Une approche SAT sensible à la mémoire pour les logiques modales PSPACE

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

Actes JIAF 2019

Une approche SAT sensible à la mémoire pour les logiques modales PSPACE

Jean-Marie Lagniez[1], Daniel Le Berre[1], Tiago de Lima[1] et Valentin Montmirail[2]

[1]CRIL, Univ Artois et CNRS, France
[2]I3S, Univ Côte-d'Azur et CNRS, France
{lagniez,leberre,delima}@cril.fr vmontmirail@i3s.unice.fr

Résumé

Le solveur SAT est devenu un oracle NP efficace pour résoudre des problèmes NP-complet (voir au-delà). En général, ces problèmes sont résolus soit par une traduction directe vers SAT soit en résolvant itérativement des problèmes SAT dans une procédure comme CEGAR. Récemment, une nouvelle boucle CEGAR récursive travaillant sur deux niveaux d'abstractions, appelée RECAR, a été proposée et instanciée pour la logique modale K. Nous visons à compléter ce travail pour les logiques modales utilisant les axiomes (B), (D), (T), (4) et (5). De plus, pour le rendre performant en pratique, nous généralisons le framework RECAR pour utiliser des compositions de fonctions d'abstraction. Les résultats expérimentaux montrent l'efficacité de cette approche et qu'elle surpasse les solveurs de l'état de l'art pour les logiques modales K, KT et S4 sur les benchmarks considérés.

Abstract

SAT technology has become an efficient running NP-oracle for solving NP-complete problems (and beyond). Usually those problems are solved by direct translation to SAT or by solving iteratively SAT problems in a procedure like CEGAR. Recently, a new recursive CEGAR loop working with two abstraction levels, called RECAR, was proposed and instantiated for modal logic K. We aim to complete this work for modal logics based on axioms (B), (D), (T), (4) and (5). Moreover, to make it efficient in practice, we generalize the RECAR framework to deal with compositions of abstraction functions. Experimental results show that the approach is efficient and outperforms state-of-the-art modal logic solvers for modal logics K, KT and S4 on considered benchmarks.

1 Introduction

La technologie SAT s'est avérée être une approche pratique très efficace pour résoudre des problèmes NP-complet [4]. L'un des principaux problèmes est de trouver le "bon" encodage pour le problème, c'est-à-dire, de trouver une réduction polynomiale du problème original vers une formule de la logique propositionnelle en forme normal conjonctive (CNF) qui peut être efficacement résolue par un solveur SAT [24]. En raison de leur efficacité, les solveurs SAT sont utilisés pour résoudre des problèmes au-delà de NP tels que la planification [25], la logique modale K [31] et QBF [14, 28] qui sont tous des problèmes PSPACE-complets ou encore le comptage de modèles en logique propositionnelle qui est un problème #P-complet et où les meilleures approches utilisent un solveur SAT [20, 23, 29, 33].

Malheureusement, en raison de la taille de la traduction, il n'est généralement pas possible de les encoder directement dans CNF. Pour pallier à ce problème, des procédures plus complexes utilisant des solveurs SAT comme oracles ont été conçues. Un exemple d'une telle procédure est la procédure CEGAR (*Counter-Example-Guided Abstraction Refinement*) [7]. Cette procédure peut être instanciée de deux manières différentes, à savoir CEGAR-over et CEGAR-under. CEGAR-over (resp. CEGAR-under) utilise des sur-abstractions (resp. des sous-abstractions). L'oracle est alimenté par une abstraction du problème original permettant moins (resp. plus) de modèles. S'il trouve un modèle (resp. s'il prouve qu'il n'y a pas de modèle), le problème d'origine est résolu. Sinon une nouvelle abstraction doit être définie, tenant compte de la réponse de l'oracle.

Récemment, Lagniez et al. [18] ont proposé une version récursive de CEGAR dans une procédure appelée RECAR (*Recursive Explore and Check Abstraction Refinement*). Ils ont instancié leur framework pour la logique modale K et ont démontré expérimentalement les avantages de basculer entre les deux types d'abstractions de manière récursive, en particulier pour repérer de petites sous-formules insatisfiables. Cependant, certaines applications pratiques nécessitent des logiques modales différentes de K, comme

KT en économie [21] ou S4 dans le domaine des logiques de descriptions [34]. Même si ces logiques modales sont PSPACE-complètes, en pratique, les réductions ne sont pas simples et des informations structurelles peuvent être perdues pendant la traduction. Afin d'étendre la portée de leur framework à d'autres logiques modales, nous proposons d'exploiter la correspondance entre les axiomes de la logique modale et les contraintes structurelles, mises en évidence dans la litterature [27], en codant les axiomes de la logique modale (D), (B), (4) et (5) en CNF. Pour ce faire, nous complétons la fonction de sur-abstraction initiale en ajoutant, pour chaque axiome, une contrainte structurelle qui force la structure de Kripke en construction à satisfaire cet axiome.

La taille de la CNF produite par [18] dépend fortement du nombre de mondes considérés pour la structure de Kripke en construction. Même si ce nombre est théoriquement exponentiel dans la taille de la formule d'entrée, ils ont démontré expérimentalement que le nombre de mondes nécessaires sur les instances considérées était souvent suffisamment petit pour faire qu'une approche basée sur SAT soit efficace en pratique pour la logique modale K. Quand ce n'est pas le cas, considérer judicieusement les sous-parties de la formule peut aider à décider de l'insatisfiabilité.

Cependant, comme démontré dans nos expériences, ceci n'est pas nécessairement vrai lorsque l'on considère des instances spécifiques dans d'autres logiques modales. Pour surmonter cette difficulté, nous proposons donc une généralisation des principes de RECAR. L'idée est de réduire agressivement la formule originale tout en préservant la satisfiabilité. Une telle idée a déjà été développée dans [18] lors de la suppression de certaines conjonctions de la formule d'entrée. Ici, nous allons encore plus loin en supprimant les disjonctions et/ou les modalités, en fonction de la logique modale considérée, en considérant des sous-parties de la formule originale à la fois pour la sur-abstraction et la sous-abstraction (seul la sous-abstraction a été considérée dans les travaux antérieurs).

Nous présentons également des simplifications supplémentaires, spécifiques aux logiques, pour les chaînes de modalités préservant la satisfiabilité. Et enfin, nous évaluons expérimentalement l'approche proposée pour les logiques modales K, KT et S4.

2 Préliminaires

Avant de passer à la logique modale et à son axiomatisation, rappelons quelques concepts fondamentaux de la logique propositionnelle.

Définition 1 (Langage de la logique propositionnelle)**.** *Soit $\mathbb{P} = \{p_0, \ldots, p_{n-1}\}$ un ensemble fini et non-vide de n variables propositionnelles. Le langage de la logique propositionnelle (noté CPL) est l'ensemble des formules contenant $\mathbb{P}$ et fermé sous l'ensemble des connecteurs propositionnels $\{\neg, \wedge\}$. Nous utilisons également les abréviations standard pour $\top$, $\bot$, $\vee$, $\rightarrow$ dt $\leftrightarrow$. Par exemple, $(\phi_1 \rightarrow \phi_2) \stackrel{def}{=} \neg(\phi_1 \wedge \neg\phi_2)$.*

Définition 2 (CNF - Forme Normale Conjonctive)**.** *Un littéral est une variable propositionnelle dans $\mathbb{P}$ ou sa négation. Une clause est une disjonction de littéraux. Une formule en forme normale conjonctive (CNF) est une conjonction de clauses.*

On sait que toute formule de *CPL* peut être traduite en une formule logiquement équivalente en CNF. Cependant, en pratique, des traductions efficaces telles que celle de Tseitin [36] exige des variables supplémentaires. Ainsi, il ne conserve que la satisfiabilité et parfois le nombre de modèles. Ici, lorsque nous utilisons le terme CNF, ou simplement formule, nous entendons une formule en CNF.

De récents solveurs SAT (logiciels capables de décider de la satisfiabilité d'une CNF) sont capables de vérifier la satisfiabilité d'une formule "sous hypothèses" [8] et de sortir un noyau inconsistant (une "raison" pour son insatisfiabilité). Le noyau insatisfiable est défini comme suit :

Définition 3 (Noyau inconsistant sous hypothèses)**.** *Soit ϕ une formule satisfiable en Forme Normal Conjonctive (CNF) construite en utilisant les variables booléennes de $\mathbb{P}$. Soit A un ensemble cohérent de littéraux construits en utilisant des variables booléennes à partir de $\mathbb{P}$, de telle sorte que $(\phi \wedge \bigwedge_{a \in A} a)$ soit inconsistant. $C \subseteq A$ est un noyau inconsistant (UNSAT core) de ϕ sous hypothèses A si et seulement si $(\phi \wedge \bigwedge_{c \in C} c)$ est inconsistant.*

Maintenant, nous pouvons introduire les notions requises pour comprendre la logique modale. De plus amples détails suer la logique modale ayant déjà été décrits dans la littérature [5, 6], passons maintenant aux notions utilisant la sémantique de Kripke [17].

Définition 4 (Langage de la logique modale)**.** *Soit $\mathbb{P} = \{p_0, \ldots, p_{n-1}\}$ un ensemble fini et non-vide de n variables propositionnelles et $\mathbb{M} = \{\Box_1, \ldots, \Box_m\}$ un ensemble fini et non-vide de m opérateur modaux unaires. Le langage de la logique modale (noté $\mathcal{L}$) est l'ensemble des formules contenant $\mathbb{P}$ et fermé sous l'ensemble des connecteurs propositionnels $\{\neg, \wedge\}$ et sous l'ensemble des opérateurs modaux dans $\mathbb{M}$. Nous utilisons également l'abréviation standard $\Diamond_a\phi \stackrel{def}{=} \neg\Box_a\neg\phi$.*

La profondeur modale d'une formule ϕ dans $\mathcal{L}$, noté depth(ϕ), est le nombre le plus élevé de modalités imbriquées. Le nombre d'atomes dans une formule ϕ dans $\mathcal{L}$ est noté $Atom(\phi)$.

Définition 5 (Structure de Kripke)**.** *Soit $\mathbb{P}$ un ensemble fini et non-vide de n variables propositionnelles et $\mathbb{M}$ un ensemble fini et non-vide de m opérateurs modaux unaires.*

TABLE 1 – Axiomes et leurs propriétés structurelles

Axiome	Contrainte du premier ordre	Schéma
Réflexivité (T)	$R_a(w,w)$	$\Box_a\phi \rightarrow \phi$
Symétrie (B)	$R_a(w_1,w_2) \rightarrow R_a(w_2,w_1)$	$\phi \rightarrow \Box_a\Diamond_a\phi$
Sérialité (D)	$R_a(w_1,w_2)$	$\Box_a\phi \rightarrow \Diamond_a\phi$
Transitivité (4)	$(R_a(w_1,w_2)\wedge R_a(w_2,w_3)) \rightarrow R_a(w_1,w_3)$	$\Box_a\phi \rightarrow \Box_a\Box_a\phi$
Euclidianité (5)	$(R_a(w_1,w_2)\wedge R_a(w_1,w_3)) \rightarrow R_a(w_2,w_3)$	$\Box_a\phi \rightarrow \Diamond_a\Box_a\phi$

Une structure de Kripke est un triplet $\mathcal{K} = \langle W, \{R_a \mid \Box_a \in \mathbb{M}\}, V\rangle$, où : W est en ensemble non-vide de mondes possibles, chaque $R_a \subseteq W \times W$ est une relation d'accessibilité binaire sur W et $V : \mathbb{P} \rightarrow 2^W$ est une fonction d'évaluation qui associe, à chaque $p \in \mathbb{P}$, l'ensemble des mondes possibles de W où p est vraie. Une structure de Kripke pointée est une paire $\langle \mathcal{K}, w\rangle$, où $\mathcal{K}$ est une structure de Kripke et w est un monde possible dans W. Par la suite, chaque fois que nous utilisons le terme "structure de Kripke", nous entendons "structure de Kripke pointée".

La taille d'une structure de Kripke $\langle \mathcal{K}, w\rangle$, qui est le nombre d'éléments dans W, est notée dans la suite $|\mathcal{K}|$. Sans perte de généralité, nous considérons uniquement les formules logiques modales sous forme normale négative, notées NNF (les négations apparaissent seulement avant les variables propositionnelles) [26, p. 204].

Définition 6 (Relation de satisfiabilité)**.** *La relation de satisfiabilité $\vDash$ entre les formules de logique modale et les structures est récursivement définie comme suit :*

$\langle M, w\rangle \vDash \top$

$\langle M, w\rangle \vDash p$ *ssi* $w \in \mathcal{I}(p)$

$\langle M, w\rangle \vDash \neg\phi$ *ssi* $\langle M, w\rangle \nvDash \phi$

$\langle M, w\rangle \vDash \phi \wedge \psi$ *ssi* $\langle M, w\rangle \vDash \phi$ *et* $\langle M, w\rangle \vDash \psi$

$\langle M, w\rangle \vDash \phi \vee \psi$ *ssi* $\langle M, w\rangle \vDash \phi$ *ou* $\langle M, w\rangle \vDash \psi$

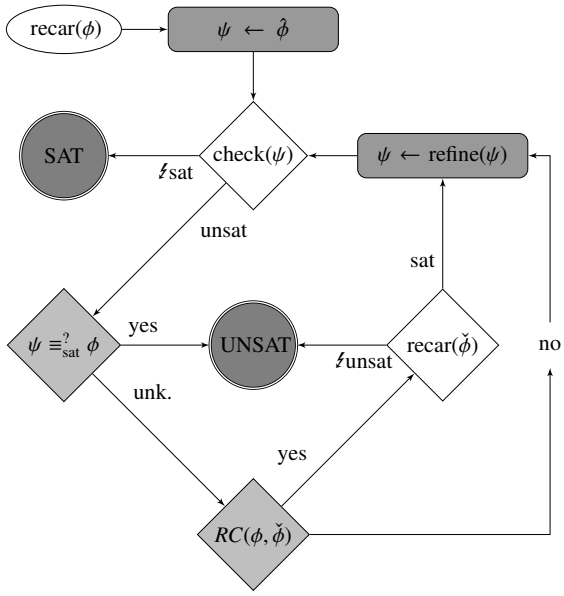
$\langle \mathcal{K}, w\rangle \models \Box_a\phi$ *ssi* $(w, w') \in R_a$ *implique* $\langle \mathcal{K}, w'\rangle \models \phi$

$\langle \mathcal{K}, w\rangle \models \Diamond_a\phi$ *ssi* $(w, w') \in R_a$ *et* $\langle \mathcal{K}, w'\rangle \models \phi$

Définition 7 (Validité)**.** *Une formule $\phi \in \mathcal{L}$ est valide (notée $\models \phi$) si et seulement si elle est satisfaite par toutes les structures de Kripke $\langle \mathcal{K}, w\rangle$. Une formule $\phi \in \mathcal{L}$ est satisfiable si et seulement si $\not\models \neg\phi$. Une structure de Kripke qui satisfait ϕ sera appelé un "modèle de ϕ".*

Par exemple, $\Box_a(\phi \rightarrow \psi) \rightarrow (\Box_a\phi \rightarrow \Box_a\psi)$ est valable pour tous $1 \leq a \leq m$ et toutes formules $\phi, \psi \in \mathcal{L}$. Cet ensemble particulier de formules est important dans la logique modale, il s'appelle "Schéma K ". Toutes les structures de Kripke satisfont K, et c'est pourquoi nous les appellerons ici "structures K". Une structure K peut satisfaire d'autres schémas comme ceux listés dans la Table 1. Toutes les combinaisons de ces schémas donnent lieu à 15 types de structures différentes [31, Table 25.2]. Ceci est dû au fait que certains schémas en impliquent d'autres (comme satisfaire (T) implique satisfaire (D), ou satisfaire (T) et (5) implique de satisfaire tous les schémas). Formellement nous avons :

FIGURE 1 – Le framework RECAR



Définition 8 ($K\star$-Validité)**.** *Soit $\star$ qui varie sur les 15 différents types de structures. Une formule $\phi \in \mathcal{L}$ est $K\star$-valide (notée $\models_{K\star} \phi$) si et seulement si elle est satisfaite par toutes les structures $K\star$ $\langle \mathcal{K}, w\rangle$. Une formule $\phi \in \mathcal{L}$ est $K\star$-satisfiable si et seulement si $\not\models_{K\star} \neg\phi$. Une structure $K\star$ qui $K\star$-satisfait une formule ϕ sera appelé un "modèle $K\star$ de ϕ'.*

Dans cet article, nous considérons seulement 3 logiques modales : K dont les modèles correspondent à toutes les structures K, KT dont les modèles correspondent à toutes les structures KT (c'est-à-dire, les structures K qui satisfont également le schéma (T)), et S4 dont les modèles correspondent à toutes les structures KT4 (c'est-à-dire, les structures K qui satisfont également les deux schémas (T) et (4)). En théorie, nous pouvons traiter toutes les logiques modales basées sur K. En pratique, nous avons testé expérimentalement seulement K, KT et S4 pour lesquels nous avons pu trouver des benchmarks.

3 Une approche RECAR pour K, KT et S4

Le framework RECAR, représenté sur la Fig.1, travaille avec deux niveaux d'abstraction. Il commence par considérer une sur-abstraction ψ de la formule d'entrée ϕ et vérifier sa satisfiabilité. Si ψ est satisfiable alors ϕ est satisfiable, la procédure s'arrête et renvoie SAT. Dans le cas contraire, il vérifie si ψ est trivialement equisatisfiable à ϕ. Si tel est le cas, alors ϕ est insatisfiable, la procédure s'arrête et renvoie

UNSAT. Sinon, il vérifie s'il est possible de construire une sous-abstraction $\check{\phi}$ de ϕ. Si une telle sous-abstraction ne peut être trouvée, la sur-abstraction est raffinée et le processus boucle. Sinon, $\check{\phi}$ est considérée comme à résoudre par un appel récursif à RECAR. Parce que $\check{\phi}$ est une sous-abstraction, si l'appel récursif prouve que $\check{\phi}$ est insatisfiable alors ϕ est également insatisfiable et la procédure s'arrête. Sinon, il ne peut pas conclure et donc il raffine ψ et répète tout le processus.

Pour être correctement instancié, certaines conditions doivent être vérifiées par les différents blocs utilisés dans RECAR [18] :

Hypothèses sur RECAR :

1. L'oracle 'check' est correct, complet et se termine.
2. $\hat{\phi}$ est satisfiable implique refine($\hat{\phi}$) est satisfiable.
3. Il existe $n \in \mathbb{N}$ tel que refine$^n(\hat{\phi}) \equiv^?_{\text{sat}} \phi$.
4. $\check{\phi}$ est insatisfiable implique ϕ est insatisfiable.
5. Let under(ϕ) $= \check{\phi}$. Il existe $n \in \mathbb{N}$ tel que RC(under$^n(\phi)$, under$^{n+1}(\phi)$) renvoie faux, où RC représente une fonction booléenne déterminant si un appel récursif doit avoir lieu.

Puisque '$\hat{\phi}$' est une CNF, il suffit d'utiliser un solveur SAT pour respecter Hypo. 1. Pour respecter Hypo. 2 et 3, [18] propose une fonction $\hat{\phi} = \text{over}(\phi, n)$ ce qui traduit la formule modale ϕ en une modélisation CNF demandant "est-ce que ϕ est satisfiable par un modèle de taille n ?". Lorsque la procédure répond positivement, ϕ est satisfiable. Sinon, il peut être raffiné avec $m > n$ jusqu'à ce que m atteint une borne supérieure théorique, qui est notée $UB(\phi)$. La borne supérieure théorique garantit également que le CNF généré est équisatisfiable à ϕ (RECAR Hypo. 3). La traduction ajoute de nouvelles variables p_i et $r^a_{i,j}$ à la formule : p_i dénote que la variable p est vraie dans le monde w_i alors que $r^a_{i,j}$ correspond à w_j étant accessible depuis w_i par la relation a.

Parce que ce sera important plus tard, nous rappelons ici que l'Hypo. 3 correspond aux lemmes suivants.

Such upper-bound is specific for each modal logic, as we can see below.

Lemme 1 ([30]). *$UB(\phi) = Atom(\phi)^{\text{depth}(\phi)}$ en logique modale K.*

Lemme 2 ([9]). *$UB(\phi) = 2^{|\phi|}$ en logique modales KT et S4.*

Dans ce qui suit, ces borne supérieures sont utilisées pour les logiques modales K, KT et S4. On sait que, s'il n'y a pas de modèle de taille $n \leq UB(\phi)$ qui satisfait ϕ alors il n'y a pas de modèle pour ϕ. Pour simplifier la lecture, bien que RECAR soit générique, dans la suite, nous utilisons RECAR pour se référer à une instanciation du framework pour la logique modale.

3.1 Comment encoder les axiomes

Il est connu que certains axiomes correspondent à des contraintes sur les structures de Kripke [27]. Par exemple, toute structure K réflexive satisfait (T). Même s'il existe aussi des structures K non-réflexives qui satisfont (T), il est toujours possible de trouver une structure K réflexive "équivalente". Deux structures K sont équivalentes si et seulement si elles sont bi-similaires [5]. Par conséquent, si l'on veut trouver un modèle KT fini, il est suffisant de ne rechercher que parmi les structures K réflexives. Un raisonnement analogue peut également être utilisé pour les autres propriétés. Par conséquent, en suivant la Table 1, nous appelons structure KT une structure K réflexive et nous appelons structure S4 (ou structure KT4) une structure K réflexive et transitive.

Par conséquent, pour traiter différentes logiques modales, nous ajoutons à la traduction en CNF les contraintes suivantes correspondant aux différents axiomes (avec m le nombre d'opérateurs modaux et n le nombre de mondes courants) :

Définition 9 (Traduction des axiomes)**.**

$$\text{over}((T), n) = \bigwedge_{a=0}^{m} \bigwedge_{i=0}^{n} (r^a_{i,i}) \quad \text{over}((D), n) = \bigwedge_{a=0}^{m} \bigwedge_{i=0}^{n} \bigvee_{j=0}^{n} (r^a_{i,j})$$

$$\text{over}((B), n) = \bigwedge_{a=0}^{m} \bigwedge_{i=0}^{n} \bigwedge_{j=0}^{n} (r^a_{i,j} \rightarrow r^a_{j,i})$$

$$\text{over}((4), n) = \bigwedge_{a=0}^{m} \bigwedge_{i=0}^{n} \bigwedge_{j=0}^{n} \bigwedge_{k=0}^{n} ((r^a_{i,j} \wedge r^a_{j,k}) \rightarrow r^a_{i,k})$$

$$\text{over}((5), n) = \bigwedge_{a=0}^{m} \bigwedge_{i=0}^{n} \bigwedge_{j=0}^{n} \bigwedge_{k=0}^{n} ((r^a_{i,j} \wedge r^a_{i,k}) \rightarrow r^a_{j,k})$$

La traduction de chaque axiome vient des relations avec la logique du premier ordre, présenté par [27]. Ainsi, lorsque l'axiome (T) est considéré (logique modal KT), la fonction de sur-abstraction est (over(ϕ, n) $\wedge$ over($(T), n$)). Quand les deux axiomes (T) et (4) (logique modale S4) sont considérés, la fonction de sur-abstraction est (over(ϕ, n) $\wedge$ over($(T), n$) $\wedge$ over($(4), n$)).

4 Abstractions agressives au niveau modal

Lorsque l'on considère des axiomes de logique modale autres que K, les formules ont généralement besoin de plus de mondes à satisfaire. Par conséquent, la traduction SAT devient parfois trop grande pour être manipulée (certains CNF ont des centaines de millions de clauses). Ainsi, nous proposons une nouvelle fonction de sur-abstraction sensible à l'espace et une nouvelle fonction de sous-abstraction consciente des axiomes, à utiliser dans le framework RECAR.

4.1 Sous-abstraction sensible aux axiomes

Jusqu'à présent, lorsque la formule était insatisfiable, la seule façon de prouver son insatisfiabilité était de couper une branche enracinée dans des nœuds ET afin de produire une sous-formule insatisfiable qui peut être traduite en CNF. En traitant avec l'axiome (T), nous avons $\Box_a\phi \rightarrow \phi$ et donc, comme démontré dans la propriété suivante, il est possible de construire une sous-abstraction de la formule qui supprime également certaines modalités.

Propriété 1. *Considérons une formule $\phi \in \mathcal{L}$ en NNF dans une logique modale satisfaisant (T). Si nous remplaçons une sous-formule $\Box_a\psi$ par ψ, alors la formule résultante ϕ' est une sous-abstraction de ϕ.*

Démonstration. Pour prouver que cette propriété est valide, il suffit de vérifier que si ϕ' est unsatisfiable alors ϕ l'est aussi. Ou, avec la contraposée, il suffit de vérifier que si ϕ est satisfiable alors ϕ' l'est aussi. Sans perte de généralité, dans ce qui suit nous supposons que tous les nœuds OU et ET sont binaires et nous rappelons que les opérateurs booléens sont commutatifs.

Considérons une formule logique modale ϕ en NNF, et ϕ' une copie de ϕ qui diffère d'une seule sous-formule enracinée sur un nœud de box $\Box_a\psi \in \phi$ où $\Box_a\psi$ a été remplacée par ψdans ϕ'.

Nous montrons que, s'il existe un modèle de Kripke $\mathcal{K} = \langle W, \{R_a \mid \Box_a \in \mathbb{M}\}, V\rangle$ et un monde possible $w \in W$, t.q. $\langle\mathcal{K}, w\rangle \models \phi$ alors $\langle\mathcal{K}, w\rangle \models \phi'$ par induction sur la structure de ϕ. Base d'induction : $\phi = \Box_a\psi$ et $\phi' = \psi$. Car ϕ et ϕ' sont en NNF, si $\exists\langle\mathcal{K}, w\rangle \models \Box_a\psi$, alors nous savons que nous avons $(w, w) \in R_a$ en raison de l'axiome de réflexivité (T), nous avons donc $\exists\langle\mathcal{K}, w\rangle \models \psi$. Montrons maintenant les différents cas sur l'étape d'induction :

(1) $\phi = (\chi_1 \wedge \chi_2)$ et χ_1 contient $(\Box_a\psi)$ et $\phi' = (\chi'_1 \wedge \chi_2)$ où χ'_1 est χ_1 où $(\Box_a\psi)$ a été remplacée par ψ. Le cas où χ_2 contient $(\Box_a\psi)$ est analogue à cause de la commutativité. Nous avons $\langle\mathcal{K}, w\rangle \models (\chi'_1 \wedge \chi_2)$ si et seulement si $\langle\mathcal{K}, w\rangle \models \chi'_1$ et $\langle\mathcal{K}, w\rangle \models \chi_2$. Par l'hypothèse d'induction, $\langle\mathcal{K}, w\rangle \models \chi_1$ et $\langle\mathcal{K}, w\rangle \models \chi_2$, si et seulement si $\langle\mathcal{K}, w\rangle \models (\chi_1 \wedge \chi_2)$. Ainsi $\langle\mathcal{K}, w\rangle \models \phi$.

(2) $\phi = (\chi_1 \vee \chi_2)$ et χ_1 contient $(\Box\psi)$ et $\phi' = (\chi'_1 \vee \chi_2)$ où χ'_1 est χ_1 où $(\Box_a\psi)$ a été remplacée par ψ. Analogue à (1).

(3) $\phi = \Box_a\chi$ et χ contient $(\Box_a\psi)$ et $\phi' = \Box_a\chi'$ où χ' est χ où $(\Box_a\psi)$ a été remplacée par ψ. Nous avons $\langle\mathcal{K}, w\rangle \models \Box_a\chi'$ si et seulement si $\forall w'$ t.q. si $(w, w') \in R_a$ alors nous avons $\langle\mathcal{K}, w'\rangle \models \chi'$. Par l'hypothèse d'induction, $\langle\mathcal{K}, w'\rangle \models \chi$, alors $\forall w'$ t.q. si $(w, w') \in R_a$ alors nous avons $\langle\mathcal{K}, w'\rangle \models \Box_a\chi$. Ainsi $\langle\mathcal{K}, w\rangle \models \phi$.

(4) $\phi = \Diamond_a\chi$ et χ contient $(\Box_a\psi)$ et $\phi' = \Diamond_a\chi'$ où χ' est χ où $(\Box_a\psi)$ a été remplacée par ψ. Analogue à (3).

Par conséquent pour toute formule $\phi \in \mathcal{L}$ en NNF dans une logique modale satisfaisant (T). Si nous remplaçons une sous-formule $\Box_a\psi$ par ψ, alors la formule résultante ϕ' est une sous-abstraction de ϕ. □

Notez que cette propriété ne s'applique pas à la logique modale K : $\Box_a\bot$ est valide en logique modale K mais incohérent en logique modale KT. Afin de sélectionner les cases qui seront remplacées, nous proposons d'améliorer la sous-abstraction présentée dans [18], qui combine sélecteurs et noyaux inconsistant pour rechercher des sous-formules insatisfiables.

Là, les auteurs ont proposé d'ajouter un sélecteur à chaque branche enracinée dans un nœud ET afin de pouvoir activer/désactiver certaines parties de la formule. Lorsque le solveur est appelé pour vérifier la satisfiabilité de la formule, il est appelé avec l'ensemble des sélecteurs comme hypothèses. Si le solveur renvoie UNSAT, le noyau inconsistant renvoyé est utilisé pour supprimer les parties de la formule qui ne font pas parti de la raison de son incohérence. Ce que nous proposons ici consiste également à remplacer chaque $\Box_a\psi$ par $((\neg s_k \vee \Box_a\psi) \wedge \psi)$. D'une certaine manière "quand nous activons le sélecteur s_k, alors nous traduisons la modalité entière, sinon nous traduisons simplement ψ dans le monde courant". Maintenant, nous pouvons à nouveau utiliser le noyau retourné par le solveur pour extraire une sous-formule insatisfiable. La sous-abstraction supprimera certaines cases qui ne sont pas satisfiables en raison de l'incohérence de la formule. À noter, quand nous traduisons une modalité □ en SAT, il n'est pas utile de traduire deux fois la formule ψ dans le monde courant.

4.2 Sur-abstraction sensible à la mémoire

Comme indiqué dans l'introduction, le goulot d'étranglement des approches `CEGAR` utilisant un oracle SAT est la taille des formules CNF générées. Pour le cas qui nous intéresse, ce goulot d'étranglement est atteint lorsque la fonction de sur-abstraction est appelée avec un trop grand nombre de mondes. Cependant, il est possible d'estimer la taille de la formule CNF avant de la calculer et ainsi prédire que la traduction épuisera la mémoire autorisée pour résoudre l'instance.

Dans ce qui suit, nous proposons une fonction de sur-abstraction spatiale qui est utilisée à la place de la fonction originale lorsque l'espace occupé par le CNF atteint un seuil donné. Cette nouvelle sur-abstraction est effectuée en désactivant certaines disjonctions de la formule d'origine. Pour que cette fonction existe et respecte les hypothèses de `RECAR`, nous devons vérifier que les branches enracinées dans un nœud OU qui ont été coupées produisent une formule plus faible, c'est-à-dire que chaque modèle de la formule résultante est également un modèle de la formule initiale.

Propriété 2. *Considérons une formule logique modale $\phi \in \mathcal{L}$ en NNF. Si nous coupons une branche enracinée dans un nœud OU, alors la formule résultante ϕ' est une sur-abstraction de ϕ.*

Démonstration. Pour prouver que cette propriété est valide, il suffit de vérifier que chaque modèle de ϕ' est aussi un modèle de ϕ. Sans perte de généralité, dans ce qui suit nous supposons que tous les nœuds OU et ET sont binaires. En effet, $(\phi_1 \oplus \phi_2 \oplus \cdots \oplus \phi_n)$ peut être réécrit comme $(\phi_1 \oplus (\phi_2 \oplus (\cdots \oplus (\phi_{n-1} \oplus \phi_n))))$, où $\oplus \in \{\wedge, \vee\}$. Rappelons aussi que les opérateurs booléens sont commutatifs, donc nous n'avons pas besoin de prouver le cas où nous ajoutons une nouvelle sous-formule à gauche. Soit ϕ une formule en NNF $\mathcal{L}$ contenant $(\psi_1 \vee \psi_2)$ et ϕ' est égale à ϕ mais avec $(\psi_1 \vee \psi_2)$ remplacée par ψ_1. Nous montrons que, s'il existe un modèle de Kripke $\mathcal{K} = \langle W, \{R_a \mid \Box_a \in \mathbb{M}\}, V\rangle$ et un monde possible $w \in W$, t.q. $\langle \mathcal{K}, w\rangle \models \phi'$ alors $\langle \mathcal{K}, w\rangle \models \phi$ par induction sur la structure de ϕ. Base d'induction : $\phi' = \psi_1$ et $\phi = (\psi_1 \vee \psi_2)$. Puisque ϕ et a fortiori ϕ' sont en NNF, la propriété est clairement valide. Laissez-nous maintenant prouver les différentes étapes :

(1) $\phi = (\chi_1 \wedge \chi_2)$ et χ_1 contient $(\psi_1 \vee \psi_2)$ et $\phi' = (\chi' \wedge \chi_2)$ où χ' est χ où $(\psi_1 \vee \psi_2)$ a été remplacée ψ_1. Le cas où χ_2 contient $(\psi_1 \vee \psi_2)$ est analogue à cause de la commutativité. Nous avons $\langle \mathcal{K}, w\rangle \models (\chi'_1 \wedge \chi_2)$ si et seulement si $\langle \mathcal{K}, w\rangle \models \chi'_1$ et $\langle \mathcal{K}, w\rangle \models \chi_2$. Par l'hypothèse d'induction, $\langle \mathcal{K}, w\rangle \models \chi_1$ et $\langle \mathcal{K}, w\rangle \models \chi_2$, si et seulement si $\langle \mathcal{K}, w\rangle \models (\chi_1 \wedge \chi_2)$. Ainsi $\langle \mathcal{K}, w\rangle \models \phi$.

(2) $\phi = (\chi_1 \vee \chi_2)$ et χ_1 contient $(\psi_1 \vee \psi_2)$ et $\phi' = (\chi' \vee \chi_2)$ où χ' est χ où $(\psi_1 \vee \psi_2)$ a été remplacée ψ_1. Analogue à (1).

(3) $\phi = \Box_a \chi$ et χ contient $(\psi_1 \vee \psi_2)$ et $\phi' = \Box_a \chi'$ où χ' est χ où $(\psi_1 \vee \psi_2)$ a été remplacée ψ_1. Nous avons $\langle \mathcal{K}, w\rangle \models \Box_a \chi'$ iff $\forall w'$ s.t. if $(w, w') \in R_a$ alors nous avons $\langle \mathcal{K}, w'\rangle \models \chi'$. Par l'hypothèse d'induction, $\langle \mathcal{K}, w'\rangle \models \chi$, alors $\forall w'$ t.q. si $(w, w') \in R_a$ alors nous avons $\langle \mathcal{K}, w'\rangle \models \Box_a \chi$. Ainsi $\langle \mathcal{K}, w\rangle \models \phi$.

(4) $\phi = \Diamond_a \chi$ et χ contient $(\psi_1 \vee \psi_2)$ et $\phi' = \Diamond_a \chi'$ où χ' est χ où $(\psi_1 \vee \psi_2)$ a été remplacée ψ_1. Analogue à (3).

Ainsi, $\forall \phi$ en NNF, si $\langle \mathcal{K}, w\rangle \models \phi'$ alors $\langle \mathcal{K}, w\rangle \models \phi$. □

Remarquons que la Propriété 2 peut être étendue au cas où un ensemble de branches enracinées dans des nœuds OU sont coupés. Par conséquent, il est possible d'envisager un nouveau type de sur-abstraction qui coupe les bords enracinés dans les nœuds OU. Plus précisément, nous créons une telle sous-formule de sur-abstraction en coupant via une heuristique un ensemble de branches enracinées dans des nœuds OU qui ont le plus grand impact sur la limite supérieure et que nous traduisons en une formule CNF. Nous appelons cette fonction cut-or(ϕ, b), où le paramètre b est le nombre de nœuds OU coupés. Évidemment, la fonction de raffinement est liée à cette sur-abstraction (il n'est pas possible de couper une branche et d'augmenter le nombre de mondes ensuite sans violer les hypothèse de RECAR). Ainsi, nous considérons une nouvelle fonction de raffinement refine$_{or}(\hat{\phi}, b)$ qui est défini de manière récursive comme suit :

$$\text{refine}_{or}(\hat{\phi}, b) = \begin{cases} \hat{\phi} & \text{si } b = 0 \\ \phi' & \forall b > 0 \text{ et } \text{refine}_{or}(\hat{\phi}, b-1) \neq \phi \\ \phi & \text{sinon} \end{cases}$$

où ϕ' est défini de telle sorte que $\text{refine}_{or}(\hat{\phi}, b-1) \subsetneq \phi' \subseteq \phi$. Intuitivement, refine$_{or}(\hat{\phi}, b)$ rétablit b nœuds OU à $\hat{\phi}$ si possible.

Fondamentalement, la fonction de raffinement consiste à restaurer au moins une branche coupée à chaque étape d'induction. Maintenant, montrons que le couple proposé sur-abstraction cut-or(ϕ, b) et fonction de raffinement refine$^n_o(\psi)$ satisfait les conditions requises pour RECAR rappelées dans les préliminaires.

Théorème 1. $\psi = \text{cut-or}(\phi, n)$ *est satisfiable implique* $\text{refine}_{or}(\psi, 1)$ *est satisfiable (Hypo. 2) et* $\text{refine}_{or}(\psi, n) \equiv_{\text{sat}} \phi$ *(Hypo. 3).*

Démonstration. Par la Propriété 2 nous savons qu'ajouter une branche à un nœud enraciné en un OU préserve la satisfiabilité. Puisque refine$_{or}(\psi, b)$ peut seulement rajouter des branches, alors nous avons directement que $\psi = \text{cut-or}(\phi, n)$ est satisfiable implique refine$_{or}(\psi, 1)$ est satisfiable. Pour l'hypothèse 3, le résultat vient directement de la définition de refine$_{or}(\psi, n)$ et le fait que nous ne pouvons ajouter qu'un nombre fini de branches. □

4.3 Simplifications des chaînes de modalités

Une fois que toutes les simplifications ci-dessus sont effectuées, la formule "abstraite" qui en résulte peut contenir des chaînes de modalités. C'est une information critique qui peut être exploitée si nous voulons améliorer encore les performances du solveur.

Simplifications pour la logique modale S4 Les simplifications des chaîne des modalités dans la logique modale S4 sont connues et elles préservent l'équivalence logique [37, Sec. 5.5].

$$\Box_a \Box_a \phi \leftrightarrow \Box_a \phi \quad (1)$$

$$\Diamond_a \Diamond_a \phi \leftrightarrow \Diamond_a \phi \quad (2)$$

$$\Box_a \Diamond_a \Box_a \Diamond_a \phi \leftrightarrow \Box_a \Diamond_a \phi \quad (3)$$

Ainsi, cela signifie que toute chaîne d'opérateurs modaux (impliquant la même modalité a) peut être réduite à une chaîne d'au maximum 3 opérateurs modaux dans la logique modale S4, ce qui est tolérable pour la traduction en CNF faite par la fonction de sur-abstraction.

Puisque la logique modale K et la logique modale KT ont une infinité de modalités non équivalentes (*infinitely many non-equivalent modalities*) [37], il n'y a pas de tel résultat au meilleur de nos connaissances pour K et KT. Pour traiter des chaînes de modalités dans ces logiques, nous proposons les simplifications suivantes qui préservent la satisfiabilité.

Simplifications pour la logique modale K Considérons un cas simple où le préfixe modal ne contient que des diamants. Dans ce cas, il est suffisant de tester uniquement la fin de la chaîne sans tenir compte des modalités. La sous-formule résultante est équisatisfiable à la formule originale.

Théorème 2. $\Diamond_a \Diamond_a \dots \Diamond_a \psi \equiv_{\text{sat}} \psi$.

Démonstration. ($\Rightarrow$) Si $\langle \mathcal{K}, w \rangle \models \Diamond_a \dots \Diamond_a \psi$ alors il y a $w' t.q. \langle \mathcal{K}, w' \rangle \models \psi$. ($\Leftarrow$) Si $\langle \mathcal{K}, w \rangle \models \psi$ alors nous pouvons ajouter un monde w' à $\mathcal{K}$ t.q. $w \in R_a(w')$. Dans ce cas, $\langle \mathcal{K}, w' \rangle \models \Diamond_a \psi$. En continuant ainsi, nous pouvons montrer que $\Diamond_a \dots \Diamond_a \psi$ est satisfiable. □

Théorème 3. $\Diamond_a \dots \Diamond_a \Box_a \circ \dots \circ \psi$ *est satisfiable, pour* $\circ \in \{\Box_a, \Diamond_a\}$.

Démonstration. $\Box_a \psi$ est satisfiable pour tous $\psi \in \mathcal{L}$. Par conséquent, par Theo 2, $\Diamond_a \dots \Diamond_a \Box_a \chi$ est satisfiable pour tous $\chi \in \mathcal{L}$. □

Simplifications pour la logique modale KT La logique modale KT est différente. La réflexivité (Axiome (T)) implique que $\Box_a \bot$ est unsatisfiable, ce qui signifie que nous ne pouvons pas appliquer la même technique qu'en logique modale K. De plus, en raison de l'absence de transitivité (Axiome (4)), nous ne pouvons pas simplifier $(\Box_a \Diamond_a \Box_a \Diamond_a \phi)$ en $(\Box_a \Diamond_a \phi)$, comme c'est le cas en S4. Nous proposons donc de nouvelles simplifications qui préservent la satisfiabilité (mais pas l'équivalence) dans KT. Cependant, il est toujours possible de récupérer un modèle pour la formule d'origine en trouvant un modèle pour la formule simplifiée.

Théorème 4. $\Box_a \circ \psi \equiv_{\text{sat}} \circ \psi$, *for* $\circ \in \{\Box_a, \Diamond_a\}$.

Démonstration. Cas 1 : $\circ = \Diamond_a$. ($\Rightarrow$) If $\Box_a \Diamond_a \psi$ est satisfiable alors, par l'axiome (T), $\Diamond_a \psi$ est satisfiable. ($\Leftarrow$) Supposons $\langle \mathcal{K}, w \rangle \models \Diamond_a \psi$. Nous pouvons créer un nouveau modèle $\langle \mathcal{K}', w \rangle$ qui est le déroulement de $\langle \mathcal{K}, w \rangle$. Ceci est un arbre infini avec w comme racine, qui est bisimilaire à $\langle \mathcal{K}, w \rangle$. Par hypothèse, il y a au moins un monde $w' \in R_a(w)$ t.q. $\langle \mathcal{K}', w' \rangle \models \psi$. Notez également que la propriété de réflexivité de la structure d'origine $\mathcal{K}$ implique $\langle \mathcal{K}', w' \rangle \models \Diamond_a \psi$. Il peut aussi y avoir des mondes $w'' \in R_a(w)$ t.q. $\langle \mathcal{K}, w'' \rangle \not\models \psi$. Nous pouvons supprimer toutes ces branches de la racine. Dans le modèle obtenu, tous les mondes accessibles depuis w satisfont ψ. Pour faire de ce modèle un modèle KT, nous ajoutons une flèche réflexive uniquement sur la racine w. Ce modèle final satisfait $\Box_a \Diamond_a \psi$. Cas 2 est similaire. La seule différence est qu'il n'y a pas de branches w'' à supprimer. □

Théorème 5. $\Diamond_a \psi \equiv_{\text{sat}} \psi$

Démonstration. ($\Rightarrow$) S'il y a $\langle \mathcal{K}, w \rangle \models \Diamond_a \psi$, alors il y a $w' \in R_a(w)$ t.q. $\langle \mathcal{K}, w' \rangle \models \psi$. ($\Leftarrow$) If $\langle \mathcal{K}, w \rangle \models \psi$, par la contraposée de l'axiome (T), $\langle \mathcal{K}, w \rangle \models \Diamond_a \psi$. □

5 Résultats expérimentaux

Nous avons implémenté les abstractions et les simplifications proposées en plus de MoSaiC 1.0 [18, 19], dans un solveur appelé MoSaiC 2.0. Nous avons choisi de comparer les solveurs sur les benchmarks LWB classiques pour les logiques modales K, KT et S4 [3]. Ces benchmarks sont générés en utilisant le script de [22] en utilisant 56 formules avec 18 paramètres, pour un total de 1008 formules, 504 satisfiables, 504 insatisfiables.

La Table 2 synthétise les résultats. Elle montre le nombre de benchmarks résolus, ou une cellule grise si le solveur ne gère pas la logique considérée. En gras, les meilleurs résultats d'une ligne donnée et, entre parenthèses, le nombre de cas-tests qui ne peuvent être résolus par manque de mémoire.

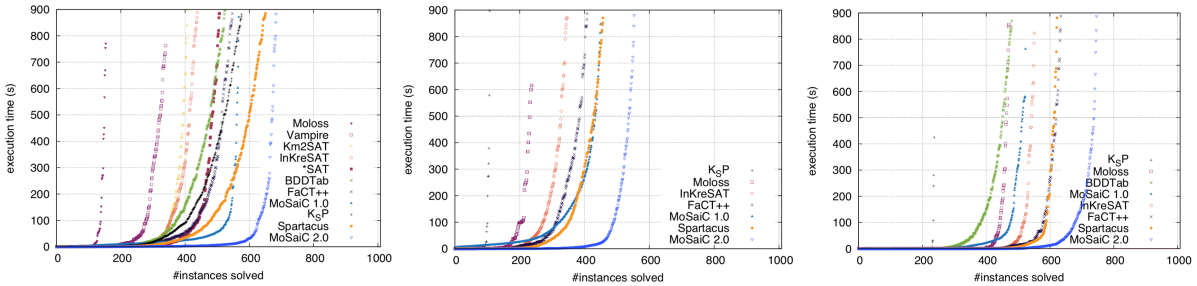
La ligne du VBS représente le *Virtual Best Solver* (une limite supérieure pratique sur la performance réalisable en choisissant le meilleur solveur pour chaque benchmark). Les expériences ont été exécutées sur un cluster de Xeon, 4 cœurs, 3.3 GHz avec CentOS 6.4 avec une limite de mémoire de 32Go et une limite de temps de 900 secondes par solveur par instance, quel que soit la logique considérée. Toutes les réponses aux solveurs ont été vérifiées car la satisfiabilité des instances est connu par construction. Aucune différence n'a été trouvée.

Nous avons comparé MoSaiC 2.0, avec les solveurs de l'état de l'art pour les logiques modales K, KT et S4, à savoir :

Moloss 0.9 [1],
$K_S P$ 0.1.2 [22],
BDDTab 1.0 [11],
FaCT++ 1.6.4 [35],
InKreSAT 1.0 [15],
*SAT 1.3 [10],
Km2SAT 1.0 [32] combiné avec le même solveur SAT Glucose (4.1) utilisé dans MoSaiC (1.0 et 2.0) [2],
Spartacus 1.0 [12],
MoSaiC 1.0 [19],
Vampire 4.0 [16] avec une combinaison de la traduction fonctionnelle optimisée [13].

Solver #Instances	**LWB$_K$ SAT** 504	**LWB$_K$ UNSAT** 504	**Total$_K$** 1008	**LWB$_{KT}$ SAT** 504	**LWB$_{KT}$ UNSAT** 504	**Total$_{KT}$** 1008	**LWB$_{S4}$ SAT** 504	**LWB$_{S4}$ UNSAT** 504	**Total$_{S4}$** 1008
Moloss	71 (0)	83 (0)	154 (0)	68 (0)	170 (0)	238 (0)	269 (0)	203 (0)	472 (0)
InKreSAT	192 (24)	247 (0)	439 (24)	155 (9)	193 (0)	348 (9)	248 (0)	304 (0)	552 (0)
BDDTab	248 (5)	277 (4)	525 (9)	–	–	–	211 (0)	270 (0)	481 (0)
FaCT++	264 (10)	284 (19)	548 (29)	184 (30)	226 (59)	410 (89)	298 (42)	338 (25)	636 (67)
MoSaiC 1.0	263 (241)	306 (198)	569 (439)	230 (251)	222 (253)	452 (504)	277 (229)	247 (255)	524 (484)
$K_S P$	249 (4)	**328** (3)	577 (7)	130 (2)	93 (0)	223 (2)	223 (0)	205 (0)	428 (0)
Spartacus	331 (33)	320 (10)	651 (43)	207 (74)	251 (59)	458 (133)	273 (17)	350 (13)	623 (30)
MoSaiC 2.0	**362** (142)	321 (73)	**684** (215)	**304** (167)	**249** (219)	**553** (386)	**360** (8)	**390** (31)	**750** (39)
VBS	362	342	704	304	249	553	364	390	750

TABLE 2 – Number of LWB instances solved in K, KT and S4



(a) en LWB K (b) en LWB KT (c) en LWB S4

FIGURE 2 – Distribution des temps d'exécution

Pour le basculement entre les deux fonctions de sur-abstraction dans MoSaiC 2.0, nous avons déterminé expérimentalement que le seuil ($|\phi| \times borneCourante > 5000$) a bien fonctionné pour ces instances sur notre ordinateur avec une limite de mémoire de 32 Go.

Les résultats obtenus Il est important de remarquer que $K_S P$ (aimablement fourni par ses auteurs) est encore en développement pour les logiques modales KT et S4. Ses résultats doivent être considérés comme préliminaires. Nous pouvons voir que le nombre de *Memory-Out* entre MoSaiC 1.0 et MoSaiC 2.0 réduit drastiquement dans les trois logiques grâce à la nouvelle sur-abstraction, qui réduit la taille de la formule qui doit être traduite.

En logique modale S4, on peut voir que l'écart entre MoSaiC 1.0 et MoSaiC 2.0 est plus important. L'approche basée sur SAT pour S4 est extrêmement efficace en raison des simplifications des chaînes de modalité, qui réduisent considérablement le temps de traduction. De plus, le temps de traduction étant réduit en raison de la nouvelle sur-abstraction et de la nouvelle sous-abstraction, MoSaiC 2.0 est beaucoup plus rapide que MoSaiC 1.0 (2s vs 28s en temps médian). Une autre manière d'observer l'impact de ces nouvelles abstractions est de regarder la distribution des temps d'exécution entre MoSaiC 1.0 et 2.0, visible sur la Figure 2(a) pour la logique modale K, sur la Figure 2(b) pour la logique modale KT et sur la Figure 2(c) pour la logique modale S4. Le coût de la traduction était très visible pour MoSaiC 1.0 si on compare sa distribution de temps d'exécution par rapport aux autres solveurs de l'état de l'art (il résout très peu d'instance en moins de 20 secondes par exemple), alors que MoSaiC 2.0 a pallié à ce coût, d'où le temps de résolution médian beaucoup plus bas.

Afin de comprendre la différence d'efficacité de la résolution de ces logiques, nous avons collecté des informations sur la taille des modèles Kripke calculés.

TABLE 3 – Tailles des structures de Kripke pour des benchmarks satisfiable pour chaque logique modale

	min	**Q$_1$**	**med**	**mean**	**Q$_3$**	**max**
K	1	2	22	174	279	903
KT	1	52	207	364	613	1505
S4	1	33	113	256	399	1217

Comme représenté dans la Table 3, l'ajout d'axiomes tend à augmenter la taille des modèles trouvés. Ceci peut s'expliquer en partie par le fait que ces modèles doivent satisfaire plus de contraintes. Cela arrive moins souvent en S4 car nous pouvons réduire considérablement le nombre de modalités.

6 Conclusion

Dans cet article, nous présentons un nouveau solveur de logique modale MoSaiC 2.0 qui étend MoSaiC 1.0 en considérant de nouvelles fonctions d'abstraction et en ciblant de nouvelles logiques modales (à savoir KT et S4). Les nouvelles fonctions d'abstraction visent à extraire une sous-formule insatisfiable, lorsque le problème est insatisfiable, ou de trouver un modèle pour une sous-formule satisfiable qui peut être étendue à toute la formule.

Les nouvelles fonctions d'abstraction ont été conçues pour traiter les cas où MoSaiC 1.0 sature la mémoire. Nous avons montré que ces fonctions pouvaient être combinées avec celles d'origine afin d'améliorer l'efficacité de l'approche. À cet égard, nous avons étendu le principe de simplification de la formule au niveau du domaine afin d'obtenir de meilleurs résultats au niveau SAT. Des simplifications logiques supplémentaires ont été proposées pour traiter les chaînes de modalités générées par les nouvelles fonctions d'abstraction.

MoSaiC 2.0 surpasse les autres solveurs sur les benchmarks considérés. Alors que le système de sélection actuel pour les abstractions à appliquer est simple et empirique, l'étape suivante serait d'adapter les abstractions à utiliser en fonction de la mémoire disponible, à chaque étape de la procédure.

Remerciements

Les auteurs remercient Ullrich Hustadt pour ses scripts en Perl pour générer les benchmarks LWB, Cláudia Nalon pour son aide sur comment faire pour que $K_S P$ résolvent des instances en KT et S4 et Mark Kaminski pour son aide sur FaCT++ et comment lui faire résoudre des instances en KT et S4.

Une partie de ce travail a été supporté par le Ministère de l'Enseignement Supérieur et la Recherche, par le projet ANR Investissement d'Avenir UCA$^{\text{JEDI}}$ (ANR-15-IDEX-01) et par le Conseil Régional des Haut-de-France au travers du "Contrat de Plan État Région (CPER) DATA".

Références

[1] Areces, Carlos, Pascal Fontaine et Stephan Merz: *Modal Satisfiability via SMT Solving*. Dans *Software, Services, and Systems*, pages 30–45. Springer, 2015, ISBN 978-3-319-15545-6. https://doi.org/10.1007/978-3-319-15545-6_5.

[2] 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.

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

[4] 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.

[5] Blackburn, Patrick, Johan van Benthem et Frank Wolter: *Handbook of Modal Logic*, tome 3. Elsevier, 2006, ISBN 978-0444516909.

[6] Chellas, Brian F.: *Modal Logic: An Introduction*. Cambridge University Press, 1980, ISBN 978-0521295154.

[7] Clarke, Edmund M., Orna Grumberg, Somesh Jha, Yuan Lu et Helmut Veith: *CounterExample-Guided Abstraction Refinement For Symbolic Model Checking*. J. of the ACM, 50(5):752–794, 2003. http://doi.acm.org/10.1145/876638.876643.

[8] 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.

[9] Gabbay, Dov M.: *A General Filtration Method For Modal Logics*. Journal of Philosophical Logic, 1(1) :29–34, 1972. https://doi.org/10.1007/BF00649988.

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

[11] 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.

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

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

[14] Janota, Mikolás, William Klieber, Joao Marques-Silva et Edmund M. Clarke: *Solving QBF with CounterExample Guided Refinement*. Art. Int., 234:1–25, 2016.

[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] 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.

[17] Kripke, Saul: *A Completeness Theorem in Modal Logic*. J. Symb. Log., 24(1):1–14, 1959. http://dx.doi.org/10.2307/2964568.

[18] 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. https://doi.org/10.24963/ijcai.2017/94.

[19] Lagniez, Jean Marie, Daniel Le Berre, Tiago de Lima et Valentin Montmirail: *A SAT-Based Approach For PSPACE Modal Logics*. Dans *Proc. of KR'18*, pages 651–652. AAAI Press, 2018. https://aaai.org/ocs/index.php/KR/KR18/paper/view/17997.

[20] Lagniez, Jean-Marie et Pierre Marquis: *An Improved Decision-DNNF Compiler*. Dans *Proc. of IJCAI'17*, pages 667–673, 2017. https://doi.org/10.24963/ijcai.2017/93.

[21] Matsuhisa, Takashi: *Core Equivalence in Economy for Modal Logic*. Dans *Proc. of ICCS'03, Part II*, pages 74–83, 2003. https://doi.org/10.1007/3-540-44862-4_9.

[22] 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.

[23] Oztok, Umut et Adnan Darwiche: *On Compiling CNF into Decision-DNNF*. Dans *Proc. of CP'14*, pages 42–57, 2014. https://doi.org/10.1007/978-3-319-10428-7_7.

[24] Prestwich, Steven David: *CNF Encodings*. Dans Biere, Armin *et al.* [4], pages 75–97, ISBN 978-1-58603-929-5. http://dx.doi.org/10.3233/978-1-58603-929-5-75.

[25] Rintanen, Jussi: *Planning and SAT*. Dans Biere, Armin *et al.* [4], pages 483–504, ISBN 978-1-58603-929-5. https://doi.org/10.3233/978-1-58603-929-5-483.

[26] Robinson, John Alan et Andrei Voronkov (rédacteurs): *Handbook of Automated Reasoning*, tome 1. Elsevier and MIT Press, 2001, ISBN 0-444-50813-9.

[27] Sahlqvist, Henrik: *Completeness and correspondence in the first and second order semantics for modal logic*. Dans *Proc. of the 3rd Scandinavian Logic Sumposium*, 1973.

[28] Samulowitz, Horst et Fahiem Bacchus: *Using SAT in QBF*. Dans *Proc. of CP'05*, pages 578–592, 2005. https://doi.org/10.1007/11564751_43.

[29] Sang, Tian, Fahiem Bacchus, Paul Beame, Henry A. Kautz et Toniann Pitassi: *Combining Component Caching and Clause Learning for Effective Model Counting*. Dans *Proc. of SAT'04*, 2004. http://www.satisfiability.org/SAT04/programme/21.pdf.

[30] Sebastiani, Roberto et David McAllester: *New Upper Bounds for Satisfiability in Modal Logics the Case-study of Modal K*. Technical Report 9710-15, IRST, Trento, Italy, October 1997.

[31] Sebastiani, Roberto et Armando Tacchella: *SAT Techniques for Modal and Description Logics*. Dans Biere, Armin *et al.* [4], pages 781–824, ISBN 978-1-58603-929-5. http://dx.doi.org/10.3233/978-1-58603-929-5-781.

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

[33] Thurley, Marc: *sharpSAT - Counting Models with Advanced Component Caching and Implicit BCP*. Dans *Proc. of SAT'06*, pages 424–429, 2006. https://doi.org/10.1007/11814948_38.

[34] Toriz, Juan Pablo Munoz, Iván Martínez Ruiz et José Ramón Enrique Arrazola-Ramírez: *On Automatic Theorem Proving with ML*. Dans *Proc. of MICAI'14*, pages 231–236, 2014. https://doi.org/10.1109/MICAI.2014.42.

[35] 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.

[36] Tseitin, G. S: *On the Complexity of Derivation in Propositional Calculus*, pages 466–483. Springer, 1983.

[37] Van Benthem, Johan: *Modal Logic For Open Minds*, tome 1. Center for the Study of Language and Inf, 2010, ISBN 978-1575865980.