A Recursive Shortcut for CEGAR: Application to the Modal Logic K Satisfiability Problem

Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima and **Valentin Montmirail**

CRIL, Univ. Artois and CNRS, F62300 Lens, France
{lagniez,leberre,delima,montmirail}@cril.fr

Abstract

Counter-Example-Guided Abstraction Refinement (CEGAR) has been very successful in model checking. Since then, it has been applied to many different problems. It is especially proved to be a highly successful practical approach for solving the PSPACE complete QBF problem. In this paper, we propose a new CEGAR-like approach for tackling PSPACE complete problems that we call RECAR (Recursive Explore and Check Abstraction Refinement). We show that this generic approach is sound and complete. Then we propose a specific implementation of the RECAR approach to solve the modal logic K satisfiability problem. We implemented both CEGAR and RECAR approaches for the modal logic K satisfiability problem within the solver MoSaiC. We compared experimentally those approaches to the state-of-the-art solvers for that problem. The RECAR approach outperforms the CEGAR one for that problem and also compares favorably against the state-of-the-art on the benchmarks considered.

1 Introduction

SAT technology has proven to be a very successful practical approach to solve some NP-complete problems [Biere *et al.*, 2009]. One of the main issues is to find the "right" encoding for the problem, i.e. to find a polynomial reduction from the original problem into a propositional formula in Conjunctive Normal Form (CNF, a set of clauses) which can be efficiently solved by a SAT solver [Prestwich, 2009]. The SAT solver is a generic problem solving engine, whose input is a satisfiability equivalent CNF representing the original problem. It often happens that either the SAT solver can solve efficiently the CNF or not at all (see e.g. the results of the SAT competitions [Järvisalo *et al.*, 2012]). A particular case is when the resulting CNF is very large: the time for generating and reading the input is greater than the time to solve it. This is due to the limited available main memory allocated to the approach and not the SAT solver itself. We address one of such particular cases in this paper.

For huge CNF encodings, specific approaches have been designed in the past, where a SAT solver is used as an oracle in a more complex procedure. One such procedure is called Counter-Example-Guided Abstraction Refinement (CEGAR) [Clarke *et al.*, 2003]. The SAT solver is fed with an abstraction of the original problem allowing more models (which we will call under-approximation). If the abstraction is unsatisfiable, then the original problem is also unsatisfiable (UNSAT shortcut). Else the procedure is able to verify if the model found for that abstraction is a correct solution for the original problem. In this case, we have an additional SAT shortcut to decide the satisfiability of the formula. If it is not the case, new constraints are added to prevent the solver from finding such spurious examples (refinement step) and the process repeats. Eventually, a complete satisfiability equivalent propositional formula is provided, and the SAT solver can decide the problem. One of the reasons for using CEGAR is that the complete formula is in practice too large to even be generated, so the only hope to solve the original problem is to "get lucky" (satisfiable shortcut) or to be able to take into account a specific structure of the problem (unsatisfiable shortcut).

This framework is elegant and has been applied to many areas: Satisfiability Modulo Theory [Brummayer and Biere, 2009], Planning [Seipp and Helmert, 2013] and more recently QBF [Janota *et al.*, 2016]. The latter is especially inspiring, because it appears to be the best practical solution overall to solve QBF formulas according to the latest QBF competition [Pulina, 2016]. The aim in this work is to follow these steps on another PSPACE complete problem, which is the satisfiability of modal logic K formulas. Several previous SAT-based approaches have already been proposed in the field of Modal Logic [Sebastiani and Tacchella, 2009], one could even argue that *SAT [Giunchiglia *et al.*, 2002] is already a CEGAR approach for the modal logic K.

In this paper, we introduce an extension of the CEGAR approach which includes a recursive step to introduce a new shortcut in the original CEGAR procedure. In our context, the main CEGAR loop contains a SAT shortcut ($\not\!\!\,$sat), while the recursive step allows the procedure to provide an UNSAT shortcut ($\not\!\!\,$unsat). We call this extension "Recursive Explore and Check Abstraction Refinement" (RECAR). The idea of mixing SAT and UNSAT shortcuts in a CEGAR procedure is not new: it has been already used for SMT [Brummayer and Biere, 2009] and for bug detection [Wang *et al.*, 2007]. Here the novelty is that we use an abstraction of the original problem in the loop, made possible by a recursive call to the main procedure. The RECAR procedure is generic, i.e. it is not bounded to a specific domain. In this paper, we present the conditions required on the abstractions used, and the correctness of the approach. Then, we instantiate our framework for the satisfiability of modal logic K, by providing abstraction functions for this problem and experimental results of the approach against the state-of-the-art provers. We believe that the good practical results we obtained by using RECAR on that particular domain are promising for other areas. This paper presents, in that respect, a generic procedure with a successful use case. The remainder of the paper is organized as follows: we first present the CEGAR approach; then we propose our new framework called RECAR and show its soundness and completeness; we provide an implementation of the RECAR approach for modal logic K; we finally compare the efficiency of the RECAR approach against the state-of-the-art modal logic K solvers.

2 CEGAR Preliminaries

Counter-Example-Guided Abstraction Refinement, CEGAR, is an incremental way to decide the satisfiability of formulas in classical propositional logic (CPL). It has been originally designed for model checking [Clarke *et al.*, 2003], i.e. to answer questions such as "Does $S \models P$ hold?" or, equivalently, "Is $S \land \neg P$ unsatisfiable?", where S describes a system and P a property. In such highly structured problems, it is often the case that only a small part of the formula is needed to answer the question. The idea behind CEGAR is to replace $\phi = S \land \neg P$ by an approximation ϕ', where ϕ' is easier to solve in practice than ϕ. There are two kinds of approximations: (1) an over-approximation of ϕ is a formula $\hat{\phi}$ such that $\hat{\phi} \models \phi$ holds: $\hat{\phi}$ has at most as many models as ϕ; (2) an under-approximation of ϕ is a formula $\check{\phi}$ such that $\phi \models \check{\phi}$ holds: $\check{\phi}$ has at least as many models as ϕ. Usually, ϕ is in CNF. Then, a classical way to under-approximate ϕ is to "forget" some clauses, i.e. $\check{\phi}$ is a subset of the clauses in ϕ. A model of $\check{\phi}$ also may by chance satisfy ϕ. Moreover, if $\check{\phi}$ is found to be unsatisfiable, then so is ϕ. This double possibility to conclude earlier makes under-approximation based CEGAR very popular. A classical way to over-approximate is to bound the generation of the formula ϕ to a given n smaller than the one needed to reach equi-satisfiability to the original problem (as in bounded model checking [Clarke *et al.*, 2003] or planning [Seipp and Helmert, 2013]). As such a model of $\hat{\phi}$ can be extended to a model of ϕ but the unsatisfiability of ϕ means that the bound n has to be increased and the process is repeated.

An example of a CEGAR using over-approximations is given on Fig. 1. It receives a formula ϕ as input and computes an over-approximation ψ. Then it uses a SAT solver to check whether ψ is satisfiable. If so it concludes that ϕ is satisfiable. Otherwise, ψ is refined, i.e. it gets closer to ϕ, until it is satisfiable, or until the refined over-approximation is detected to be equi-satisfiable to ϕ, denoted $\psi \equiv_{\text{sat}} \phi$, (i.e. $\exists M, M \models_1 \psi$ iff $\exists M', M' \models_2 \phi$)[1], where it concludes that ϕ is unsatisfiable. In the following, $\phi \overset{?}{\equiv}_{\text{sat}} \psi$ means an incomplete efficient equi-satisfiability test which returns yes or unknown. Recent

[1] $\models_1$ and $\models_2$ denote possibly different consequence relations (for propositional logic and modal logic K for instance).

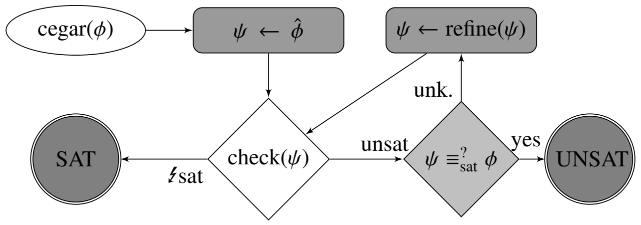
cegar(ϕ)
$\psi \leftarrow \hat{\phi}$
$\psi \leftarrow$ refine(ψ)
unk.
SAT
check(ψ)
unsat
$\psi \overset{?}{\equiv}_{\text{sat}} \phi$
yes
UNSAT
$\not\models$sat

Figure 1: The CEGAR framework with over-approximation

SAT solvers are able to check satisfiability "under assumption" [Eén and Sörensson, 2003], i.e. given the satisfiability of a set of literals called assumptions, and to provide in case of unsatisfiability a "reason" in terms of those literals for the unsatisfiability of a formula.

Definition 1 (Unsatisfiable core with assumptions)**.** *Let ϕ a CNF and A a consistent set of literals from ϕ. Let ϕ be satisfiable and $(\phi \land \bigwedge_{a \in A} a)$ be unsatisfiable. $L \subseteq A$ is an unsatisfiable core of ϕ if and only if $(\phi \land \bigwedge_{l \in L} l)$ is unsatisfiable.*

Therefore, a SAT oracle for ϕ, given A, can be seen as a procedure providing a pair (d, ψ) with $d \in \{\text{SAT}, \text{UNSAT}\}$ and ψ is a model of ϕ if $d = \text{SAT}$ or an unsatisfiable core of ϕ if $d = \text{UNSAT}$. Modern SAT-based procedures are able to take into account ψ in both cases. Unsatisfiable cores have been used for instance in a CEGAR approach for deciding the satisfiability of the propositional fragment of first-order logic [Khasidashvili *et al.*, 2015].

3 Recursive Explore and Check Abstraction Refinement

A classic CEGAR approach with over-approximation and a SAT shortcut performs well when the input is satisfiable. But generally, it does not perform well in problems which are unsatisfiable. The reason is that it may have to keep refining until it reaches equi-satisfiability with the original problem. One way to address this issue is to mix SAT and UNSAT shortcuts, as in [Brummayer and Biere, 2009] and [Wang *et al.*, 2007]. In these approaches, the methods alternate between over and under approximations.

The RECAR approach, depicted in Fig. 2 and Fun. 1, interleaves both kinds of approximations: each abstraction is performed with the information retrieved from solving the previous one. The UNSAT shortcut is implemented using a recursive call to the main procedure when a strict under-approximation $\check{\phi}$ can be built. One should also note that the proposed approach permits abstractions on two different levels: one is used to simplify the problem at the domain level (recursive call), while the other one is used to approximate the problem at the oracle level. In order to apply RECAR, the under-approximation $\check{\phi}$ and the over-approximation $\hat{\phi}$ must satisfy some properties. In the following, isSAT(ϕ) means that ϕ is satisfiable ($\not\models_1 \neg\phi$) and isUNSAT(ϕ) means ($\models_2 \neg\phi$), but on possibly different consequence relations. $RC(\phi, \check{\phi})$ denotes a Boolean function deciding if a Recursive Call should occur.

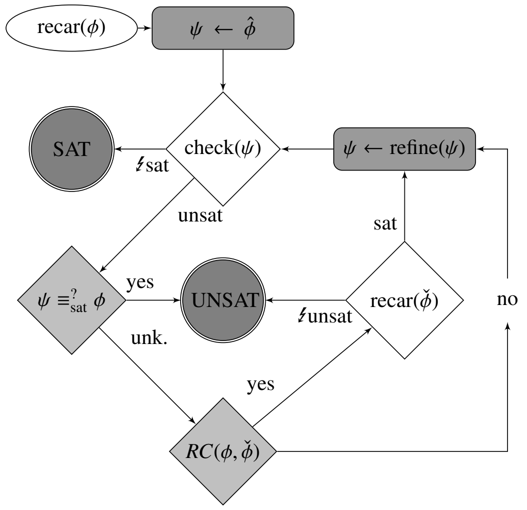

Figure 2: The RECAR framework

RECAR **assumptions:**

1. Function 'check' is a sound and complete implementation of 'isSAT' which terminates.
2. isSAT($\hat{\phi}$) implies isSAT(refine($\hat{\phi}$)).
3. There exists $n \in \mathbb{N}$ such that refine$^n(\hat{\phi}) \equiv^?_{\text{sat}} \phi$.
4. isUNSAT($\check{\phi}$) implies isUNSAT(ϕ).
5. Let under(ϕ) $= \check{\phi}$. There exists $n \in \mathbb{N}$ such that $RC(\text{under}^n(\phi), \text{under}^{n+1}(\phi))$ evaluates to false.

Note that we have isSAT($\hat{\phi}$) implies ϕ is satisfiable by Assumptions 2 and 3 together. In the following, we show that, under these assumptions, RECAR is sound, complete and terminates. To do so, we present the algorithm recar(ϕ) in Fun. 1.

Function 1: recar(ϕ)

```
1 $\psi \leftarrow \hat{\phi}$
2 while $\psi \equiv^?_{\text{sat}} \phi$ returns "unknown" do
3     if check($\psi$) = SAT then return SAT ;
4     if $RC(\phi, \check{\phi})$ then
5         if recar($\check{\phi}$) = UNSAT then return UNSAT ;
6     $\psi \leftarrow$ refine($\psi$) ;
7 return check($\psi$)
```

Theorem 1 (Soundness)**.** *If* recar(ϕ) *returns* SAT *then ϕ is satisfiable.*

Proof. Assume that recar(ϕ) returns SAT. This happens only if check(ψ) returns SAT, either from line 3 or from line 7. Thus, we know that isSAT(ψ) holds (by Assump. 1). But ψ equals to $\hat{\phi}$ or equals to refine$^n(\phi)$ for some $n \in \mathbb{N}$. Then ϕ is satisfiable (by Assump. 2 and 3). □

The intuition behind the proof of Th. 2 is that there are two ways to conclude that ϕ is UNSAT. In the first case, $\hat{\phi}$ is refined a finite number of times until it is detected equisatisfiable to ϕ and check returns UNSAT. Then ϕ is unsatisfiable. In the second case, one of the under-approximations is shown UNSAT, then ϕ is UNSAT (by Assump. 4).

Theorem 2 (Completeness)**.** *If* recar(ϕ) *returns* UNSAT *then* isUNSAT(ϕ).

Proof. By induction on the number k of recursive calls to recar (Line 5). Assume recar(ϕ) returns UNSAT after k recursive calls. In the induction base $k = 0$ (no recursive call). Then we must have exited the loop ($\psi \equiv^?_{\text{sat}} \phi$) and check($\psi$) returns UNSAT. This means that ψ is unsatisfiable (by Assump. 1) and therefore isUNSAT(ϕ) holds (because of equi-satisfiability). The induction hypothesis is: for all $k \leq n$, if recar(ϕ) returns UNSAT after k recursive calls then isUNSAT(ϕ). In the induction step $k = n + 1$. Then the conditions of lines 4 and 5 of the algorithm are true. This means that recar($\check{\phi}$) returns UNSAT after k recursive calls to recar. Then isUNSAT($\check{\phi}$) (by I.H.). Then isUNSAT(ϕ) (by Assump. 4). □

The intuition behind the proof of Th. 3 is that the function performs a finite number of recursive calls (Assump. 5). Moreover, each of these calls will have a finite number of refinements before terminating (Assump. 3).

Theorem 3 (Termination)**.** RECAR *terminates for any input ϕ.*

Proof. We have that (1) For all ϕ, there exists $n \in \mathbb{N}$ such that $RC(\text{under}^n(\phi), \text{under}^{n+1}(\phi))$ evaluates to false (by Assump. 5) and (2) For each $i \leq n$ there is $m_i \in \mathbb{N}$ such that refine$^{m_i}(\hat{\phi}) \equiv^?_{\text{sat}} \phi$ (by Assump. 3). Then, for any input ϕ, the recursive call of line 5 of the algorithm will be executed at most n times before the condition of line 4 becomes false. For each one of these recursive calls, the while-loop of the algorithm will be executed at most m_i times before the condition of line 2 becomes false. Therefore, for any input, recar halts after at most $n + \Sigma^n_{i=0}(m_i)$ recursive calls. □

4 Solving K-SAT with RECAR

In this section, we show how RECAR is applied to solve the satisfiability problem for modal logic K, which is a PSPACE complete problem [Ladner, 1977; Halpern and Moses, 1992].

4.1 Modal Logic K

Before showing how we applied the RECAR approach, let us define formally what the modal logic K is.

Let a finite non-empty set of propositional variables $\mathbb{P} = \{p_1, p_2, \dots\}$ and a set of m unary modal operators $\mathbb{M} = \{\Box_1, \dots, \Box_m\}$ be given. The language of K (noted $\mathcal{L}$) is the set of formulas containing $\mathbb{P}$, closed under the set of propositional connectives $\{\neg, \wedge\}$ and the set of modal operators in $\mathbb{M}$. We also use the standard abbreviations for $\top$, $\bot$, $\vee$ and $\Diamond$. For instance $\Diamond_a\phi = \neg\Box_a\neg\phi$.

Definition 2 (Model)**.** *A Kripke model is a triplet $M = \langle W, \{R_a \mid \Box_a \in \mathbb{M}\}, V\rangle$, where: W is a non-empty set of possible worlds; Each $R_a \subseteq W \times W$ is a binary accessibility relation on W; $V : \mathbb{P} \rightarrow 2^W$ is a valuation function which associates, to each $p \in \mathbb{P}$, the set of possible worlds from W where p is* *true. A pointed Kripke model is a pair $\langle M, w\rangle$, where M is a Kripke model and w is a possible world in W. Thereafter, whenever we use the term 'model' we refer to 'pointed Kripke model'.*

In the following, the size of a model $\langle M, w\rangle$, which is the number of elements in W, is denoted by $|M|$.

Definition 3 (Satisfaction relation)**.** *The relation $\models$ between models and formulas is recursively defined as follows:*

$$\begin{array}{lll}\langle M,w\rangle \models p & \textit{iff} & w \in V(p)\\ \langle M,w\rangle \models \neg\phi & \textit{iff} & \langle M,w\rangle \not\models \phi\\ \langle M,w\rangle \models \phi_1 \wedge \phi_2 & \textit{iff} & \langle M,w\rangle \models \phi_1 \textit{ and } \langle M,w\rangle \models \phi_2\\ \langle M,w\rangle \models \Box_a\phi & \textit{iff} & (w,w') \in R_a \textit{ implies } \langle M,w'\rangle \models \phi\end{array}$$

Definition 4 (Validity)**.** *As usual, a formula $\phi \in \mathcal{L}$ is valid (noted $\models \phi$) if and only if it is satisfied by all models $\langle M, w\rangle$. A formula $\phi \in \mathcal{L}$ is satisfiable in K (noted* isKSAT(ϕ)*) if and only if $\not\models \neg\phi$. We also use* isKUNSAT(ϕ) *to mean $\models \neg\phi$.*

In the sequel, we define a translation from modal logic K to classical propositional logic.

Definition 5 (Translation)**.**

$$\begin{aligned}\mathrm{tr}(\phi, n) &= \mathrm{tr}'(\mathrm{nnf}(\phi), 0, n)\\ \mathrm{tr}'(p, i, n) &= p_i\\ \mathrm{tr}'(\neg p, i, n) &= \neg p_i\\ \mathrm{tr}'(\phi \wedge \psi, i, n) &= \mathrm{tr}'(\phi, i, n) \wedge \mathrm{tr}'(\psi, i, n)\\ \mathrm{tr}'(\phi \vee \psi), i, n) &= \mathrm{tr}'(\phi, i, n) \vee \mathrm{tr}'(\psi, i, n)\\ \mathrm{tr}'(\Box_a\phi, i, n) &= \bigwedge_{j=0}^{n}(r^a_{i,j} \rightarrow \mathrm{tr}'(\phi, j, n))\\ \mathrm{tr}'(\Diamond_a\phi, i, n) &= \bigvee_{j=0}^{n}(r^a_{i,j} \wedge \mathrm{tr}'(\phi, j, n))\end{aligned}$$

The translation adds fresh variables p_i and $r^a_{i,j}$ to the formula: p_i denotes that variable p is true in the world w_i whereas $r^a_{i,j}$ corresponds to w_j being accessible from w_i by the relation a. We have the following as an immediate result (where $nm(\phi)$ is the number of modal operators in ϕ) based on §25.2.2 [Sebastiani and Tacchella, 2009].

Theorem 4. isKSAT(ϕ) *if and only if* isSAT(tr(ϕ, $nm(\phi)+1$)).

Therefore, in order to decide the satisfiability of a formula $\phi \in \mathcal{L}$, one can simply feed a SAT solver with tr(ϕ, $nm(\phi)+1$). In fact, this is the approach proposed in Km2SAT [Sebastiani and Vescovi, 2009]. The main issue is that the translation may generate an exponentially larger formula. In this paper, we try to circumvent this problem by using RECAR. For simplicity, from now on we use tr(ϕ) instead of tr(ϕ, $nm(\phi) + 1$).

In [Caridroit *et al.*, 2017], the authors also use this translation from modal logic S5 to classical propositional logic, but replacing $nm(\phi)$ by the diamond degree dd(ϕ), which is generally smaller. Unfortunately, this cannot be used for K. The counter-example below shows why.

Counter-Example 1. *Let $\phi = (p_1 \wedge p_2 \wedge p_3) \wedge (\Diamond_a(p_1 \wedge p_2 \wedge \neg p_3 \wedge \Box_a(p_1 \wedge \neg p_2 \wedge p_3))) \wedge (\Diamond_a(p_1 \wedge \neg p_2 \wedge \neg p_3 \wedge \Box_a(\neg p_1 \wedge \neg p_2 \wedge p_3))) \wedge (\Box_a\Diamond_a p_3)$, with* dd($\phi$) = 3. *This formula is satisfied by the model in Fig. 3. However, it is easy to see that there is no model satisfying ϕ with less than 5 possible worlds.*

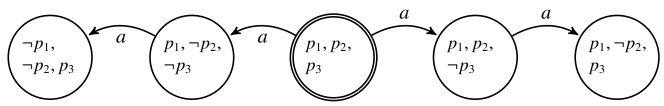

Figure 3: $M \models \phi$

4.2 Over-approximation

Now, in order to apply the RECAR approach, we first need to find an over-approximation which respects the assumptions presented in Sec. 3.

Definition 6 (Over-approximation)**.** *Let $\phi \in \mathcal{L}$. The over-approximation of ϕ, denoted $\hat{\phi}$, is the formula* tr(ϕ, 1).

Definition 7 (Refinement)**.** *Let $1 \le n \le nm(\phi)+1$. The refinement of* tr(ϕ, n)*, noted* refine(tr(ϕ, n)) *is the formula* tr(ϕ, $n+1$).

Theorem 5. *If* isSAT(tr(ϕ, n)) *then* isSAT(tr(ϕ, $n+1$))*, for all $1 < n \le nm(\phi) + 1$. (*RECAR *Assump. 2)*

Proof Sketch. The idea is that if ϕ is satisfied by a model M with n worlds, then we can find a model M' with $n+1$ worlds satisfying ϕ. The additional world is just not accessible from the ones already in M. □

The latter result allows us to use this over-approximation and refinement in the RECAR approach. It is easy to see that RECAR assumptions 2 and 3 are satisfied.

4.3 Under-approximation

To understand the intuition behind the under-approximation we use an example. Let $\phi = (\Diamond p \wedge \Box\neg p \wedge \chi)$ for some $\chi \in \mathcal{L}$, where $nm(\chi)$ is huge. This is clearly unsatisfiable because $(\Diamond p \wedge \Box\neg p)$ is unsatisfiable. One can see that right away without even knowing what χ looks like. However, a CEGAR approach using the over-approximation and refinement defined earlier will take a long time before finally conclude it. The reason is that each refinement tr(ϕ, $n + 1$) of the original formula will be shown unsatisfiable and it will not stop until the huge number $nm(\phi) + 1$ is reached.

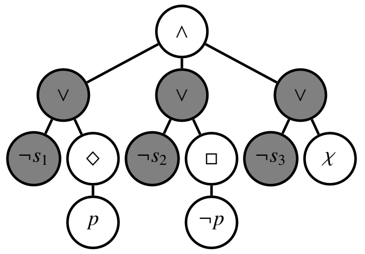

Figure 4: How selectors are applied to $\phi = (\Diamond p \wedge \Box\neg p \wedge \chi)$

To avoid these pathological cases, the RECAR approach also performs under-approximations. To see how it works, let us take that formula ϕ again. First, we add to each conjunct in ϕ a fresh variable s_i (a selector) that will be assumed to be true by the SAT solver, as done in Fig. 4. Then, we make the first over-approximation tr(ϕ, 1) and give it to a modern SAT solver. The solver will return UNSAT with an unsatisfiable core. From this core, we extract a set of selectors *core*. Let us assume, in our example, that $core = \{s_1, s_2\}$. This means that the formula $\check{\phi} = (\Diamond p \wedge \Box\neg p)$, which is the one labelled by the selectors, is enough to prove the unsatisfiability of ϕ with only 1 possible world. Proving the unsatisfiability of $\check{\phi}$ will imply that ϕ is unsatisfiable. Note that, in this specific case, $nm(\check{\phi})$ is much smaller than $nm(\phi)$. Thus the CEGAR approach applied to $\check{\phi}$ will succeed much earlier, while it may have failed for the entire formula ϕ. Formally, we have the following.

Definition 8 (Under-Approximation)**.**

$$\text{under}(p, core) = p$$
$$\text{under}(\neg p, core) = \neg p$$
$$\text{under}(\Box_a\phi, core) = \Box_a(\text{under}(\phi, core))$$
$$\text{under}(\Diamond_a\phi, core) = \Diamond_a(\text{under}(\phi, core))$$
$$\text{under}((\phi \wedge \psi), core) = \text{under}(\phi, core) \wedge \text{under}(\psi, core)$$
$$\text{under}((\psi \vee \chi), core) = \begin{cases} \text{under}(\chi, core) & \textit{if } \psi = \neg s_i, s_i \in core \\ \top & \textit{if } \psi = \neg s_i, s_i \notin core \\ (\text{under}(\psi, core) & \\ \vee\, \text{under}(\chi, core)) & \textit{otherwise} \end{cases}$$

Theorem 6. isKUNSAT(under(ϕ, *core*)) *implies* isKUNSAT(ϕ).

The intuition of the proof is that each selector s_i enables an operand in a conjunction of the formula. Each time function 'under' is called with a non-empty *core*, operands not enabled with a selector from the *core* will be removed from the formula.

Proof. Let ϕ be in NNF. We show that isKSAT(ϕ) implies isKSAT(under(ϕ, *core*)) by induction on the structure of ϕ. Assume isKSAT(ϕ). Then $\exists\, M, w$ s.t. $\langle M, w\rangle \models \phi$. There are two cases in the induction base: (1) $\phi = p$ and (2) $\phi = \neg p$. In both of them under(ϕ, *core*) = ϕ. There are four cases in the induction step:
(1) $\phi = \Diamond_a(\psi)$. $\exists\, M, w$ s.t. $\langle M, w\rangle \models \Diamond_a(\psi)$. Then $\exists\, M, w'$ s.t. $(w, w') \in R$, $\langle M, w'\rangle \models \psi$. Then $\langle M, w'\rangle \models$ under(ψ, *core*) by induction hypothesis. Thus $\langle M, w\rangle \models$ under(ϕ, *core*);
(2) $\phi = \Box_a(\psi)$. This case is analogous to (1).
(3) $\phi = (\psi \wedge \chi)$. $\exists\, \langle M, w\rangle \models (\psi \wedge \chi)$. Then $\langle M, w\rangle \models \psi$ and $\langle M, w\rangle \models \chi$. Then $\langle M, w\rangle \models$ under(ψ, *core*) and $\langle M, w\rangle \models$ under(χ, *core*) by induction hypothesis. Thus $\langle M, w\rangle \models$ under(ϕ, *core*);
(4) $\phi = (\psi \vee \chi)$. We consider the three cases:
(4.a) $\psi = \neg s_i$ and $s_i \in core$. Then $\exists\, \langle M, w\rangle \models (\neg s_i \vee \chi)$ but $s_i \in V(w)$, then $\langle M, w\rangle \models \chi$. Then $\langle M, w\rangle \models$ under(χ, *core*) by induction hypothesis. Thus $\langle M, w\rangle \models$ under(ϕ, *core*);
(4.b) $\psi = \neg s_i$ and $s_i \notin core$. $\exists\, \langle M, w\rangle \models (\neg s_i \vee \chi)$. but we always have $\langle M, w\rangle \models \top$. Thus $\langle M, w\rangle \models$ under(ϕ, *core*).
(4.c) This case is analogous to (3). □

Theo. 6 shows that function 'under' satisfies RECAR Assump. 4. To see that it also satisfies Assump. 5, note that the length of under^{n+1}(ϕ, *core*) is smaller or equal to that of under^{n}(ϕ, *core'*) (even though the sets *core* and *core'* usually differ).

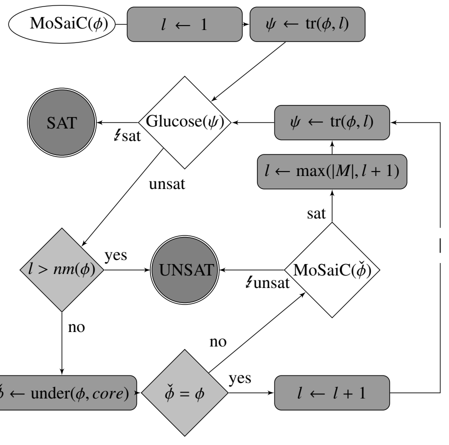

Figure 5: MoSaiC : RECAR for modal logic K

5 MoSaiC : RECAR for K-SAT

We implemented the RECAR approach for modal logic K satisfiability problem within the solver MoSaiC, using the over and under approximations defined in the previous section. MoSaiC combines several features found in state-of-the-art solvers. As in Km2SAT [Sebastiani and Vescovi, 2009], it optimises the input by performing the rules: Box Lifting, Flattening, and Truth Propagation through modal and Boolean operators (see [Sebastiani and Vescovi, 2009] for more details). MoSaiC also uses the SAT solver Glucose in incremental mode [Eén and Sörensson, 2003; Audemard *et al.*, 2013] to decide the satisfiability of each ψ.

Note that some implementation details differ a bit from Fig. 5. For instance, we do not call Glucose on ψ but on an updated ψ' with selectors on conjuncts under the assumption that these selectors are satisfied; we do not need to generate the under approximation $\check{\phi}$ to test the condition $\check{\phi} = \phi$: we just need to know the number of selectors involved in the unsatisfiability of the formula. We also return a Kripke model in the main procedure, not just SAT/UNSAT. We take advantage of such information to provide a new bound for l. And finally, note that in our case max($|M|$, $l+1$) always returns $|M|$ because it is not possible to find a model smaller than M by construction of $\check{\phi}$. The CEGAR solver presented in Sec. 6 uses a similar schema, but without the recursive RECAR step.

6 Experiments

We compared our approach against existing solvers considered state-of-the-art in [Nalon *et al.*, 2016], namely $K_S P$ 0.1 [Nalon *et al.*, 2016], BDDTab 1.0 [Goré *et al.*, 2014], FaCT++ 1.6.4 [Tsarkov and Horrocks, 2006], InKreSAT 1.0 [Kaminski and Tebbi, 2013], *SAT 1.3 [Giunchiglia *et al.*, 2002], Km2SAT 1.0 [Sebastiani and Vescovi, 2009] combined with the same Glucose SAT solver we use in MoSaiC, Spartacus 1.0 [Götzmann *et al.*, 2010] and a combination of the optimized functional translation [Horrocks *et al.*, 2007] with Vampire 4.0 [Kovács and Voronkov, 2013]. MoSaiC was configured as a standard CEGAR approach or with the proposed RECAR approach. We chose to execute these solvers on the following benchmarks: the complete set of TANCS-2000

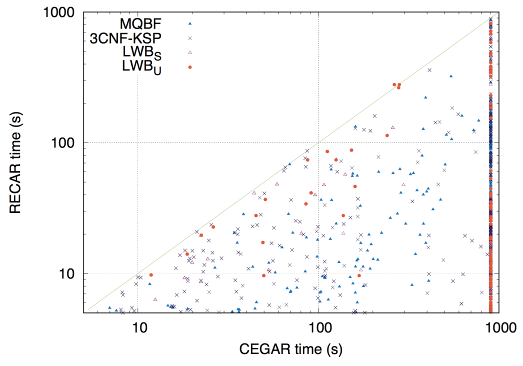

Figure 6: Scatter-plot CEGAR vs RECAR

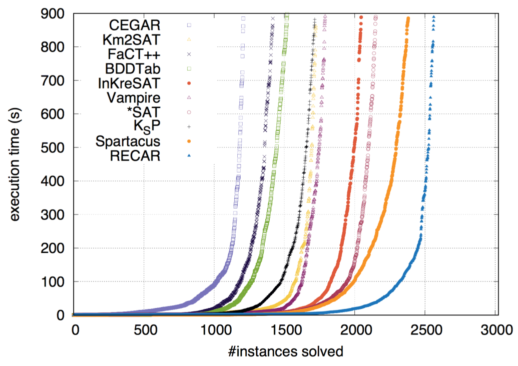

Figure 7: Runtime distribution on all the benchmarks

modalised random QBF (MQBF) formulae [Massacci and Donini, 2000] complemented by the additional MQBF formulae provided by [Kaminski and Tebbi, 2013], 1016 formulae, 617 satisfiable, 399 unsatisfiable; LWB basic modal logic benchmark formulae [Balsiger *et al.*, 2000], with 56 formulae chosen from each of the 18 parametrized classes, generated from the script given by the authors of [Nalon *et al.*, 2016], 1008 formulae, 504 satisfiable, 504 unsatisfiable; randomly generated $3CNF_{KSP}$ formulae [Patel-Schneider and Sebastiani, 2011] of depth 1 or 2, 1000 formulae, 457 satisfiable, 464 unsatisfiable. Few of the considered solvers could deal with multiple modalities like MoSaiC, however to the best of our knowledge, there is no standard benchmark for modal logic K containing multiple modalities. We used a memory limit of 32GB and a runtime limit of 900 seconds per solver per benchmark. Note that due to lack of space, the results presented here are global[2]. The behavior of the solvers vary a lot with the benchmark families. We believe however that they provide an interesting insight of the capabilities of the proposed approach.

RECAR vs CEGAR We can see on Fig. 6 that for most benchmarks, the RECAR approach outperforms the CEGAR one. The under approximation often provides a formula with a much smaller nm value, which produces a CNF of reasonable size to be handed in to the SAT solver. Note that in this plot, memory out for the CEGAR approach is denoted by a timeout, i.e. a point at 900s. This is mainly due to improvements in solving unsatisfiable benchmarks (1118/1367) for RECAR vs (155/1367) for CEGAR. For satisfiable benchmarks, the bound update resulting from the recursive call helps to reach faster a satisfiable formula. 1446 for RECAR vs 1053 for CEGAR.

Comparison with state-of-the-art We can see on Fig. 7 that our over-approximation CEGAR approach is the worst solver whereas our RECAR approach outperforms the other solvers. *Km2SAT* performs specific reasoning to detect earlier some UNSAT benchmarks without generating the CNF, which explains why it performs much better than our CEGAR approach. *SAT interleaves SAT reasoning and domain reasoning, and can be considered as an under-approximation CEGAR approach. It shows good results, despite being tied with the old SAT solver SATO[3]. Our best competitor, Spartacus, is based on a tableaux method, not on SAT: SAT based-techniques were not the best way to tackle such problems up to now. Spartacus reaches the timeout on unsolved benchmarks while we exhaust the available memory: the solvers behave quite differently and have different limits.

7 Conclusion

We proposed here a new approach to solve decision problems using a recursive abstraction refinement approach. We showed it is sound and complete and we instantiated it for the modal logic K satisfiability problem. We compared our approach against solvers representing, to the best of our knowledge, the state-of-the-art for practical K-SAT solving, on a wide range of classical modal logic benchmarks. A basic over-approximation based CEGAR approach is not competitive at all, because many of the available benchmarks are UNSAT. Our RECAR approach, mixing SAT and UNSAT shortcuts, outperformed the other solvers on the benchmarks considered. Those promising results are a first step toward more efficient modal logic solvers: MoSaiC can be extended to other modal logics such as KT, S4, S5 and KD45, by adapting the current translation into CNF. We also believe that RECAR is an appealing framework for tackling decision problems above NP in the polynomial hierarchy. It is not clear yet for us if existing CEGAR related approaches for QBF such as [Janota *et al.*, 2016] could fit in such framework. This is an exciting future work.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments and for detecting a flaw which is now corrected. Part of this work was supported by the ANR project SATAS (ANR-15-CE40-0017), the French Ministry for Higher Education and Research and the Nord-Pas de Calais Regional Council through the "Contrat de Plan État Région (CPER)" and by an EC FEDER grant.

[2]Additional results can be found on `http://www.cril.univ-artois.fr/~montmirail/mosaic`

[3]*SAT is deeply integrated with SATO, which makes very difficult an update to a more recent SAT solver.

References

[Audemard *et al.*, 2013] Gilles Audemard, Jean-Marie Lagniez, and Laurent Simon. Improving glucose for incremental SAT solving with assumptions: Application to MUS extraction. In *Proc. of SAT'13*, pages 309–317, 2013.

[Balsiger *et al.*, 2000] Peter Balsiger, Alain Heuerding, and Stefan Schwendimann. A Benchmark Method for the Propositional Modal Logics K, KT, S4. *J. Autom. Reasoning*, 24(3):297–317, 2000.

[Biere *et al.*, 2009] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[Brummayer and Biere, 2009] Robert Brummayer and Armin Biere. Effective bit-width and under-approximation. In *Proc. of EUROCAST'09*, pages 304–311, 2009.

[Caridroit *et al.*, 2017] Thomas Caridroit, Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima, and Valentin Montmirail. A SAT-based Approach For Solving The Modal Logic S5-Satisfiability Problem. In *Proc. of AAAI'17*, 2017.

[Clarke *et al.*, 2003] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, 2003.

[Eén and Sörensson, 2003] Niklas Eén and Niklas Sörensson. An Extensible SAT-solver. In *Proc. of SAT'03*, pages 502–518, 2003.

[Giunchiglia *et al.*, 2002] Enrico Giunchiglia, Armando Tacchella, and Fausto Giunchiglia. SAT-Based Decision Procedures for Classical Modal Logics. *J. Autom. Reasoning*, 28(2):143–171, 2002.

[Goré *et al.*, 2014] Rajeev Goré, Kerry Olesen, and Jimmy Thomson. Implementing Tableau Calculi Using BDDs: BDDTab System Description. In *Proc. of IJCAR'14*, pages 337–343, 2014.

[Götzmann *et al.*, 2010] Daniel Götzmann, Mark Kaminski, and Gert Smolka. Spartacus: A Tableau Prover for Hybrid Logic. *Electr. Notes Theor. Comput. Sci.*, 262:127–139, 2010.

[Halpern and Moses, 1992] Joseph Y. Halpern and Yoram Moses. A Guide to Completeness and Complexity for Modal Logics of Knowledge and Belief. *Artif. Intell.*, 54(2):319–379, 1992.

[Horrocks *et al.*, 2007] Ian Horrocks, Ullrich Hustadt, Ulrike Sattler, and Renate A. Schmidt. 4 Computational modal logic. *Studies in Logic and Practical Reasoning*, 3:181–245, 2007.

[Janota *et al.*, 2016] Mikolás Janota, William Klieber, Joao Marques-Silva, and Edmund M. Clarke. Solving QBF with counterexample guided refinement. *Artif. Intell.*, 234:1–25, 2016.

[Järvisalo *et al.*, 2012] Matti Järvisalo, Daniel Le Berre, Olivier Roussel, and Laurent Simon. The international SAT solver competitions. *AI Magazine*, 33(1), 2012.

[Kaminski and Tebbi, 2013] Mark Kaminski and Tobias Tebbi. InKreSAT: Modal Reasoning via Incremental Reduction to SAT. In *Proc. of CADE'13*, pages 436–442, 2013.

[Khasidashvili *et al.*, 2015] Zurab Khasidashvili, Konstantin Korovin, and Dmitry Tsarkov. EPR-based k-induction with Counterexample Guided Abstraction Refinement. In *Proc. of GCAI'15*, volume 36, pages 137–150, 2015.

[Kovács and Voronkov, 2013] Laura Kovács and Andrei Voronkov. First-Order Theorem Proving and Vampire. In *Proc. of CAV'13*, pages 1–35, 2013.

[Ladner, 1977] Richard E. Ladner. The Computational Complexity of Provability in Systems of Modal Propositional Logic. *SIAM J. Comput.*, 6(3):467–480, 1977.

[Massacci and Donini, 2000] Fabio Massacci and Francesco M. Donini. Design and results of TANCS-2000 non-classical (modal) systems comparison. In *Proc. of TABLEAUX'00*, pages 52–56, 2000.

[Nalon *et al.*, 2016] Cláudia Nalon, Ullrich Hustadt, and Clare Dixon. $K_S P$: A Resolution-Based Prover for Multimodal K. In *Proc. of IJCAR'16*, pages 406–415, 2016.

[Patel-Schneider and Sebastiani, 2011] Peter F. Patel-Schneider and Roberto Sebastiani. A new general method to generate random modal formulae for testing decision procedures. *CoRR*, abs/1106.5261, 2011.

[Prestwich, 2009] Steven David Prestwich. CNF encodings. In *Handbook of Satisfiability*, pages 75–97. IOS Press, 2009.

[Pulina, 2016] Luca Pulina. The ninth QBF solvers evaluation - preliminary report. In *Proc. of the 4th International Workshop (QBF 2016) co-located with (SAT 2016)*, pages 1–13, 2016.

[Sebastiani and Tacchella, 2009] Roberto Sebastiani and Armando Tacchella. SAT techniques for modal and description logics. In *Handbook of Satisfiability*, pages 781–824. IOS Press, 2009.

[Sebastiani and Vescovi, 2009] Roberto Sebastiani and Michele Vescovi. Automated Reasoning in Modal and Description Logics via SAT Encoding: the Case Study of K(m)/ALC-Satisfiability. *J. Artif. Intell. Res. (JAIR)*, 35:343–389, 2009.

[Seipp and Helmert, 2013] Jendrik Seipp and Malte Helmert. Counterexample-guided cartesian abstraction refinement. In *Proc. of ICAPS'13*, 2013.

[Tsarkov and Horrocks, 2006] Dmitry Tsarkov and Ian Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proc. of IJCAR'06*, pages 292–297, 2006.

[Wang *et al.*, 2007] Chao Wang, Aarti Gupta, and Franjo Ivancic. Induction in CEGAR for detecting counterexamples. In *Proc. of FMCAD'07*, pages 77–84, 2007.