoretical Computer Science, 65(1):1-33, 1989.
5. H.J. Ohlbach. "Translation methods for non-classical logics – an overview." Bulletin of the Interest Group in Propositional and Predicate Logics (IGPL), 1(1):69–90, 1993.