Un raccourci récursif pour CEGAR : Application au problème de satisfiabilité en logique modale K

Jean-Marie Lagniez[1] **Daniel Le Berre**[1] **Tiago de Lima**[1] **Valentin Montmirail**[1]

[1] CRIL, Univ Artois et CNRS, France
{lagniez,leberre,delima,montmirail}@cril.fr

Résumé

L'approche Counter-Example Guided Abstraction Refinement (CEGAR) a été un grand succès dans la vérification de modèle. Depuis lors, elle a été appliquée à de nombreux problèmes différents. Il s'avère qu'il s'agit d'une approche pratique très efficace pour résoudre le problème QBF qui est PSPACE-complet. Dans cet article, nous proposons une nouvelle approche semblable à CEGAR pour aborder des problèmes PSPACE, approche que nous appelons RECAR (Recursive Explore and Check Abstraction Refinement). Nous montrons que cette approche générique est correcte et complète. Ensuite, nous proposons une instantiation du framework RECAR pour résoudre le problème de satisfiabilité en logique modale K. Nous implémentons les deux approches CEGAR et RECAR pour déterminer la cohérence d'une formule en logique modale K au sein du solveur MoSaiC. Nous comparons expérimentalement ces approches face aux solveurs de l'état de l'art. L'approche RECAR surpasse l'approche CEGAR sur ces problèmes et se compare favorablement aux solveurs de l'état de l'art sur les benchmarks considérés.

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

L'idée d'utiliser un solveur SAT s'est révélée être une approche pratique très efficace pour résoudre certains problèmes NP-complets [3]. Un des principaux problème est de trouver le "bon" encodage pour le problème, c'est-à-dire de trouver la réduction polynomiale du problème original vers une formule de la logique propositionnelle en Forme Normal Conjonctive (CNF, un ensemble de clauses) qui peut être efficacement résolue par un solveur SAT [23]. Le solveur SAT est un moteur générique de résolution de problème, dont le paramètre d'entrée est une CNF équicohérent au problème original. Il arrive souvent que soit le solveur SAT résout efficacement la CNF soit pas du tout (voir les résultats de la compétition SAT [14]). Un cas particulier est quand la CNF générée est trop grande : le temps pour générer et lire la CNF est plus grand que le temps pour la résoudre. Ceci est due à une quantité limitée de mémoire allouée à la résolution et pas au solveur SAT directement. Nous abordons l'un de ces cas particuliers dans cet article.

Pour les CNF possédant énormément de clauses, des approches spécifiques ont été créées dans le passé, où le solveur SAT est utilisé tel un oracle dans une procédure plus complexe. Une de ces procédure est appelée Counter-Example-Guided Abstraction Refinement (CEGAR) [6]. Le solveur SAT est alimenté avec une abstraction du problème original qui permet plus de modèles (ce que nous appellerons ici une sous-abstraction). Si cette sous-abstraction est incohérent, alors le problème original l'est aussi (court-circuit UNSAT). Sinon la procédure est capable de vérifier si le modèle trouvé pour la sous-abstraction est une solution pour le problème original. Dans ce cas, nous avons un court-circuit SAT additionnel capable de décider la cohérence de la formule. Si ce n'est pas le cas, de nouvelles contraintes sont ajoutées pour prévenir le solveur de trouver de tel exemple fallacieux (étape de raffinement) et la procédure se répète. Éventuellement une formule équi-

cohérente est détectée, et le solveur SAT peut décider ainsi le problème. Une des raisons pour utiliser CEGAR est qu'en pratique, la formule complète est trop grande pour ne serait-ce qu'être générée, par conséquent, le seul espoir pour résoudre ce problème est "d'avoir de la chance' (court-circuit SAT) ou d'être capable de prendre en compte une structure spécifique du problème (court-circuit UNSAT). Cette approche est élégante et a été appliquée à de nombreux problèmes : la cohérence Modulo Théorie (SMT) [4], la Planification [27] ou plus récemment à QBF [13]. Il apparaît que CEGAR est la meilleur approche en pratique pour résoudre des problèmes QBF (selon la dernière compétition QBF [24]). Ce résultat étonnant a motivé notre travail. Notre but est d'appliquer une approche basée sur SAT à un autre problème PSPACE-complet, déterminer la cohérence d'une formule en logique modale K. De très nombreuses approches basées sur SAT ont déjà été proposées dans le cadre de la logique modale [25], d'aucun pourrait même argumenter que *SAT [8] est déjà une approche CEGAR pour la logique modale K.

Dans cet article, nous introduisons une extension à l'approche CEGAR qui propose une étape récursive pour introduire un nouveau court-circuit dans la procédure CEGAR originale. Dans notre contexte, la boucle CEGAR principale contient un court-circuit SAT ($\lightning$sat), pendant que l'étape récursive permet à la procédure de fournir un court-circuit UNSAT ($\lightning$unsat). Nous appelons cette extension "Recursive Explore and Check Abstraction Refinement" (RECAR). L'idée de mixer les court-circuits SAT et UNSAT dans une procédure CEGAR n'est pas nouvelle : cela a déjà été employée pour SMT [4] et pour la détection de bug [29]. Ici, la nouveauté est que nous utilisons une abstraction du problème original dans la boucle, ce qui est rendu possible grâce à l'appel récursif dans la boucle principale.

L'approche RECAR est générique, elle n'est pas spécifique à un domaine. Dans cet article, nous présentons les conditions requises sur les abstractions utilisées et nous prouvons la correction et la complétude de l'approche. Ensuite, nous instancions notre approche pour déterminer la cohérence d'une formule en logique modale K, en fournissant des fonctions d'abstractions pour ce problème et des résultats expérimentaux de cette approche face aux solveurs de l'état de l'art. Nous pensons que les bons résultats pratiques que nous obtenons en utilisant RECAR sur ce domaine particulier sont prometteurs pour d'autres domaines. Cet article présente, dans ce but, une approche générique avec un cas d'utilisation réussi.

Le reste de l'article est organisé comme suit : tout d'abord, nous présentons l'approche CEGAR ; Puis, nous proposons notre framework appelé RECAR et nous montrons sa correction et sa complétude ; Nous fournissons une implémentation de l'approche RECAR pour la logique modale K ; Et finalement, nous comparons l'efficacité de l'approche RECAR face aux solveurs de l'état de l'art pour la logique modale K.

2 Préliminaires sur CEGAR

Counter-Example Guided Abstraction Refinement, CEGAR, est une méthode incrémentale pour décider de la cohérence d'une formule.

Il a été conçu à l'origine pour la vérification du modèle [6], c'est-à-dire de répondre aux questions du type "Est-ce que $S \models P$?" ou de manière équivalente, "Est-ce que $(S \wedge \neg P)$ est incohérent ?", où S décrit un système et P une propriété. Dans les problèmes très structurés, il est souvent le cas que seul une petite partie de la formule soit nécessaire pour répondre à la question. L'idée derrière CEGAR est de remplacer $\phi = (S \wedge \neg P)$ par une abstraction ϕ', où ϕ' est plus simple à résoudre en pratique que ϕ. Il existe deux types d'abstractions :

— Une sur-abstraction de ϕ est une formule $\hat{\phi}$ telle que $\hat{\phi} \models \phi$, $\hat{\phi}$ a au plus autant de modèles que ϕ ;
— Une sous-abstraction de ϕ est une formule $\check{\phi}$ telle que $\phi \models \check{\phi}$, $\check{\phi}$ a au moins autant de modèles que ϕ.

Typiquement, ϕ est une CNF et sous/sur doivent être lu dans le sens "sous-contraint" et "sur-contraint". Ensuite, une méthode classique pour sous-approximer ϕ est "d'oublier" des clauses, c'est-à-dire $\check{\phi}$ est un sous-ensemble de clauses de ϕ. Un modèle de $\check{\phi}$ peut potentiellement, avec de la chance, satisfaire ϕ. De plus, si $\check{\phi}$ est montrée incohérent, alors ϕ l'est aussi. Cette double possibilité de conclure plus rapidement fait que les approches CEGAR basées sur les sous-abstractions sont très populaire. Une méthode classique pour sur-approximer ϕ est de borner la génération de la formule $\hat{\phi}$ en lui donnant un n plus petit que celui nécessaire pour atteindre l'équi-cohérence avec le problème original (comme dans la Vérification de Modèle Bornée (BMC) [6] ou encore en planification [27]). Ainsi un modèle de $\hat{\phi}$ peut être étendu pour devenir un modèle de ϕ mais l'incohérence de $\hat{\phi}$ signifie que la borne n a besoin d'être augmentée et la procédure doit être répétée.

Un exemple de CEGAR utilisant des sur-abstractions est donnée en Figure 1. Il reçoit en entrée une formule ϕ et calcule une sur-abstraction ψ. Ensuite, il utilise un solveur SAT pour vérifier si ψ est cohérent, si c'est le cas, il conclut que ϕ est cohérent. Sinon ψ est raffinée, c'est-à-dire, qu'elle se rapproche de ϕ, jusqu'à ce qu'elle soit cohérente ou jusqu'à ce que le raffinement de la sur-abstraction est détecté comme étant équi-cohérent à ϕ, noté $\psi \equiv^{?}_{\text{sat}} \phi$ (c-à-d. $\exists M, M \models_1 \psi$ iff $\exists M', M' \models_2 \phi$) [1] où il conclut que ϕ est incohérent. Dans la suite, $\phi \equiv^{?}_{\text{sat}} \psi$ signifie un test incomplet mais efficace pour l'équi-cohérence, test qui répond 'oui' ou 'inconnu'.

Récemment, les solveurs SAT sont devenus capable de vérifier la cohérence "sous hypothèses" [7], c'est-à-dire

1. $\models_1$ et $\models_2$ désignent éventuellement différentes relations de conséquence (en logique propositionnelle et en logique modale par exemple).

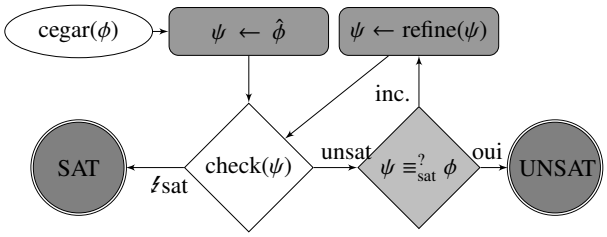

FIGURE 1 – Le framework CEGAR avec des sur-abstractions

donner la cohérence d'un ensemble de littéraux appelés hypothèses et de fournir dans le cas d'une incohérence, une "raison" de l'incohérence de la formule en fonction de ces littéraux.

Définition 1 (Coeur incohérent avec hypothèse)**.** *Soit ϕ une CNF et A un ensemble cohérent de littéraux venant de ϕ. Prenons ϕ cohérent et $(\phi \wedge \bigwedge_{a \in A} a)$ incohérent. $L \subseteq A$ est un coeur incohérent de ϕ si et seulement si $(\phi \wedge \bigwedge_{l \in L} l)$ est incohérent.*

Par conséquent, un oracle SAT pour ϕ, sachant A, peut être vu comme une procédure fournissant une paire (d, ψ) avec $d \in \{\text{SAT}, \text{UNSAT}\}$ et ψ est un modèle de ϕ si $d =$ SAT ou un coeur incohérent de ϕ si $d =$ UNSAT.

Les solveurs SAT modernes sont capables de prendre ψ en compte dans les deux cas. Les coeurs incohérents ont été utilisés par exemple dans une approche CEGAR pour décider la cohérence du fragment propositionnelle de la logique du premier ordre [16].

3 Recursive Explore and Check Abstraction Refinement

Une approche CEGAR classique avec une sur-abstraction et un court-circuit SAT fonctionne bien quand l'entrée est cohérent. Généralement, il ne fonctionne pas aussi bien sur les problèmes incohérents. La raison à cela est qu'il est nécessaire de continuer à raffiner jusqu'à obtenir l'équi-cohérence avec le problème original.

Une manière d'aborder ce problème est de mixer les court-circuits SAT et UNSAT, comme dans [4] et [29]. Dans ces approches, les méthodes alternent les sur et sous abstractions. L'approche RECAR, illustrée dans la Figure 2 et l'Algorithme 1 intercale les deux types d'abstractions : chaque abstraction est réalisée avec les informations retournées par le solveur sur la précédente. Le court-circuit UNSAT est réalisé en utilisant un appel récursif à la procédure principale quand une sous-abstraction stricte $\check{\phi}$ peut être construite. Il convient également de noter que l'approche proposée permet des abstractions sur deux niveaux : une utilisée pour simplifier le problème au niveau du domaine (l'appel récursif), tandis que l'autre permet de simplifier le problème au niveau de l'oracle.

Pour pouvoir appliquer l'approche RECAR, la sous-abstraction $\check{\phi}$ et la sur-abstraction $\hat{\phi}$ doivent satisfaire certaines conditions. Dans ce qui suit estSAT(ϕ) signifie que ϕ est cohérente ($\not\models_1 \neg\phi$) et estUNSAT(ϕ) signifie ($\models_2 \neg\phi$), mais possiblement avec différentes relations de conséquence. $RC(\phi, \check{\phi})$ désigne une fonction booléenne décidant si un appel récursif (Recursive Call) doit être effectué.

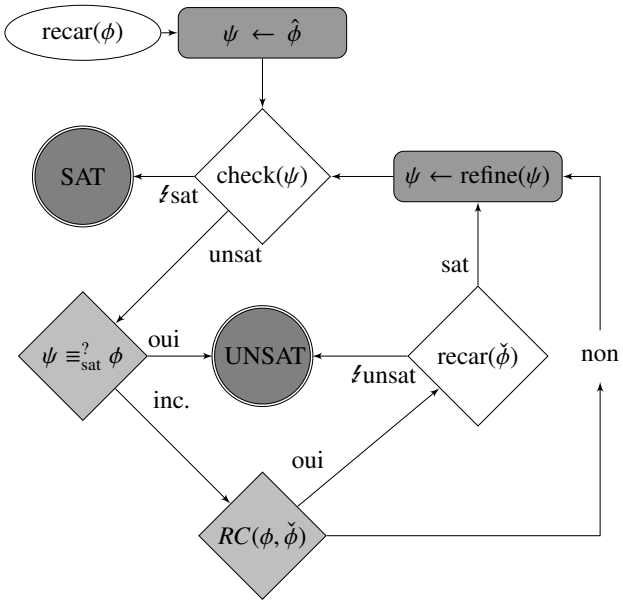

FIGURE 2 – L'approche RECAR

3.1 Conditions pour utiliser RECAR

1. La fonction 'check' est une implémentation de 'estSAT' qui est correcte, complète et qui se termine.
2. estSAT($\hat{\phi}$) implique estSAT(refine($\hat{\phi}$)).
3. Il existe $n \in \mathbb{N}$ tel que refine$^n(\hat{\phi}) \equiv^{?}_{\text{sat}} \phi$.
4. estUNSAT($\check{\phi}$) implique estUNSAT(ϕ).
5. Soit under(ϕ) $= \check{\phi}$. Il existe $n \in \mathbb{N}$ tel que $RC(\text{under}^n(\phi), \text{under}^{n+1}(\phi))$ évalue à faux.

Notons que nous avons estSAT($\hat{\phi}$) implique ϕ est cohérent avec les Conditions 2 et 3 ensemble. Dans ce qui suit, nous montrons que, sous ces conditions, l'approche RECAR est correcte, complète et termine. Pour ce faire, nous présentons l'algorithme recar(ϕ) dans l'Algorithme 1.

Théorème 1 (Justesse)**.** *Si* recar(ϕ) *retourne* SAT *alors ϕ est cohérente.*

Démonstration. Supposons que recar(ϕ) retourne SAT. Cela arrive seulement si check(ψ) retourne SAT, soit à la ligne 3 soit à la ligne 7. Ainsi, nous savons que estSAT(ψ) est vérifiée (Condition 1) mais ψ vaut $\hat{\phi}$ ou refine$^n(\phi)$ pour

Algorithme 1 : recar(ϕ)

1 $\psi \leftarrow \hat{\phi}$
2 **tant que** $\psi \equiv^{?}_{\text{sat}} \phi$ *retourne "inconnu"* **faire**
3 **si** check(ψ) = SAT **alors retourner** SAT ;
4 **si** $RC(\phi, \check{\phi})$ **alors**
5 **si** recar($\check{\phi}$) = UNSAT **alors retourner** UNSAT ;
6 $\psi \leftarrow$ refine(ψ) ;
7 **retourner** check(ψ)

un certain $n \in \mathbb{N}$. Par conséquent, ϕ est cohérente (par les Conditions 2 et 3). □

L'intuition derrière la preuve du Théorème 2 est qu'il y a deux manières de conclure que ϕ est incohérente. Dans le premier cas, $\hat{\phi}$ est raffinée un nombre fini de fois jusqu'à ce que l'on détecte l'équi-cohérence avec ϕ et que check retourne UNSAT et donc que ϕ est incohérente. Dans le second cas, une des sous-abstractions est montrée UNSAT et donc par construction ϕ est UNSAT (Condition 4).

Théorème 2 (Complétude)**.** *Si* recar(ϕ) *retourne* UNSAT *alors* estUNSAT(ϕ).

Démonstration. Par induction sur la taille k de la pile d'appels récursifs à recar (ligne 5). Supposons que recar(ϕ) retourne UNSAT après k appels récursifs. Dans la base d'induction ($k = 0$) (pas d'appel récursif). Nous devons avoir quitté la boucle ($\psi \equiv^{?}_{\text{sat}} \phi$) et check($\psi$) a retourné UNSAT. Cela signifie que ψ est incohérent (par la Condition 1) et par conséquent estUNSAT(ϕ) est vraie (à cause de l'équi-cohérence). L'hypothèse d'induction est la suivante : pour tous $k \leq n$, si recar(ϕ) retourne UNSAT après k appel récursif alors estUNSAT(ϕ). Dans l'étape d'induction $k = n + 1$. Les conditions des lignes 4 et 5 de l'algorithme sont vraies. Cela signifie que recar($\check{\phi}$) retourne UNSAT après k appels récursifs à recar. estUNSAT($\check{\phi}$) (par l'hypothèse d'induction). Donc estUNSAT(ϕ) (par la Condition 4). □

L'intuition derrière la preuve du Théorème 3 est que l'algorithme va effectuer un nombre fini d'appels récursifs (Condition 5). De plus, chacun de ces appels récursifs a un nombre fini d'étapes de raffinement avant de terminer (Condition 3).

Théorème 3 (Terminaison)**.** RECAR *termine pour n'importe quelle entrée ϕ.*

Démonstration. Nous avons que (1) Pour toutes les formules ϕ, il existe $n \in \mathbb{N}$ tel que $RC(\text{under}^n(\phi), \text{under}^{n+1}(\phi))$ évalue à faux (Condition. 5) et (2) Pour chaque $i \leq n$ il existe $m_i \in \mathbb{N}$ tel que $\text{refine}^{m_i}(\hat{\phi}) \equiv^{?}_{\text{sat}} \phi$ (Condition. 3). Par conséquent, pour chaque entrée ϕ, l'appel récursif à la ligne 5 de l'Algorithme va s'exécuter au plus n fois avant que la condition à la ligne 4 devienne fausse. Pour chacun de ces appels récursifs, la boucle tant-que de l'algorithme va être effectuée au plus m_i fois avant que la condition à la ligne 2 devienne fausse. Donc, pour toute les entrées, recar se termine après au plus $n + \Sigma_{i=0}^{n}(m_i)$ appels récursifs. □

4 Résoudre K-SAT avec RECAR

Dans cette section, nous montrons comment RECAR est utilisé pour résoudre le problème de satisfiabilité en logique modale K, qui est un problème PSPACE-complet [11, 18].

4.1 La logique modale K

Avant de montrer comment nous appliquons l'approche RECAR, définissons formellement ce qu'est la logique modale K. Soit un ensemble non-vide fini de variables propositionnelles $\mathbb{P} = \{p_1, p_2, \ldots\}$ et un ensemble de m opérateurs modaux unaires $\mathbb{M} = \{\Box_1, \ldots, \Box_m\}$. Le langage de K (noté $\mathcal{L}$) est l'ensemble des formules contenant $\mathbb{P}$, fermé sous l'ensemble des connecteurs propositionnels $\{\neg, \wedge\}$ et l'ensemble des opérateurs modaux dans $\mathbb{M}$. Nous utiliserons les abréviations standard pour $\top$, $\bot$, $\vee$ et $\Diamond$. Par exemple $\Diamond_a \phi = \neg\Box_a\neg\phi$.

Définition 2 (Modèle de Kripke)**.** *un modèle de Kripke est un triplet $M = \langle W, \{R_a \mid \Box_a \in \mathbb{M}\}, V\rangle$, où : W est un ensemble non-vide de "mondes possibles" ; Chaque $R_a \subseteq W \times W$ est une relation d'accessibilité binaire sur W ; $V : \mathbb{P} \rightarrow 2^W$ est une fonction de valuation qui associe, à chaque $p \in \mathbb{P}$, l'ensemble des mondes possibles de W où p est vraie. Un modèle de Kripke pointé est une paire $\langle M, w\rangle$, où M est un modèle de Kripke et w est un monde possible dans W.*

Par la suite, chaque fois que nous utilisons le terme 'modèle', nous nous référons au 'modèle de Kripke pointé'

Dans ce qui suit, la taille d'un modèle $\langle M, w\rangle$ est le nombre d'éléments dans W, noté $|M|$.

Définition 3 (Relation de satisfiabilité)**.** *La relation $\models$ entre les modèles et les formules est définie récursivement de la façon suivante :*

$\langle M, w\rangle \models p$ *ssi* $w \in V(p)$
$\langle M, w\rangle \models \neg\phi$ *ssi* $\langle M, w\rangle \not\models \phi$
$\langle M, w\rangle \models \phi_1 \wedge \phi_2$ *ssi* $\langle M, w\rangle \models \phi_1$ *et* $\langle M, w\rangle \models \phi_2$
$\langle M, w\rangle \models \Box_a\phi$ *ssi* $(w, w') \in R_a$ *implique* $\langle M, w'\rangle \models \phi$

Définition 4 (Validité)**.** *Comme d'habitude, une formule $\phi \in \mathcal{L}$ est valide (noté $\models \phi$) si et seulement si elle est satisfaite par tous les modèles $\langle M, w\rangle$. Une formule $\phi \in \mathcal{L}$ est satisfiable en K (noté* estKSAT(ϕ)*) si et seulement si $\not\models \neg\phi$. Nous utilisons aussi* estKUNSAT(ϕ) *pour signifier $\models \neg\phi$.*

5 Les fonctions d'abstractions

Dans la suite, nous définissons une traduction de la logique modale K à la logique propositionnelle classique.

Définition 5 (Traduction)**.**

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

La traduction ajoute de nouvelles variables p_i et $r^a_{i,j}$ à la formule. p_i signifie que la variable p est vraie dans le monde w_i alors que $r^a_{i,j}$ signifie que w_j est accessible depuis w_i par la relation a.

Théorème 4. estKSAT(ϕ) *si et seulement si* estSAT(tr($\phi, nm(\phi) + 1$)).

Démonstration. Le résultat présenté dans le Théorème 4 provient directement de [25], §25.2.2 (où $nm(\phi)$ est le nombre d'opérateurs modaux dans la formule ϕ). □

Pour décider la cohérence de la formule $\phi \in \mathcal{L}$, on peut simplement alimenter le solveur SAT avec tr($\phi, nm(\phi) + 1$). En fait, c'est ce qui est globalement proposé dans Km2SAT [26].

Le principal problème avec cela est que la traduction peut générer une formule exponentiellement plus grande. Dans cet article, nous essayons de contourner ce problème en utilisant RECAR. Pour des raisons de simplicité, nous utiliserons maintenant tr(ϕ) pour signifier tr($\phi, nm(\phi) + 1$).

Dans [5], les auteurs utilisent aussi ce type de traduction pour la logique modale S5 vers la logique propositionnelle, mais en remplaçant $nm(\phi)$ par le diamond-degree dd(ϕ) qui est généralement plus petit. Malheureusement, ceci ne peut pas être reproduit pour la logique modale K, voici un contre-exemple :

Contre-Exemple 1. *Soit* $\phi = (p_1 \wedge p_2 \wedge p_3) \wedge (\Diamond_a(p_1 \wedge p_2 \wedge \neg p_3 \wedge \Box_a(p_1 \wedge \neg p_2 \wedge p_3))) \wedge (\Diamond_a(p_1 \wedge \neg p_2 \wedge \neg p_3 \wedge \Box_a(\neg p_1 \wedge \neg p_2 \wedge p_3))) \wedge (\Box_a \Diamond_a p_3)$*, avec* dd($\phi$) = 3. *Cette formule est satisfaite par le modèle illustré dans la Figure 3. Cependant, il est facile de voir qu'il n'existe pas de modèle satisfiant ϕ avec moins de 5 mondes possibles.*

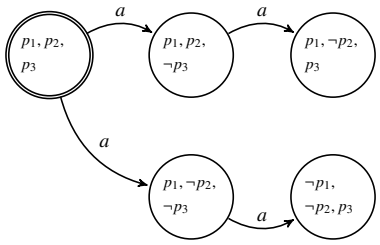

FIGURE 3 – $M \models \phi$

5.1 Sur-abstraction

Maintenant, dans le but d'appliquer le framework RECAR, nous avons besoin de trouver une sur-abstraction qui respecte les conditions présentées dans la Section 3.

Définition 6 (Sur-abstraction)**.** *Soit* $\phi \in \mathcal{L}$*. La sur-abstraction de* ϕ*, noté* $\hat{\phi}$*, est la formule* tr(ϕ, 1).

Définition 7 (Raffinement)**.** *Soit* $1 \leq n \leq nm(\phi) + 1$*. Le raffinement de* tr(ϕ, n)*, noté* refine(tr(ϕ, n)) *est la formule* tr($\phi, n + 1$).

Théorème 5. *Si* estSAT(tr(ϕ, n)) *alors* estSAT(tr($\phi, n+1$))*, pour tous* $1 < n \leq nm(\phi) + 1$*. (*RECAR *Condition. 2)*

Esquisse de preuve. L'idée est que si ϕ est satisfaite par un modèle M avec n mondes, alors nous pouvons trouver un modèle M' avec $n + 1$ mondes satisfaisant ϕ. Le monde additionnel n'est juste pas accessible depuis les mondes déjà dans M. □

Ce dernier résultat nous permet d'utiliser cette sur-abstraction et cette fonction de raffinement dans l'approche RECAR. Il est facile de voir que les Conditions 2 et 3 de RECAR sont satisfaites.

5.2 Sous-abstraction

Pour faire comprendre l'intuition derrière la fonction de sous-abstraction, nous utilisons un exemple. Soit $\phi = (\Diamond p \wedge \Box\neg p \wedge \chi)$ pour un $\chi \in \mathcal{L}$, où $nm(\chi)$ est grand. ϕ est clairement incohérente car $(\Diamond p \wedge \Box\neg p)$ est incohérente. N'importe qui peut le voir directement sans même savoir à quoi χ ressemble. Cependant, une approche CEGAR utilisant la sur-abstraction et le raffinement définis plus tôt va prendre un temps considérable avant d'être capable de conclure à l'incohérence. La raison étant que chaque raffinement tr($\phi, n + 1$) de la formule originale va être montré incohérente et la procédure ne va s'arrêter que lorsque la borne $nm(\phi) + 1$ sera atteinte.

Pour éviter ces cas pathologiques, l'approche RECAR effectue aussi des sous-abstractions. Pour voir comment cela fonctionne, reprenons notre exemple ϕ. Tout d'abord, nous ajoutons à chaque conjonction dans ϕ, une nouvelle variable s_i (un sélecteur) qui va être supposée vraie par le

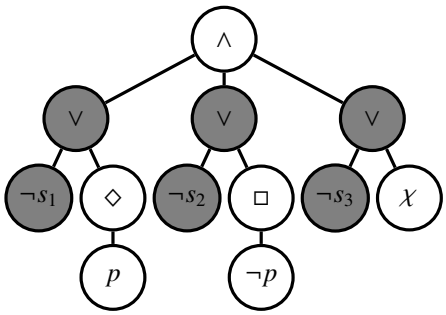

FIGURE 4 – Comment les sélecteurs sont appliqués à ϕ

solveur SAT, comme illustré par la Figure 4. Ensuite, nous construisons notre première sur-abstraction $\text{tr}(\phi, 1)$ et nous alimentons le solveur SAT avec. Le solveur va répondre incohérent en fournissant un coeur incohérent exprimé avec les s_i. De ce coeur incohérent, nous pouvons donc extraire un ensemble de sélecteurs *core*. Supposons que, dans notre exemple, $core = \{s_1, s_2\}$. Cela signifie que la formule $\check{\phi} = (\Diamond p \wedge \Box \neg p)$, étiquetée par les sélecteurs est suffisante pour montrer l'incohérence de ϕ avec seulement 1 monde possible. Prouver l'incohérence de $\check{\phi}$ va impliquer que ϕ est incohérente. Il est à noter que, dans ce cas, $nm(\check{\phi})$ est bien plus petit que $nm(\phi)$. Ainsi, l'approche `CEGAR` appliquée à $\check{\phi}$ va répondre bien plus rapidement, alors qu'elle aurait probablement échouée sur l'ensemble de la formule ϕ. Formellement, nous avons :

Définition 8 (Sous-Abstraction)**.**

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

Théorème 6. $\text{estKUNSAT}(\text{under}(\phi, core))$ *implique* $\text{estKUNSAT}(\phi)$.

L'intuition de la preuve qui suit est que chaque sélecteur s_i active une branche dans une conjonction de la formule. A chaque fois que la fonction 'under' est appelée avec un *core* non-vide, les branches non-activées avec un sélecteur venant de *core* vont être supprimées de la formule.

Démonstration. Soit ϕ une formule en NNF. Nous montrons que $\text{estKSAT}(\phi)$ implique $\text{estKSAT}(\text{under}(\phi, core))$ par induction sur la structure de ϕ. Supposons $\text{estKSAT}(\phi)$. Alors $\exists\, M, w$ t.q. $\langle M, w\rangle \models \phi$. Il y a deux cas possible dans la base d'induction : (1) $\phi = p$ et (2) $\phi = \neg p$. Dans ces deux cas, $\text{under}(\phi, core) = \phi$. Il y a quatre cas dans les étapes d'induction :
(1) $\phi = \Diamond_a(\psi)$. $\exists\, M, w$ t.q. $\langle M, w\rangle \models \Diamond_a(\psi)$. Alors $\exists\, M, w'$ t.q. $(w, w') \in R$, $\langle M, w'\rangle \models \psi$. Alors $\langle M, w'\rangle \models \text{under}(\psi, core)$ par hypothèse d'induction. Ainsi $\langle M, w\rangle \models \text{under}(\phi, core)$;
(2) $\phi = \Box_a(\psi)$. Ce cas est similaire à (1).
(3) $\phi = (\psi \wedge \chi)$. $\exists\, \langle M, w\rangle \models (\psi \wedge \chi)$. Alors $\langle M, w\rangle \models \psi$ et $\langle M, w\rangle \models \chi$. Alors $\langle M, w\rangle \models \text{under}(\psi, core)$ et $\langle M, w\rangle \models \text{under}(\chi, core)$ par hypothèse d'induction. Ainsi $\langle M, w\rangle \models \text{under}(\phi, core)$;
(4) $\phi = (\psi \vee \chi)$. Nous considérons trois cas :
(4.a) $\psi = \neg s_i$ et $s_i \in core$. Alors $\exists\, \langle M, w\rangle \models (\neg s_i \vee \chi)$ mais $s_i \in V(w)$, alors $\langle M, w\rangle \models \chi$. Alors $\langle M, w\rangle \models \text{under}(\chi, core)$ par hypothèse d'induction. Ainsi $\langle M, w\rangle \models \text{under}(\phi, core)$;
(4.b) $\psi = \neg s_i$ et $s_i \notin core$. $\exists\, \langle M, w\rangle \models (\neg s_i \vee \chi)$. mais nous avons toujours $\langle M, w\rangle \models \top$. Ainsi $\langle M, w\rangle \models \text{under}(\phi, core)$.
(4.c) Ce cas est similaire à (3). □

Le Théorème 6 montre que la fonction 'under' satisfait la Condition 4 de `RECAR`. Pour montrer qu'elle satisfait aussi la Condition 5, notons que la taille de $\text{under}^{n+1}(\phi, core)$ est plus petit ou égale à celle de $\text{under}^{n}(\phi, core')$ (même si généralement les ensembles *core* et $core'$ sont différents).

6 MoSaiC : RECAR pour K-SAT

Nous avons implémenté l'approche `RECAR` pour le problème de cohérence en logique modale dans le solveur MoSaiC, en utilisant les fonctions de sur et sous abstractions définies dans les sections précédentes. MoSaiC possède différentes fonctionnalités trouvées dans les solveurs de l'état-de-l'art. Comme dans Km2SAT [26], il simplifie la formule en entrée en effectuant différentes règles : le Box Lifting, le Flattening et la règle de Truth Propagation sur les opérateurs booléens et modaux (voir [26] pour plus d'informations). MoSaiC utilise aussi le solveur Glucose en mode incrémental [7, 1] pour décider la cohérence de chaque sur-abstraction (ψ).

Notons que l'implémentation diffère légèrement de la Figure 5. Par exemple, nous n'appelons pas Glucose sur ψ mais sur ψ sur laquelle nous avons appliqué les sélecteurs ; nous n'avons pas besoin de générer la sous-abstraction $\check{\phi}$ pour tester la condition ($\check{\phi} = \phi$) : il suffit de connaître le nombre de sélecteurs impliqués dans l'incohérence de la formule. Nous retournons aussi un modèle de Kripke dans la procédure principale, pas simplement SAT/UNSAT.

Nous tirons avantage d'une telle information pour fournir une nouvelle borne l. Et finalement, notons que dans

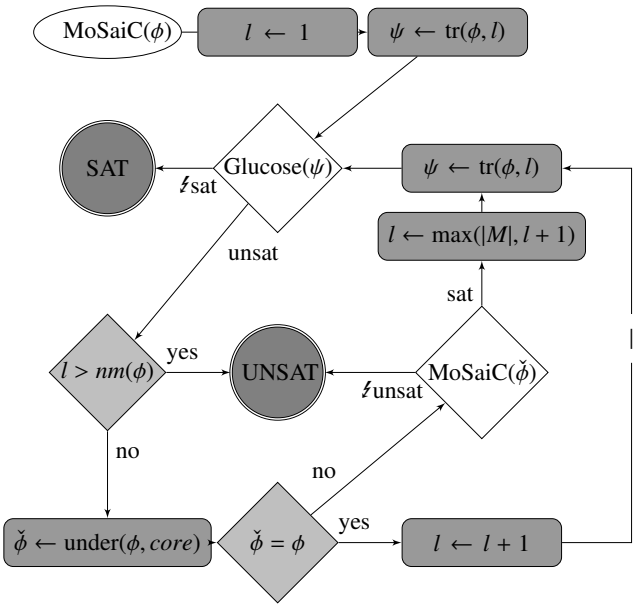

FIGURE 5 – MoSaiC : RECAR pour la logique modale K

notre cas $\max(|M|, l+1)$ retourne toujours $|M|$ car il n'est pas possible de trouver un plus petit modèle de Kripke que M par construction de $\check{\phi}$. L'approche CEGAR du solveur présentée dans la Section 7 utilise un schéma similaire, mais sans l'étape récursif de RECAR.

7 Expérimentations

Nous comparons notre approche aux solveurs de l'état de l'art considérés dans [21], c'est-à-dire à : $K_S P$ 0.1 [21], BDDTab 1.0 [9], FaCT++ 1.6.4 [28], InKreSAT 1.0 [15], *SAT 1.3 [8], Km2SAT 1.0 [26] combiné avec le même Glucose que dans MoSaiC, Spartacus 1.0 [10] et une combinaison de l'Optimized Functional Translation [12] et Vampire 4.0 [17]. MoSaiC est configuré comme une approche CEGAR classique et comme une approche RECAR. Nous avons choisi d'exécuter ces solveurs sur les benchmarks suivant : L'ensemble complet des formules "TANCS-2000 modalised QBF (MQBF)" [20] complété avec les formules MQBF fournit dans [15], 1016 formules, 617 cohérents, 399 incohérents ; Les formules LWB [2], avec 56 formules choisit parmi les 18 classes, générées par le script fournit par les auteurs de [21], 1008 formules, 504 cohérents, 504 incohérents ; Les formules $3CNF_{KSP}$ générées aléatoirement [22] de profondeur modale égale à 1 ou 2, 1000 formules, 457 cohérents, 464 incohérents. Peu des solveurs considérés peuvent gérer des modalités multiples comme MoSaiC, de plus, à notre connaissance, il n'existe pas de benchmarks pour la logique modale K contenant plusieurs modalités. Nous avons utilisé une limite de mémoire de 32 Go et une limite de temps de 900

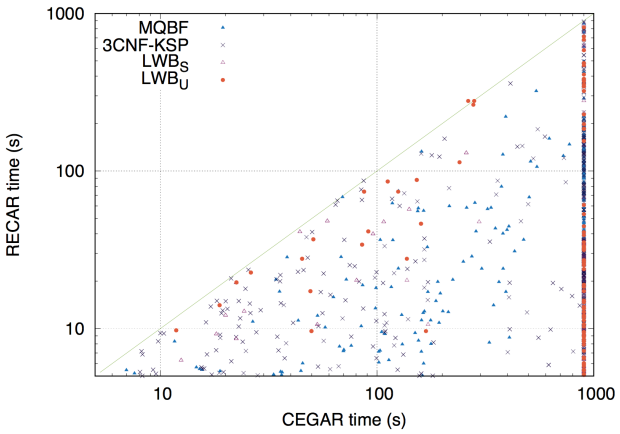

FIGURE 6 – Scatter-plot CEGAR contre RECAR

secondes par solveur par benchmark. Notons que pour des soucis d'espace, les résultats présentés ici sont globaux [2]. Le comportement des solveurs varie beaucoup en fonction des familles de benchmarks. Nous croyons cependant que ces résultats fournissent une vision intéressante des capacités de l'approche proposée.

7.1 RECAR contre CEGAR

Nous pouvons voir sur la Figure 6 que dans la plus grande partie des benchmarks, l'approche RECAR surpasse l'approche CEGAR. La sous-abstraction fournit souvent une formule avec une borne plus petite, ce qui produit des CNF d'une taille raisonnable que le solveur SAT peut gérer. Notons que sur cette Figure, les memory-out des approches CEGAR et RECAR sont affichés comme des time-out, c'est-à-dire des points à 900 secondes. La différence dans les résultats vient majoritairement de la capacité de RECAR de résoudre des benchmarks incohérents : (1118/1367) benchmarks résolus par RECAR contre (155/1367) benchmarks résolus par CEGAR. Pour les benchmarks cohérents, l'amélioration de la borne venant de la sortie d'un appel récursif permet d'atteindre plus rapidement une sur-abstraction cohérent.

7.2 Comparaison avec les solveurs de l'état-de-l'art

Nous pouvons voir sur la Figure 7 que l'approche CEGAR utilisant notre sur-abstraction est le pire solveur alors que notre approche RECAR surpasse les autres solveurs. Km2SAT effectue un raisonnement spécifique capable de détecter rapidement certains des benchmarks incohérents sans avoir besoin de générer la CNF complètement, ce qui explique pourquoi il réussi mieux que notre approche CEGAR. *SAT alterne des raisonnements au niveau propositionnelle et au niveau modale, il peut donc être considéré

2. Des résultats additionnels peuvent être trouvés à l'adresse suivante : http://www.cril.univ-artois.fr/~montmirail/mosaic

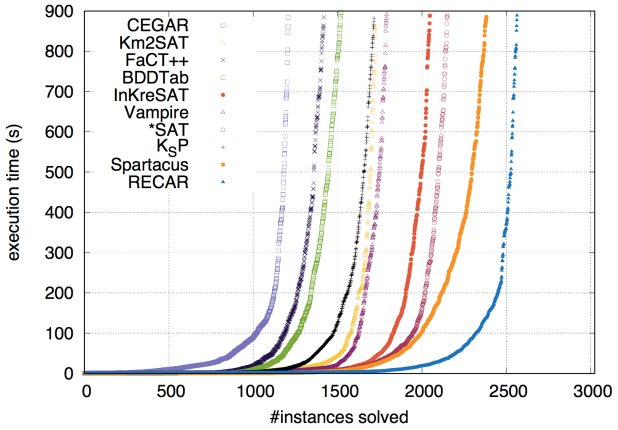

FIGURE 7 – Distribution des temps de résolutions

comme une approche CEGAR utilisant une sous-abstraction. Il obtient de bons résultats, malgré le fait qu'il est lié à un vieux solveur, SATO [3]. Notre meilleur compétiteur, Spartacus, est basé sur une méthode des tableaux, pas sur une approche utilisant un solveur SAT : les approches basées sur SAT n'étaient pas les meilleurs approches pour attaquer des problèmes de cohérence en logique modale K jusqu'à présent. Spartacus atteint la plus part du temps la limite de temps sur ses benchmarks non-résolus alors que nous épuisons la mémoire disponible : les solveurs se comportent différemment et ont différentes limites.

8 Conclusion

Nous avons proposé ici une nouvelle approche pour résoudre des problèmes de décision en utilisant une approche basée sur du raffinement d'abstractions récursives. Nous avons montré la correction et la complétude de cette approche et nous l'avons instancié pour le problème de la cohérence d'une formule en logique modale K. Nous avons comparé notre approche face aux solveurs représentant, à notre connaissance, l'état-de-l'art pour la résolution pratique du problème K-SAT, sur une large gamme de benchmarks de logique modale K. Une simple approche CEGAR utilisant une sur-abstraction n'est pas compétitive du tout, due aux nombreux benchmarks disponibles qui sont incohérents. Notre approche RECAR, mixant les court-circuits SAT et UNSAT, surpasse les autres solveurs sur les benchmarks considérés. Ces résultats prometteurs sont une première étape vers des solveurs de logique modale plus efficaces : MoSaiC peut être étendu à d'autres logiques modales tel que KT, S4, S5 et KD45, en adaptant la traduction vers la logique propositionnelle. Nous croyons aussi que RECAR est une approche prometteuse pour attaquer des problèmes de décision au dessus de NP dans la hiérarchie polynomiale. Il n'est pas encore clair pour nous si les approches CEGAR existantes pour QBF telles que [13] peuvent être exprimées dans notre approche. C'est un futur travail passionnant.

3. *SAT est profondément lié à SATO, ce qui rend très difficile une mise-à-jour avec un solveur SAT plus récent

Remerciements

Les auteurs remercient les relecteurs d'IJCAI et de JIAF pour leurs remarques éclairées. Une partie de ce travail est supportée par l'ANR SATAS (ANR-15-CE40-0017), le Ministère de l'Enseignement supérieur, de la Recherche et de l'Innovation et le conseil régional des Hauts-De-France à travers le "Contrat de Plan État Région (CPER)" et les financements FEDER.

Remarques

Cet article sera présenté à IJCAI 2017 [19]

Références

[1] Audemard, Gilles, Jean-Marie Lagniez et Laurent Simon: *Improving glucose for incremental SAT solving with assumptions: Application to MUS extraction*. Dans *Proc. of SAT'13*, pages 309–317, 2013. http://dx.doi.org/10.1007/978-3-642-39071-5_23.

[2] Balsiger, Peter, Alain Heuerding et Stefan Schwendimann: *A Benchmark Method for the Propositional Modal Logics K, KT, S4*. J. Autom. Reasoning, 24(3):297–317, 2000. http://dx.doi.org/10.1023/A:1006249507577.

[3] Biere, Armin, Marijn Heule, Hans van Maaren et Toby Walsh (rédacteurs): *Handbook of Satisfiability*, tome 185 de *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009, ISBN 978-1-58603-929-5.

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

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

[6] Clarke, Edmund M., Orna Grumberg, Somesh Jha, Yuan Lu et Helmut Veith: *Counterexample-guided abstraction refinement for symbolic model checking*. J. ACM, 50(5):752–794, 2003. http://doi.acm.org/10.1145/876638.876643.

[7] Eén, Niklas et Niklas Sörensson: *An Extensible SAT-solver*. Dans *Proc. of SAT'03*, pages 502–518, 2003. http://dx.doi.org/10.1007/978-3-540-24605-3_37.

[8] Giunchiglia, Enrico, Armando Tacchella et Fausto Giunchiglia: *SAT-Based Decision Procedures for Classical Modal Logics*. J. Autom. Reasoning, 28(2):143–171, 2002. http://dx.doi.org/10.1023/A:1015071400913.

[9] Goré, Rajeev, Kerry Olesen et Jimmy Thomson: *Implementing Tableau Calculi Using BDDs: BDDTab System Description*. Dans *Proc. of IJCAR'14*, pages 337–343, 2014. http://dx.doi.org/10.1007/978-3-319-08587-6_25.

[10] Götzmann, Daniel, Mark Kaminski et Gert Smolka: *Spartacus: A Tableau Prover for Hybrid Logic*. Electr. Notes Theor. Comput. Sci., 262:127–139, 2010. http://dx.doi.org/10.1016/j.entcs.2010.04.010.

[11] Halpern, Joseph Y. et Yoram Moses: *A Guide to Completeness and Complexity for Modal Logics of Knowledge and Belief*. Artif. Intell., 54(2):319–379, 1992. http://dx.doi.org/10.1016/0004-3702(92)90049-4.

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

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

[14] Järvisalo, Matti, Daniel Le Berre, Olivier Roussel et Laurent Simon: *The international SAT solver competitions*. AI Magazine, 33(1), 2012. http://www.aaai.org/ojs/index.php/aimagazine/article/view/2395.

[15] Kaminski, Mark et Tobias Tebbi: *InKreSAT: Modal Reasoning via Incremental Reduction to SAT*. Dans *Proc. of CADE'13*, pages 436–442, 2013. http://dx.doi.org/10.1007/978-3-642-38574-2_31.

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

[17] Kovács, Laura et Andrei Voronkov: *First-Order Theorem Proving and Vampire*. Dans *Proc. of CAV'13*, pages 1–35, 2013. http://dx.doi.org/10.1007/978-3-642-39799-8_1.

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

[19] Lagniez, Jean Marie, Daniel Le Berre, Tiago de Lima et Valentin Montmirail: *A Recursive Shortcut for CEGAR: Application To The Modal Logic K Satisfiability Problem*. Dans *Proc. of IJCAI'17*, 2017.

[20] Massacci, Fabio et Francesco M. Donini: *Design and results of TANCS-2000 non-classical (modal) systems comparison*. Dans *Proc. of TABLEAUX'00*, pages 52–56, 2000. http://dx.doi.org/10.1007/10722086_4.

[21] Nalon, Cláudia, Ullrich Hustadt et Clare Dixon: *$K_S P$: A Resolution-Based Prover for Multimodal K*. Dans *Proc. of IJCAR'16*, pages 406–415, 2016. http://dx.doi.org/10.1007/978-3-319-40229-1_28.

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

[23] Prestwich, Steven David: *CNF encodings*. Dans *Handbook of Satisfiability*, pages 75–97. IOS Press, 2009. http://dx.doi.org/10.3233/978-1-58603-929-5-75.

[24] Pulina, Luca: *The ninth QBF solvers evaluation - preliminary report*. Dans *Proc. of the 4th International Workshop (QBF 2016) co-located with (SAT 2016)*, pages 1–13, 2016. http://ceur-ws.org/Vol-1719/paper0.pdf.

[25] Sebastiani, Roberto et Armando Tacchella: *SAT techniques for modal and description logics*. Dans *Handbook of Satisfiability*, pages 781–824. IOS Press, 2009. http://dx.doi.org/10.3233/978-1-58603-929-5-781.

[26] Sebastiani, Roberto et 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.

[27] Seipp, Jendrik et Malte Helmert: *Counterexample-guided cartesian abstraction refinement*. Dans *Proc. of ICAPS'13*, 2013. http://www.aaai.org/ocs/index.php/ICAPS/ICAPS13/paper/view/6034.

[28] Tsarkov, Dmitry et Ian Horrocks: *FaCT++ Description Logic Reasoner: System Description*. Dans *Proc. of IJCAR'06*, pages 292–297, 2006. http://dx.doi.org/10.1007/11814771_26.

[29] Wang, Chao, Aarti Gupta et Franjo Ivancic: *Induction in CEGAR for detecting counterexamples*. Dans *Proc. of FMCAD'07*, pages 77–84, 2007. http://dx.doi.org/10.1109/FAMCAD.2007.21.