Sur le calcul des max-réfutations minimales

Meriam Ayari[1,2], Sami Cherif[1], Felip Manyà[2]

[1] Laboratoire MIS UR 4290, Université de Picardie Jules Verne, Amiens, France
[2] Artificial Intelligence Research Institute (IIIA), CSIC, Bellaterra, Spain

{meriam.ayari,sami.cherif}@u-picardie.fr felip@iiia.csic.es

Résumé

MaxSAT est une version d'optimisation de SAT qui consiste à déterminer le nombre maximum de clauses qu'il est possible de satisfaire par une affectation des variables. Les certificats d'optimalité pour les solutions MaxSAT sont essentiels pour la vérification et ont été principalement étudiés à travers le prisme des systèmes de preuve, la max-résolution étant la règle d'inférence la plus étudiée dans la littérature. Dans cet article, nous abordons le problème de la recherche de réfutations par max-résolution de taille minimale. Nous proposons une approche basée sur SAT, qui exploite les encodages propositionnels pour modéliser la recherche de réfutations par max-résolution de taille fixée, tout en respectant les contraintes de compensation et de consommation de clauses imposées par la règle.

Mots-clés

Satisfiabilité Maximum, Max-résolution, Réfutation

Abstract

MaxSAT is an optimization variant of SAT that aims to find an assignment maximizing the number of satisfied clauses. Certificates of optimality play a key role in the verification of MaxSAT solutions and are commonly studied through proof systems, with MaxSAT resolution being the most studied inference rule. In this paper, we study the problem of computing minimum-length refutations, measured by the number of inference steps, under the MaxSAT resolution rule. We propose a SAT-based approach that uses propositional encodings to model and search for MaxSAT resolution refutations of fixed size, while respecting structural, consumption, and compensation constraints imposed by the rule.

Keywords

Maximum Satisfiability, MaxSAT resolution, Refutation

1 Introduction

La satisfiabilité propositionnelle (SAT) et la satisfiabilité maximum (MaxSAT) sont deux formalismes puissants, avec une grande capacité d'expression en logique propositionnelle [4]. Étant donné une formule propositionnelle en forme normale conjonctive (FNC), le problème SAT consiste à déterminer s'il existe une affectation des variables qui satisfait toutes les clauses de la formule. MaxSAT est une extension naturelle de SAT sous forme de problème d'optimisation. Son objectif est de déterminer le nombre maximal de clauses pouvant être satisfaites par une affectation des variables.

L'importance pratique de MaxSAT a fortement augmenté ces dernières années. Ses applications couvrent plusieurs domaines, notamment la vérification matérielle et logicielle [13, 29], les problèmes de planification et d'ordonnancement [9, 30], ainsi que la bioinformatique [11, 12], parmi d'autres. Cependant, malgré ces avancées, MaxSAT reste fondamentalement plus difficile à résoudre que SAT, à la fois sur le plan théorique et pratique. Cette difficulté vient de la complexité plus élevée liée à la recherche de solutions et, surtout, à la preuve de leur optimalité.

L'étude des systèmes de preuve pour MaxSAT est devenue un sujet majeur pour mieux comprendre la complexité et les possibilités de résolution efficace des problèmes d'optimisation en logique propositionnelle. Un système de preuve clausal pour MaxSAT est généralement défini par un ensemble de règles d'inférence, dont les prémisses et les conclusions sont exprimées sous forme d'ensembles de clauses. Dans ce cadre, une preuve d'optimalité, souvent appelée certificat d'optimalité, fournit une garantie formelle que le coût retourné par un solveur MaxSAT correspond effectivement à l'optimum réel de la formule donnée en entrée.

Bien que les réfutations fondées sur la résolution sont bien établies pour SAT, l'étude systématique et la génération pratique de tels certificats dans le cadre de MaxSAT ont reçu relativement peu d'attention jusqu'à récemment [2, 26, 27, 31, 32]. Dans ce contexte, la max-résolution [6], qui généralise la règle de résolution classique pour SAT [28], est l'un des systèmes de preuve les plus étudiés pour MaxSAT. En particulier, cette règle a été largement utilisée dans la résolution effective de MaxSAT, notamment dans les algorithmes de type séparation et évaluation (*Branch and Bound*) pour MaxSAT [7, 20, 21], ainsi que dans les approches MaxSAT basées sur des appels itératifs à des oracles SAT [14, 24]. Des avancées récentes ont également montré que la max-résolution, lorsqu'elle est complétée par des règles d'inférence simples, peut produire des preuves de taille polynomiale pour le principe des tiroirs [18, 5].

Cependant, il existe des différences fondamentales entre la max-résolution et la résolution classique utilisée dans SAT. En effet, dans SAT, l'insatisfiabilité peut être démontrée

par une réfutation composée d'une suite d'étapes de résolution qui permettent de dériver la clause vide. La règle de résolution classique permet à plusieurs étapes d'inférence de partager les mêmes clauses prémisses, car ces clauses restent dans la formule après chaque inférence. En revanche, les suites d'étapes de max-résolution sont beaucoup plus contraintes. En effet, la max-résolution remplace les clauses prémisses par les conclusions dérivées. Cette différence de fonctionnement est souvent décrite comme un « mouvement de connaissances ».[17], modifie profondément la structure et les propriétés des dérivations dans les preuves pour MaxSAT. De plus, afin de garantir la correction du raisonnement, la construction de preuves par max-résolution nécessite des mécanismes de compensation supplémentaires, mis en place par l'introduction de clauses de compensation. Ces exigences mettent en évidence un écart fondamental dans notre compréhension de la relation entre les systèmes d'inférence SAT et MaxSAT [5, 8, 25].

Des travaux récents dans [23] ont proposé une approche basée sur SAT pour calculer les plus courtes preuves de résolution dans le cadre de SAT. Dans cet article, nous lançons une étude préliminaire visant à étendre cette approche au cadre de la max-résolution. En nous appuyant sur le cadre proposé dans [23], nous adaptons l'encodage basé sur SAT afin de prendre en compte les contraintes supplémentaires et les propriétés structurelles propres à la max-résolution. En particulier, nous proposons un cadre pour calculer les plus courtes réfutations sous la règle de max-résolution, en encodant la recherche de réfutations MaxSAT de taille fixée sous forme de problèmes SAT. Notre objectif est d'établir des bornes précises sur la longueur des preuves d'optimalité pour les instances MaxSAT. Cela permet de combler une lacune identifiée dans la littérature et de construire des certificats plus courts et plus faciles à interpréter. Plus largement, ce travail contribue à une meilleure compréhension de la complexité des preuves dans les variantes d'optimisation de la satisfiabilité. Il soutient également le développement de certificats d'optimalité plus courts et plus efficaces pour les solveurs MaxSAT.

Le reste de cet article est organisé comme suit. La section 2 introduit le problème (Max-)SAT et présente la règle de max-résolution. La section 3 présente notre méthodologie pour la recherche de réfutations minimales. La section 4 détaille l'encodage SAT utilisé pour calculer des réfutations par max-résolution de longueur fixée. La section 5 présente nos expérimentations préliminaires ; enfin, nous concluons et discutons des perspectives de recherche dans la section 6.

2 Preliminaries

2.1 Définitions et notations

Soit $X = \{x_1, x_2, \ldots, x_n\}$ l'ensemble des variables propositionnelles. Un littéral l est soit une variable $x \in X$, soit sa négation $\neg x$ également notée $\overline{x}$. Une *clause* $c = (l_1 \vee l_2 \vee \cdots \vee l_k)$ est une disjonction de k littéraux, et peut aussi être représentée comme un ensemble de littéraux. Une *formule en forme normale conjonctive (FNC)* $\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_m$ est une conjonction de m clauses, et peut être représentée comme un ensemble de clauses. Une *affectation* $\alpha : X \to \{0, 1\}$ associe chaque variable à une valeur booléenne et peut être représentée comme un ensemble de littéraux satisfaits. Un littéral l est *satisfait* par une affectation α si $l \in \alpha$, et il est *falsifié* si $\neg l \in \alpha$. Une clause c est *satisfaite* par une affectation α si au moins l'un de ses littéraux est satisfait par α ; sinon, elle est *falsifiée* par α. La *clause vide*, notée $\square$, ne contient aucun littéral et ne peut être satisfaite par aucune affectation. Deux clauses c et c' sont dites *opposées* s'il existe un littéral $l \in c$ tel que $\neg l \in c'$. Nous notons par $var(c)$ l'ensemble des variables qui apparaissent dans la clause c. Une formule FNC ϕ est *satisfaite* par une affectation α appelée *modèle* de ϕ, si toutes les clauses de ϕ sont satisfaites par α ; sinon, ϕ est *falsifiée*. Une affectation α est dite *complète* si chaque variable de ϕ reçoit une valeur ; sinon, elle est dite *partielle*. Le problème de satisfiabilité (SAT) consiste à déterminer s'il existe une affectation des variables qui satisfait une formule FNC donnée ϕ. Si une telle affectation existe, ϕ est dite *satisfiable* ; sinon, elle est dite *insatisfiable*.

Le problème de satisfiabilité maximum (MaxSAT) est une extension naturelle de SAT sous forme de problème d'optimisation. Il consiste simplement, étant donnée une formule FNC ϕ, à déterminer le nombre maximal de clauses qui peuvent être satisfaites par une affectation des variables. Une définition plus courante dans la littérature consiste à déterminer le nombre minimal de clauses que toute affectation de ϕ doit falsifier ; ce nombre est appelé l'optimum de ϕ. Formellement, soit $cost_\alpha(\phi)$ le nombre de clauses falsifiées par l'affectation α dans ϕ. L'*optimum* de ϕ est défini par $opt(\phi) = \min_\alpha cost_\alpha(\phi)$. Notons que si une formule FNC ϕ est satisfiable, alors $opt(\phi) = 0$; sinon, $opt(\phi) \geq 1$, car au moins une clause est falsifiée par chaque affectation de ϕ.

2.2 Réfutations par max-résolution

Pour certifier qu'une formule FNC est satisfiable, il suffit de donner un modèle de cette formule. Cependant, pour prouver qu'une formule est insatisfiable, il faut réfuter l'existence de tout modèle. Le système de réfutation le plus connu pour SAT est basé sur la règle de *résolution* [10]. Cette règle permet de dériver une clause appelée *résolvante* à partir de deux clauses opposées sur un littéral, comme défini ci-dessous. Cette résolvante peut être ajoutée à la formule tout en préservant sa satisfaisabilité. Ainsi, une *réfutation par résolution* d'une formule insatisfiable donnée ϕ est une suite finie d'étapes de résolution qui commence avec des clauses de ϕ et qui finit par dériver la clause vide $\square$. Les réfutations par résolution peuvent être représentées sous forme de *graphe orienté acyclique (DAG)*, où les nœuds représentent les clauses et les arcs représentent les dépendances entre les résolvantes et leurs prémisses.

Definition 1 (Résolution [10])**.** *Étant donné deux clauses $c_1 = (x \vee A)$ et $c_2 = (\neg x \vee B)$ opposées sur la variable x, où A et B sont des disjonctions de littéraux, la règle de résolution permet de dériver une clause c_r appelée la*

résolvante, comme suit :

$$\frac{c_1 = (x \vee A) \quad c_2 = (\neg x \vee B)}{c_r = (A \vee B)}$$

Contrairement à SAT, où les systèmes de preuve cherchent à démontrer l'insatisfiabilité, les systèmes de preuve pour MaxSAT ont un objectif différent. Ils permettent de calculer l'*optimum* d'une formule, c'est-à-dire le nombre minimal de clauses falsifiées parmi toutes les affectations possibles. Plusieurs systèmes de preuve complets pour MaxSAT ont été proposés dans la littérature, notamment la max-résolution [28] et les tableaux sémantiques [22]. Dans ces systèmes, la formule est réfutée un nombre de fois égal à son optimum, au moyen de transformations qui préservent l'équivalence, comme défini ci-dessous.

Definition 2 (Équivalence MaxSAT)**.** *Deux formules FNC ϕ et ϕ' sont équivalentes au sens de MaxSAT si, pour toute affectation $\alpha : var(\phi) \cup var(\phi') \rightarrow \{0,1\}$, nous avons $cost_\alpha(\phi) = cost_\alpha(\phi')$.*

L'un des premiers systèmes complets, et aussi l'un des plus étudiés pour MaxSAT, est basé sur la règle de max-résolution. Cette règle étend la règle de résolution classique de SAT au cadre de MaxSAT. La max-résolution, définie formellement ci-dessous, prend comme prémisses un ensemble de clauses opposées et déduit une résolvante, comme dans la résolution classique. Cependant, deux différences fondamentales entre la max-résolution et la résolution classique apparaissent afin de garantir la correction de la règle, au sens de l'équivalence MaxSAT. Premièrement, d'autres clauses sont dérivées en même temps que la résolvante. Ces clauses sont appelées clauses de compensation et doivent être ajoutées à la formule après l'application de la règle afin de préserver l'équivalence. Deuxièmement, contrairement à la résolution classique, où les prémisses sont conservées après chaque étape d'inférence, la max-résolution consomme ses prémisses. Autrement dit, une fois les clauses prémisses résolues, elles sont remplacées par les conclusions dérivées, formées par la résolvante et les clauses de compensation. Elles ne peuvent donc plus être utilisées comme prémisses dans des dérivations ultérieures sous leur forme initiale.

Definition 3 (Max-résolution [16])**.** *Étant donné deux clauses $C_1 = x \vee A$ et $C_2 = \neg x \vee B$ opposées sur la variable x, où $A = a_1 \vee \cdots \vee a_s$ et $B = b_1 \vee \cdots \vee b_t$, la règle de max-résolution est définie comme suit :*

$$\frac{C_1 = x \vee A \qquad C_2 = \neg x \vee B}{C_r = A \vee B}$$

$$CC_1 = x \vee A \vee \neg b_1$$
$$CC_2 = x \vee A \vee b_1 \vee \neg b_2$$
$$\vdots$$
$$CC_t = x \vee A \vee b_1 \vee \cdots \vee b_{t-1} \vee \neg b_t$$
$$CC_{t+1} = \neg x \vee B \vee \neg a_1$$
$$CC_{t+2} = \neg x \vee B \vee a_1 \vee \neg a_2$$
$$\vdots$$
$$CC_{t+s} = \neg x \vee B \vee a_1 \vee \cdots \vee a_{s-1} \vee \neg a_s$$

Notons que la max-résolution, telle qu'elle est présentée dans la définition précédente, dépend de l'ordre des littéraux. Autrement dit, selon l'ordre des littéraux dans les clauses prémisses, les clauses de compensation dérivées peuvent avoir des formes différentes. Cependant, toutes ces formes sont équivalentes. Nous pouvons donc les représenter de manière abstraite à l'aide d'une écriture plus compacte de la règle, comme suit :

$$\frac{c_1 = (x \vee A) \qquad c_2 = (\neg x \vee B)}{\begin{array}{ll} c_r = A \vee B & \text{(Résolvante)} \\ CC_1 \equiv x \vee A \vee \overline{B} & \text{(Ensemble de compensation 1)} \\ CC_2 \equiv \neg x \vee B \vee \overline{A} & \text{(Ensemble de compensation 2)} \end{array}}$$

où $\overline{A} = (\overline{a_1}) \wedge (a_1 \vee \overline{a_2}) \wedge \cdots \wedge (a_1 \vee \cdots \vee a_{s-1} \vee \overline{a_s})$, et de manière similaire pour B.

Une *réfutation par max-résolution*, ou simplement *max-réfutation*, est une suite finie d'étapes de max-résolution qui commence avec les clauses de ϕ et dérive une formule ϕ' telle que $\square \in \phi'$. De telles preuves peuvent aussi être représentées sous forme de DAG, où les nœuds représentent les clauses et les arcs représentent les dépendances d'inférence. La *taille* d'une réfutation est le nombre de ses étapes d'inférence, c'est-à-dire le nombre total de clauses résolvantes générées par la max-résolution.

Example 1. *Considérons la formule FNC $\phi = \{(\neg x_1 \vee x_3), (x_1), (\neg x_1 \vee x_2), (\neg x_2 \vee \neg x_3)\}$. La figure 1 illustre deux max-réfutations valides issues de [8]. Elles se distinguent par l'ordre dans lequel sont effectuées les étapes de résolution impliquant la clause non-read-once x_1, avec des tailles égales à 4 pour la dérivation de gauche et à 5 pour celle de droite.*

3 Méthodologie

Dans cette section, nous nous intéressons au problème de la recherche des plus courtes réfutations par max-résolution. Cette tâche est fondamentalement plus difficile que dans le cadre de la résolution classique étudié dans [23], en raison du caractère consommant de la max-résolution, de la présence de contraintes de compensation et de sa sensibilité à l'ordre des étapes d'inférence. À ce stade, nous nous limitons aux plus courtes réfutations, plutôt qu'aux plus courtes preuves d'optimalité de MaxSAT. Néanmoins, notre approche ouvre des pistes prometteuses pour traiter ce dernier problème. Contrairement à la résolution classique, où les prémisses peuvent être réutilisées et où l'ordre des littéraux n'a pas d'importance, la max-résolution impose des contraintes strictes sur ces deux aspects. Cela rend la recherche systématique de preuves de longueur minimale non triviale. De manière similaire à [23], nous abordons le problème du calcul des plus courtes réfutations par max-résolution en développant une méthodologie d'encodage basée sur SAT. Nous expliquons maintenant l'idée et les hypothèses centrales avant de présenter l'encodage formel. Tout d'abord, on veut décider si une formule FNC insatisfiable donnée F admet une réfutation par max-résolution

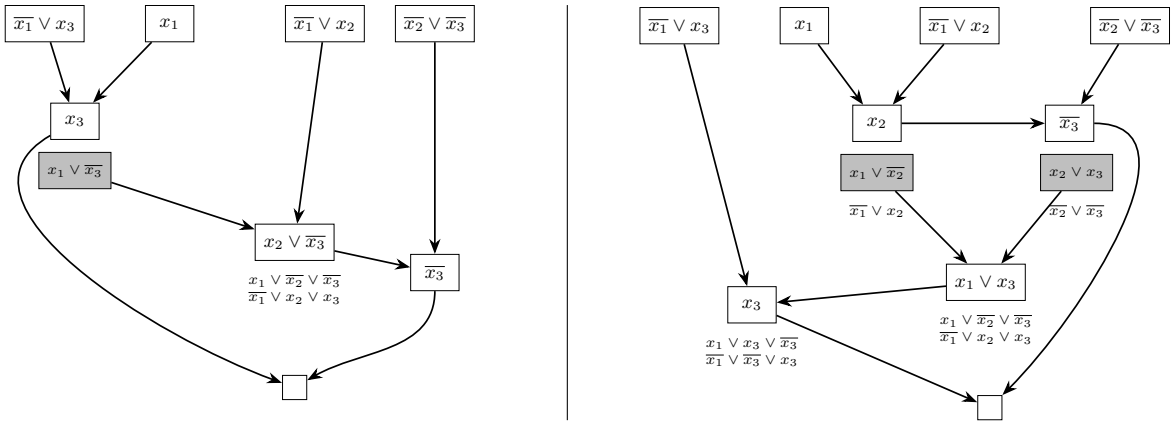

FIGURE 1 – Deux réfutations possibles par max-résolution pour une même formule. Les clauses utilisées sont indiquées dans des boîtes, et les clauses de compensation utilisées sont surlignées en gris.

Algorithm 1 Max-réfutation de longueur minimale

Require: Formule insatisfiable F

```
K ← 1
loop
    Φ_K ≡ ∃ une max-réfutation de F de taille K ?
    (résultat, modèle) ← SOLVE(Φ_K)
    if résultat = SAT then
        return (K, modèle)
    else
        K ← K + 1
    end if
end loop
```

avec exactement K étapes d'inférence. Pour cela, nous construisons une formule propositionnelle Φ_K telle que :

- si Φ_K est satisfiable, alors on peut extraire, à partir d'une affectation satisfaisante, une réfutation par max-résolution de F de taille K ;
- si Φ_K est insatisfiable, alors aucune telle réfutation n'existe dans l'abstraction capturée par l'encodage.

Comme la profondeur d'une plus courte réfutation n'est pas connue à l'avance, nous adoptons une approche itérative sur la longueur de preuve K, décrite dans l'algorithme 1. À chaque itération, nous encodons la formule Φ_K qui représente l'existence d'une max-réfutation de F de taille fixée K. Si le solveur retourne $UNSAT$, alors K constitue naturellement une borne inférieure stricte sur la taille minimale de la réfutation. Nous augmentons donc K de manière itérative, jusqu'à ce que le solveur retourne SAT. Dans ce cas, nous obtenons la taille optimale de la preuve, et nous pouvons également extraire la preuve calculée à partir de l'affectation retournée par le solveur.

Comme la max-résolution est plus complexe que la résolution classique, nous adoptons des des choix pour des fins de simplification, qui sont décrits dans la suite. Rappelons que la max-résolution est un système de preuve destructif : une fois qu'une clause est utilisée comme prémisse, elle est consommée et ne peut plus être réutilisée, sous sa forme inchangée, plus tard dans la dérivation. Pour représenter fidèlement ce comportement, notre encodage doit modéliser explicitement la disponibilité temporelle des clauses ainsi que leur consommation. De plus, à chaque étape d'inférence, une application de la max-résolution produit une résolvante et plusieurs clauses de compensation dérivées de chaque parent. La principale difficulté consiste donc à représenter ce qui reste disponible après chaque étape d'inférence. Les clauses standards sont simples à représenter, car elles sont soit encore disponibles, soit déjà consommées. Les clauses de compensation sont plus difficiles à gérer, car elles forment des ensembles structurés dont l'utilisation future dépend des littéraux déjà fixés par les étapes précédentes et des littéraux qui peuvent encore participer à des résolutions ultérieures. Pour capturer cela sans énumérer explicitement tous les descendants possibles des clauses de compensation, nous représentons chaque ensemble de compensation, comme dans la forme condensée de la max-résolution, par une paire (F, R), où :

- F est la *partie fixe*, contenant les littéraux déjà engagés et qui persistent dans toute clause future extraite de cet ensemble de compensation ;
- R est la *partie restante*, contenant les littéraux encore actifs et pouvant être consommés plus tard.

Intuitivement, F enregistre l'historique irréversible de l'ensemble de compensation, tandis que R représente son potentiel restant pour la recherche de preuve. Cette abstraction traite donc les clauses de compensation sous la forme $F \vee \overline{R}$, constituant ainsi le choix de modélisation central de notre cadre. En toute généralité, utiliser un ensemble de compensation comme parent peut le diviser en plusieurs descendants, selon la position du pivot et la clause extraite de l'ensemble. Encoder directement toutes ces possibilités conduirait à une forte explosion combinatoire. Pour cette raison, nous limitons notre attention à trois schémas de transition canoniques, suffisants pour notre procédure de recherche de preuve. L'un de ces cas repose en plus sur une *restriction de plus grande clause*, afin d'éviter qu'un parent de compensation ne génère plusieurs descendants simultanément. Cette restriction simplifie l'encodage, mais elle implique aussi que notre cadre calcule les plus courtes réfutations

dans le système de transition restreint que nous encodons. Plus précisément, lorsqu'un ensemble de compensation (F, R) est sélectionné comme parent à une étape d'inférence, la clause extraite doit contenir le pivot avec la bonne polarité pour le côté choisi de l'inférence : à gauche, la clause extraite doit contenir x, tandis qu'à droite elle doit contenir $\neg x$. La transition dépend alors de la manière dont ce pivot est obtenu à partir de l'ensemble de compensation. Pour éviter toute ambiguïté, nous adoptons la convention suivante dans cette section : x désigne toujours la variable pivot sélectionnée à l'étape de résolution courante, tandis que y désigne un littéral non pivot choisi dans la partie restante R lorsqu'un tel choix est nécessaire. De plus, sauf mention contraire, nous supposons que le parent gauche fournit le pivot positivement et que le parent droit le fournit négativement. Sous cette convention, nous distinguons trois cas canoniques, selon la position du pivot dans la structure de compensation.

Cas 1 : le pivot est déjà dans la partie fixe. Supposons d'abord que l'ensemble de compensation soit utilisé comme parent gauche. La clause extraite doit donc contenir le pivot positif x. Si $x \in F$, alors le pivot apparaît déjà dans toutes les clauses représentées par l'objet de compensation. Dans ce cas, il n'est pas nécessaire de consommer le pivot depuis la partie restante. À la place, nous consommons un autre littéral $y \in R$, avec $y \neq x$, et nous extrayons la clause

$$F \vee x \vee \neg y.$$

L'ensemble de compensation mis à jour (F', R') est donné par

$$F' = F, \qquad R' = R \setminus \{y\}.$$

Intuitivement, le pivot est hérité de la partie fixe. La transition réduit donc seulement la partie restante. C'est le cas le plus simple, car le pivot ne modifie pas la structure interne de l'ensemble de compensation.

Example 2. *Soient $F = \{x, a\}$ et $R = \{b, c\}$, et supposons que le parent gauche doive fournir le pivot x. Comme $x \in F$, nous pouvons choisir $y = b$ et extraire la clause $x \vee a \vee \neg b$. L'ensemble de compensation obtenu est $(\{x, a\}, \{c\})$.*

Cas 2 : le pivot est dans la partie restante avec la polarité opposée. Supposons encore que l'ensemble de compensation soit utilisé comme parent gauche et qu'il doit fournir le pivot positif x. Si $x \notin F$ mais $\neg x \in R$ sous une forme qui permet une extraction directe, alors le pivot est obtenu en consommant ce littéral depuis la partie restante. La clause extraite est

$$F \vee x.$$

L'ensemble de compensation mis à jour (F', R') est alors donné par

$$F' = F \cup \{\neg x\}, \qquad R' = R \setminus \{\neg x\}.$$

Intuitivement, le littéral pris dans la partie restante est consommé et devient une partie de l'historique fixe avec la polarité opposée. Cela reflète le fait qu'une fois que l'ensemble de compensation a été utilisé pour fournir le pivot à partir de la partie restante, ce choix devient irréversible.

Example 3. *Soient $F = \{a\}$ et $R = \{\neg x, b\}$, et supposons que le parent gauche doit fournir le pivot x. Nous extrayons la clause $a \vee x$. L'ensemble de compensation mis à jour devient $(\{a, \neg x\}, \{b\})$.*

Cas 3 : le pivot est dans la partie restante avec la même polarité. La situation la plus délicate apparaît lorsque l'ensemble de compensation est utilisé comme parent gauche, que la clause extraite doit contenir x, et que x n'est pas déjà disponible dans la partie fixe, mais apparaît avec la même polarité dans la partie restante, c'est-à-dire $x \in R$. Extraire x de la partie restante peut alors nécessiter de diviser l'ensemble de compensation en plusieurs descendants. Pour garder l'encodage compact, nous ne représentons pas ce cas dans toute sa généralité. À la place, nous limitons la transition à la forme de *plus grande clause*

$$F \vee x \vee (R \setminus \{y\}) \vee \neg y,$$

pour un certain littéral choisi $y \in R$, puis nous mettons à jour l'ensemble de compensation comme suit :

$$F' = F, \qquad R' = R \setminus \{y\}.$$

Intuitivement, ce cas simule l'effet d'une consommation dans la partie restante sans générer explicitement plusieurs objets de compensation successifs. La restriction de la plus grande clause n'est donc pas une propriété intrinsèque de la max-résolution, mais plutôt une restriction de notre cadre d'encodage introduite pour contrôler l'espace de recherche.

Example 4. *Soient $F = \{a\}$ et $R = \{x, b, c\}$, et supposons que le parent gauche doit fournir le pivot x. Au lieu de diviser la structure de compensation, nous choisissons $y = b$ et nous extrayons la clause*

$$a \vee x \vee c \vee \neg b.$$

L'ensemble de compensation mis à jour est donc $(\{a\}, \{x, c\})$.

Supposons maintenant que nous extrayions plutôt la clause $a \vee x \vee \neg b$, c'est-à-dire sans inclure les autres littéraux restants de R. Cette clause correspond à l'utilisation d'un préfixe partiel de la partie restante. Dans ce cas, les clauses restantes, c'est-à-dire $a \vee \neg x$ et $a \vee x \vee c \vee \neg b$, ne peuvent pas être représentées par un seul ensemble de compensation successeur : il faut distinguer les clauses qui incluent c de celles qui ne l'incluent pas. Cela conduit à une division *de l'ensemble de compensation en plusieurs descendants, chacun représentant un sous-ensemble différent des littéraux restants. En revanche, l'extraction de la plus grande clause détermine de manière unique la structure résiduelle, qui peut être représentée par un seul ensemble de compensation.*

Nous remarquons que les cas 1 et 2 ne nécessitent jamais de division : dans le cas 1, le pivot est déjà présent dans la partie fixe et donc disponible globalement, tandis que dans le cas 2, le pivot complémentaire est directement retiré de la partie restante et transféré vers la partie fixe. Seul le cas 3, où x apparaît positivement dans la partie restante R sans être déjà fixé, introduit un choix de branchement sur la quantité de R à déplier avant d'utiliser x. La restriction de la plus grande clause résout cette ambiguïté en sélectionnant une seule option canonique, ce qui garantit que chaque parent de compensation produit exactement un seul descendant de son côté. Les deux premiers cas correspondent directement à des manières naturelles d'extraire le pivot de la partie fixe et la partie restante. Le troisième cas est une restriction volontaire utilisée pour éviter un branchement sur plusieurs descendants de compensation. Par conséquent, la minimalité dans notre cadre est régie par la sémantique de transition de la forme de plus grande clause, ce qui peut entraîner un surcoût dans la taille des preuves générées.

4 Modélisation formelle des max-réfutations de longueur fixée

Avant de présenter les contraintes une par une, nous donnons d'abord une vue d'ensemble de l'encodage. Un modèle de Φ_K représente une réfutation par max-résolution de taille exactement K, dans le système de transition restreint de la section 3. Plus précisément, pour chaque étape $t \in \{1, \dots, K\}$, le modèle spécifie : (i) une variable pivot, (ii) un parent gauche et un parent droit, chacun pouvant être soit une clause standard, soit un ensemble de compensation, (iii) le contenu de la résolvante produite à l'étape t, (iv) les objets de compensation générés à cette étape, et (v) l'état de disponibilité de tous les objets pour les étapes futures. Pour capturer ces informations, nous introduisons d'abord plusieurs familles de variables, puis nous organisons les contraintes en couches sémantiques, afin de garantir que chaque étape correspond à une inférence valide.

4.1 Univers et variables

Afin de garantir un indexage correct, nous utilisons les univers suivants :

- $\mathcal{X}$: ensemble des variables de la formule.
- $\mathcal{S} = \{1, \dots, K\}$: indices des étapes d'inférence.
- $\mathcal{D} = \{1, \dots, M + K\}$: indices des clauses *standards*, où $\mathcal{I} = \{1, \dots, M\}$ sont les indices des clauses initiales de la formule d'entrée, et $\mathcal{J} = \{M + 1, \dots, M + K\}$ sont les indices des résolvantes générées pendant la dérivation.
- $\mathcal{C} = \{1, \dots, 2K\}$: indices des ensembles de compensation, où les ensembles générés à l'étape t sont indexés par $2t-1$ pour l'ensemble de compensation gauche, et par $2t$ pour l'ensemble de compensation droit.

À l'étape t, seuls les objets créés strictement avant t peuvent être sélectionnés comme parents. Nous définissons donc les ensembles disponibles suivants afin de garantir la cohérence temporelle :

$$\mathcal{D}^t = \{i \in \mathcal{D} \mid i < M + t\}$$

$$\mathcal{C}^t = \{k \in \mathcal{C} \mid k < 2t - 1\}.$$

Nous pouvons maintenant définir les variables propositionnelles utilisées dans l'encodage :

Variables de sélection.

- $v_{x,t}$: la variable $x \in \mathcal{X}$ est sélectionnée comme pivot à l'étape $t \in \mathcal{S}$.
- $ls_{t,i}, rs_{t,i}$: à l'étape t, la clause standard $i \in \mathcal{D}^t$ est sélectionnée comme parent gauche/droit.
- $lcc_{t,k}, rcc_{t,k}$: à l'étape t, l'ensemble de compensation $k \in \mathcal{C}^t$ est sélectionné comme parent gauche/droit.

Variables de contenu.

- $ps_{x,i}, ns_{x,i}$: le littéral x, respectivement $\neg x$, apparaît dans la clause standard $i \in \mathcal{D}$.
- $pccF_{x,k,t}, nccF_{x,k,t}$: le littéral x, respectivement $\neg x$, appartient à la partie fixe de l'ensemble de compensation k à l'étape t.
- $pccR_{x,k,t}, nccR_{x,k,t}$: le littéral x, respectivement $\neg x$, appartient à la partie restante de l'ensemble de compensation k à l'étape t.

Variables de disponibilité.

- $as_{t,i}$: la clause standard i est disponible à l'étape t.
- $acc_{t,k}$: l'ensemble de compensation k est disponible à l'étape t.
- $accx_{x,t,k}$: à l'étape t, un littéral sur la variable x est consommé depuis la partie restante de l'ensemble de compensation k.

4.2 Sémantique de sélection

Nous imposons d'abord la validité structurelle de la dérivation. À chaque étape, exactement une variable pivot est sélectionnée, avec exactement un parent gauche et exactement un parent droit. Les parents peuvent être soit des clauses standards, soit des ensembles de compensation. De plus, le même objet ne peut pas être sélectionné des deux côtés de l'inférence.

$$\forall t \in \mathcal{S} :$$

$$\sum_{x \in \mathcal{X}} v_{x,t} = 1 \tag{1}$$

$$\sum_{i \in \mathcal{D}^t} ls_{t,i} + \sum_{k \in \mathcal{C}^t} lcc_{t,k} = 1 \tag{2}$$

$$\sum_{i \in \mathcal{D}^t} rs_{t,i} + \sum_{k \in \mathcal{C}^t} rcc_{t,k} = 1 \tag{3}$$

$$\overline{ls_{t,i}} \vee \overline{rs_{t,i}} \qquad \forall i \in \mathcal{D}^t \tag{4}$$

$$\overline{lcc_{t,k}} \vee \overline{rcc_{t,k}} \qquad \forall k \in \mathcal{C}^t \tag{5}$$

Pour chaque parent de compensation sélectionné, au plus une variable peut être consommée depuis sa partie restante. Cela impose qu'un objet de compensation évolue selon une seule transition à chaque étape d'inférence.

$$\forall t \in \mathcal{S}, \forall k \in \mathcal{C}^t :$$
$$\sum_{x \in \mathcal{X}} accx_{x,t,k} \leq 1. \quad (6)$$

Enfin, un objet ne peut être sélectionné comme parent que s'il est disponible à l'étape courante. Cette condition s'applique à la fois aux clauses standards et aux ensembles de compensation.

$$(ls_{t,i} \vee rs_{t,i}) \implies as_{t,i} \qquad \forall t \in \mathcal{S}, \forall i \in \mathcal{D}^t \quad (7)$$
$$(lcc_{t,k} \vee rcc_{t,k}) \implies acc_{t,k} \qquad \forall t \in \mathcal{S}, \forall k \in \mathcal{C}^t \quad (8)$$

Les contraintes (1)–(8) garantissent la cohérence structurelle de chaque étape d'inférence. La contrainte (1) sélectionne exactement une variable pivot. Les contraintes (2) et (3) sélectionnent respectivement un parent gauche et un parent droit. Les contraintes (4) et (5) empêchent la même clause standard ou le même ensemble de compensation d'être sélectionné des deux côtés de la même inférence. La contrainte (6) garantit qu'un parent de compensation sélectionné consomme au plus une variable depuis sa partie restante. Enfin, les contraintes (7) et (8) garantissent que seuls les objets disponibles peuvent être sélectionnés comme parents.

4.3 Sémantique FNC

Nous encodons maintenant le contenu des clauses standards et assurons la cohérences des objets traités. D'abord, pour chaque clause initiale $i \in \mathcal{I}$ et chaque variable $x \in \mathcal{X}$, la valeur de vérité des variables de contenu correspondantes est fixée par la formule d'entrée.

$$ps_{x,i} \quad \text{si } x \in C_i, \quad \overline{ps_{x,i}} \quad \text{sinon,} \quad (9)$$
$$ns_{x,i} \quad \text{si } \neg x \in C_i, \quad \overline{ns_{x,i}} \quad \text{sinon.} \quad (10)$$

Ensuite, nous imposons la cohérence des polarités : aucune variable ne peut apparaître avec les deux polarités dans la même clause standard, dans la même partie fixe ou dans la même partie restante.

$$\overline{ps_{x,i}} \vee \overline{ns_{x,i}} \qquad \forall i \in \mathcal{J}, \forall x \in \mathcal{X} \quad (11)$$
$$\overline{pccF_{x,k,t}} \vee \overline{nccF_{x,k,t}} \qquad \forall t \in \mathcal{S}, \forall k \in \mathcal{C}, \forall x \in \mathcal{X} \quad (12)$$
$$\overline{pccR_{x,k,t}} \vee \overline{nccR_{x,k,t}} \qquad \forall t \in \mathcal{S}, \forall k \in \mathcal{C}, \forall x \in \mathcal{X} \quad (13)$$

Enfin, nous imposons la séparation entre la partie fixe et la partie restante. Autrement dit, un littéral peut appartenir soit à la partie fixe, soit à la partie restante d'un ensemble de compensation, mais jamais aux deux.

$$\forall x \in \mathcal{X}, \forall t \in \mathcal{S}, \forall k \in \mathcal{C} :$$
$$\overline{pccF_{x,k,t}} \vee \overline{pccR_{x,k,t}} \quad (14)$$
$$\overline{pccF_{x,k,t}} \vee \overline{nccR_{x,k,t}} \quad (15)$$
$$\overline{nccF_{x,k,t}} \vee \overline{pccR_{x,k,t}} \quad (16)$$
$$\overline{nccF_{x,k,t}} \vee \overline{nccR_{x,k,t}} \quad (17)$$

Les contraintes (9) et (10) initialisent le contenu des clauses d'entrée selon la formule originale. Les contraintes (11)–(13) imposent une cohérence locale des polarités, en garantissant qu'aucune variable n'apparaît avec les deux polarités dans une même clause standard, une même partie fixe ou une même partie restante. Enfin, les contraintes (14)–(17) garantissent que les parties fixe et restante de chaque ensemble de compensation sont disjointes.

4.4 Sémantique d'inférence

Dans cette section, nous imposons la sémantique de la max-résolution. Par convention, dans la suite de l'encodage, la polarité du pivot est fixée comme suit : le parent gauche doit contenir la variable pivot x avec une polarité positive, tandis que le parent droit doit la contenir avec une polarité négative. Pour les clauses standards, cette exigence est directe : si une clause standard est sélectionnée comme parent gauche, elle doit contenir x ; si elle est sélectionnée comme parent droit, elle doit contenir $\neg x$.

$$\forall t \in \mathcal{S}, \forall x \in \mathcal{X}, \forall i \in \mathcal{D}^t :$$
$$(ls_{t,i} \wedge v_{x,t}) \implies ps_{x,i} \quad (18)$$
$$(rs_{t,i} \wedge v_{x,t}) \implies ns_{x,i} \quad (19)$$

Pour les ensembles de compensation, la situation est plus subtile, car un objet de compensation (F, R) représente une famille de clauses plutôt qu'une seule clause. Ainsi, les contraintes suivantes ne doivent pas être interprétées comme imposant que plusieurs littéraux apparaissent simultanément dans une même clause concrète. Elles expriment plutôt une *condition d'admissibilité* : un ensemble de compensation peut servir de parent pour le pivot x lorsqu'il contient assez d'information pour extraire une clause concrète où le pivot apparaît avec la polarité requise par le côté sélectionné.

Plus précisément, le pivot peut être justifié par l'un des trois cas de transition introduits dans la section 3 :

- le pivot est déjà présent dans la partie fixe ($pccF_{x,k,t-1}$ pour le parent gauche, $nccF_{x,k,t-1}$ pour le parent droit), ce qui correspond au cas 1 ;
- le pivot apparaît dans la partie restante avec la polarité requise par le côté sélectionné ($pccR_{x,k,t-1}$ pour le parent gauche, $nccR_{x,k,t-1}$ pour le parent droit), ce qui correspond au cas 3 ;
- le pivot complémentaire apparaît dans la partie restante ($nccR_{x,k,t-1}$ pour le parent gauche, $pccR_{x,k,t-1}$ pour le parent droit), ce qui correspond au cas 2.

Les contraintes suivantes imposent cette condition d'admissibilité pour les parents de compensation.

$$\forall t \in \mathcal{S} \setminus \{1\}, \forall x \in \mathcal{X}, \forall k \in \mathcal{C}^t :$$
$$(lcc_{t,k} \wedge v_{x,t}) \implies$$
$$\left(pccF_{x,k,t-1} \vee pccR_{x,k,t-1} \vee nccR_{x,k,t-1}\right) \quad (20)$$
$$(rcc_{t,k} \wedge v_{x,t}) \implies$$
$$\left(nccF_{x,k,t-1} \vee pccR_{x,k,t-1} \vee nccR_{x,k,t-1}\right) \quad (21)$$

Les contraintes (20) et (21) garantissent donc qu'un ensemble de compensation sélectionné est admissible comme parent pour le pivot choisi. La clause concrète extraite de l'ensemble de compensation, ainsi que la mise à jour de sa représentation (F', R'), sont ensuite déterminées par les contraintes de transition introduites ci-dessous.
De plus, la variable pivot elle-même est éliminée par résolution. Elle ne doit donc pas apparaître dans la résolvante obtenue, ni dans la partie restante des ensembles de compensation générés, comme précisé dans les contraintes (22) à (24).

$$\forall t \in \mathcal{S}, \forall x \in \mathcal{X} :$$
$$v_{x,t} \implies \overline{ps_{x,M+t}} \wedge \overline{ns_{x,M+t}} \tag{22}$$
$$v_{x,t} \implies \overline{pccR_{x,2t-1,t}} \wedge \overline{nccR_{x,2t-1,t}} \tag{23}$$
$$v_{x,t} \implies \overline{pccR_{x,2t,t}} \wedge \overline{nccR_{x,2t,t}} \tag{24}$$

Nous nous focalisons maintenant sur la sémantique de transition après l'application d'une étape de max-résolution. Nous distinguons ici deux cas : (i) une clause standard comme parent et (ii) un ensemble de compensation comme parent.

(i) Clause standard comme parent. Si une clause standard est sélectionnée comme parent, le littéral pivot est transféré vers la partie fixe de l'ensemble de compensation correspondant, tandis que chaque littéral non pivot est copié à la fois dans la résolvante et dans la partie restante de cet ensemble de compensation. Les contraintes suivantes décrivent cette mise à jour pour une clause standard sélectionnée comme parent gauche.

$$\forall t \in \mathcal{S}, \forall i \in \mathcal{D}^t, \forall x \in \mathcal{X} :$$
$$(ls_{t,i} \wedge ps_{x,i} \wedge v_{x,t}) \implies pccF_{x,2t-1,t} \tag{25}$$
$$(ls_{t,i} \wedge ps_{x,i} \wedge \overline{v_{x,t}}) \implies \left(ps_{x,M+t} \wedge pccR_{x,2t-1,t}\right) \tag{26}$$
$$(ls_{t,i} \wedge ns_{x,i} \wedge \overline{v_{x,t}}) \implies \left(ns_{x,M+t} \wedge nccR_{x,2t-1,t}\right) \tag{27}$$
$$(ls_{t,i} \wedge \overline{ps_{x,i}} \wedge \overline{ns_{x,i}}) \implies \left(\overline{pccF_{x,2t-1,t}} \wedge \overline{nccF_{x,2t-1,t}} \wedge \overline{pccR_{x,2t-1,t}} \wedge \overline{nccR_{x,2t-1,t}}\right) \tag{28}$$

La contrainte (25) stocke le littéral pivot dans la partie fixe de l'ensemble de compensation gauche généré à l'étape t. Les contraintes (26) et (27) copient chaque littéral non pivot du parent gauche sélectionné à la fois dans la résolvante et dans la partie restante de l'ensemble de compensation gauche généré. Enfin, la contrainte (28) garantit que les variables absentes du parent sélectionné n'apparaissent pas dans l'ensemble de compensation gauche généré.
Les contraintes pour une clause standard sélectionnée comme parent droit sont obtenues de manière symétrique, en stockant le pivot négatif dans la partie fixe de l'ensemble de compensation droit $2t$, et en copiant tous les littéraux non pivots dans la résolvante et dans la partie restante correspondante.

(ii) Ensemble de compensation comme parent. Lorsqu'un ensemble de compensation k est sélectionné comme parent, une variable doit être consommée depuis sa partie restante. Avec la contrainte (6), introduite dans la sémantique de sélection, les contraintes suivantes imposent qu'exactement une telle variable soit consommée. De plus, une variable ne peut être consommée que si au moins l'une de ses polarités apparaît dans la partie restante de l'ensemble de compensation sélectionné.

$$\forall t \in \mathcal{S}, \forall k \in \mathcal{C}^t :$$
$$lcc_{t,k} \implies \bigvee_{x \in \mathcal{X}} accx_{x,t,k} \tag{29}$$
$$rcc_{t,k} \implies \bigvee_{x \in \mathcal{X}} accx_{x,t,k} \tag{30}$$

$$\forall t \in \mathcal{S}, \forall k \in \mathcal{C}^t, \forall x \in \mathcal{X} :$$
$$accx_{x,t,k} \implies \left(pccR_{x,k,t-1} \vee nccR_{x,k,t-1}\right) \tag{31}$$

Les contraintes (29) et (30) exigent qu'un parent de compensation sélectionné consomme une variable depuis sa partie restante. Combinées avec la contrainte (6), elles garantissent qu'exactement une variable est consommée. La contrainte (31) garantit que cette consommation est bien définie, c'est-à-dire que la variable consommée apparaît réellement dans la partie restante de l'ensemble de compensation sélectionné.
Si le pivot complémentaire est directement disponible dans la partie restante, il doit être consommé. En revanche, si le pivot apparaît dans la partie restante avec la polarité requise par le côté sélectionné, alors la consommation directe du pivot lui-même est interdite. Dans ce cas, la transition de plus grande clause doit être utilisée, et un autre littéral de la partie restante doit être consommé.

$$\forall t \in \mathcal{S}, \forall k \in \mathcal{C}^t, \forall x \in \mathcal{X} :$$
$$(lcc_{t,k} \wedge v_{x,t} \wedge nccR_{x,k,t-1}) \implies accx_{x,t,k} \tag{32}$$
$$(rcc_{t,k} \wedge v_{x,t} \wedge pccR_{x,k,t-1}) \implies accx_{x,t,k} \tag{33}$$
$$(lcc_{t,k} \wedge v_{x,t} \wedge pccR_{x,k,t-1}) \implies \overline{accx_{x,t,k}} \tag{34}$$
$$(rcc_{t,k} \wedge v_{x,t} \wedge nccR_{x,k,t-1}) \implies \overline{accx_{x,t,k}} \tag{35}$$
$$(lcc_{t,k} \wedge \overline{pccR_{x,k,t-1}} \wedge \overline{nccR_{x,k,t-1}}) \implies \overline{accx_{x,t,k}} \tag{36}$$
$$(rcc_{t,k} \wedge \overline{pccR_{x,k,t-1}} \wedge \overline{nccR_{x,k,t-1}}) \implies \overline{accx_{x,t,k}} \tag{37}$$
$$(\overline{lcc_{t,k}} \wedge \overline{rcc_{t,k}}) \implies \overline{accx_{x,t,k}} \tag{38}$$

Les contraintes (32) et (33) encodent le cas de consommation directe. Du côté gauche, le parent sélectionné doit fournir le pivot positif x. Ainsi, si $\neg x$ apparaît dans la partie restante, il s'agit du pivot complémentaire et il doit être consommé. Symétriquement, du côté droit, le parent sélectionné doit fournir le pivot négatif $\neg x$; donc, si x apparaît dans la partie restante, il doit être consommé. Les contraintes (34) et (35) interdisent la consommation directe du pivot lorsqu'il apparaît dans la partie restante avec la polarité requise par le côté sélectionné. Cela correspond à la transition de plus grande clause : le pivot reste disponible

dans la clause extraite, et un autre littéral de la partie restante doit être consommé à sa place. Les contraintes (36) et (37) indiquent qu'aucun littéral sur la variable x ne peut être consommé depuis un parent de compensation sélectionné si aucune polarité de x n'apparaît dans sa partie restante. Enfin, la contrainte (38) garantit qu'aucune consommation n'est associée à un objet de compensation qui n'est pas sélectionné comme parent à l'étape t.

Les contraintes suivantes, c'est-à-dire (39) à (42), décrivent comment la partie fixe d'un parent de compensation gauche sélectionné est propagée. Si un littéral appartient déjà à la partie fixe de l'ensemble de compensation sélectionné, alors il reste fixé dans le nouvel ensemble de compensation gauche généré. De plus, si ce littéral ne porte pas sur la variable pivot, il est aussi copié dans la résolvante. Cela reflète le fait que les littéraux fixes sont présents dans toutes les clauses représentées par l'objet de compensation, et contribuent donc à la clause parent extraite, sauf s'ils sont éliminés comme pivot.

$$\forall t \in \mathcal{S}, \forall k \in \mathcal{C}^t, \forall x \in \mathcal{X} :$$
$$(lcc_{t,k} \wedge pccF_{x,k,t-1}) \implies pccF_{x,2t-1,t} \quad (39)$$
$$(lcc_{t,k} \wedge nccF_{x,k,t-1}) \implies nccF_{x,2t-1,t} \quad (40)$$
$$(lcc_{t,k} \wedge pccF_{x,k,t-1} \wedge \overline{v_{x,t}}) \implies ps_{x,M+t} \quad (41)$$
$$(lcc_{t,k} \wedge nccF_{x,k,t-1} \wedge \overline{v_{x,t}}) \implies ns_{x,M+t} \quad (42)$$

Les contraintes suivantes, (43) à (46), précisent l'effet de la consommation d'un littéral depuis la partie restante d'un parent de compensation gauche sélectionné. Lorsqu'un littéral positif sur la variable x est consommé depuis la partie restante, le littéral négatif correspondant est ajouté à la partie fixe du nouvel ensemble de compensation. Symétriquement, consommer un littéral négatif ajoute le littéral positif à la partie fixe. Si la variable consommée n'est pas le pivot, le littéral complémentaire est aussi ajouté à la résolvante. Ainsi, ces contraintes encodent la mise à jour irréversible de l'objet de compensation après la consommation d'un littéral de sa partie restante.

$$\forall t \in \mathcal{S}, \forall k \in \mathcal{C}^t, \forall x \in \mathcal{X} :$$
$$(accx_{x,t,k} \wedge pccR_{x,k,t-1}) \implies nccF_{x,2t-1,t} \quad (43)$$
$$(accx_{x,t,k} \wedge nccR_{x,k,t-1}) \implies pccF_{x,2t-1,t} \quad (44)$$
$$(accx_{x,t,k} \wedge pccR_{x,k,t-1} \wedge \overline{v_{x,t}}) \implies ns_{x,M+t} \quad (45)$$
$$(accx_{x,t,k} \wedge nccR_{x,k,t-1} \wedge \overline{v_{x,t}}) \implies ps_{x,M+t} \quad (46)$$

Les contraintes (47) et (48) garantissent que chaque littéral de la partie restante qui n'est pas consommé persiste dans la partie restante du nouvel ensemble de compensation gauche généré. Ainsi, la transition retire uniquement la variable consommée de la partie restante active, tout en conservant les autres littéraux disponibles pour les étapes d'inférence futures.

$$\forall t \in \mathcal{S}, \forall k \in \mathcal{C}^t, \forall x \in \mathcal{X} :$$
$$(lcc_{t,k} \wedge \overline{accx_{x,t,k}} \wedge pccR_{x,k,t-1}) \implies pccR_{x,2t-1,t} \quad (47)$$
$$(lcc_{t,k} \wedge \overline{accx_{x,t,k}} \wedge nccR_{x,k,t-1}) \implies nccR_{x,2t-1,t} \quad (48)$$

Les contraintes (49) et (50) encodent la transition de plus grande clause pour un parent de compensation gauche sélectionné. Lorsque le pivot est justifié par son occurrence dans la partie restante avec la polarité requise, la variable consommée n'est pas le pivot lui-même. Ainsi, tout autre littéral non consommé de la partie restante est copié dans la résolvante. Cela réalise l'extraction de la plus grande clause décrite dans la section 3, tout en évitant la création de plusieurs objets de compensation successeurs.

$$\forall t \in \mathcal{S}, \forall k \in \mathcal{C}^t, \forall x, y \in \mathcal{X},\ x \neq y :$$
$$(lcc_{t,k} \wedge \overline{accx_{x,t,k}} \wedge pccR_{x,k,t-1} \wedge v_{y,t} \wedge pccR_{y,k,t-1}) \implies ps_{x,M+t} \quad (49)$$
$$(lcc_{t,k} \wedge \overline{accx_{x,t,k}} \wedge nccR_{x,k,t-1} \wedge v_{y,t} \wedge pccR_{y,k,t-1}) \implies ns_{x,M+t} \quad (50)$$

Enfin, la contrainte (51) traite les variables qui n'apparaissent pas dans le parent de compensation sélectionné. Si aucune polarité d'une variable n'apparaît dans la partie fixe ni dans la partie restante de l'ensemble de compensation parent, alors aucune polarité n'est introduite dans le nouvel ensemble de compensation gauche généré. Cela empêche la création de littéraux parasites pendant la transition.

$$\forall t \in \mathcal{S}, \forall k \in \mathcal{C}^t, \forall x \in \mathcal{X} :$$
$$(lcc_{t,k} \wedge \neg pccF_{x,k,t-1} \wedge \neg nccF_{x,k,t-1} \wedge \neg pccR_{x,k,t-1} \wedge \neg nccR_{x,k,t-1}) \implies (\neg pccF_{x,2t-1,t} \wedge \neg nccF_{x,2t-1,t} \wedge \neg pccR_{x,2t-1,t} \wedge \neg nccR_{x,2t-1,t}) \quad (51)$$

Notons que les règles de transition omises pour un parent de compensation droit sont obtenues par symétrie avec le cas gauche.

4.5 Persistance conditionnelle et dynamique destructive

Si un ensemble de compensation n'est pas sélectionné comme parent à l'étape t, son contenu persiste à l'étape $t+1$. Les contraintes suivantes implémentent cette condition séparément pour la partie fixe et la partie restante, à la fois pour les littéraux présents et pour les variables absentes.

$$\forall t \in \{1, \ldots, K-1\}, \forall k \in \mathcal{C}, \forall x \in \mathcal{X} :$$
$$(\neg lcc_{t,k} \wedge \neg rcc_{t,k} \wedge pccF_{x,k,t}) \implies pccF_{x,k,t+1} \quad (52)$$
$$(\neg lcc_{t,k} \wedge \neg rcc_{t,k} \wedge nccF_{x,k,t}) \implies nccF_{x,k,t+1} \quad (53)$$
$$(\neg lcc_{t,k} \wedge \neg rcc_{t,k} \wedge pccR_{x,k,t}) \implies pccR_{x,k,t+1} \quad (54)$$
$$(\neg lcc_{t,k} \wedge \neg rcc_{t,k} \wedge nccR_{x,k,t}) \implies nccR_{x,k,t+1} \quad (55)$$
$$(\neg lcc_{t,k} \wedge \neg rcc_{t,k} \wedge \neg pccF_{x,k,t} \wedge \neg nccF_{x,k,t}) \implies (\neg pccF_{x,k,t+1} \wedge \neg nccF_{x,k,t+1}) \quad (56)$$
$$(\neg pccR_{x,k,t} \wedge \neg nccR_{x,k,t}) \implies (\neg pccR_{x,k,t+1} \wedge \neg nccR_{x,k,t+1}) \quad (57)$$

Les contraintes (52) à (55) encodent la persistance positive des ensembles de compensation non sélectionnés. Si un ensemble de compensation k n'est utilisé ni comme parent gauche ni comme parent droit à l'étape t, alors chaque littéral présent dans sa partie fixe ou dans sa partie restante est

préservé à l'étape $t+1$. Ainsi, les objets de compensation non utilisés restent inchangés au fil des étapes d'inférence. Les contraintes (56) et (57) encodent la persistance négative correspondante. Si aucune polarité d'une variable x n'apparaît dans la partie fixe, respectivement dans la partie restante, à l'étape t, alors aucune telle occurrence n'est introduite à l'étape $t+1$ uniquement par persistance. Ensemble, ces contraintes garantissent que le contenu d'un ensemble de compensation ne peut changer que lorsque cet ensemble est sélectionné comme parent ; sinon, la présence et l'absence des littéraux sont propagées à l'étape suivante.
De plus, l'encodage impose la dynamique destructive requise par la sémantique de la max-résolution. En particulier, une clause standard est consommée dès qu'elle est utilisée comme parent, comme décrit par les contraintes (58) et (59).

$$\forall t \in \{1,\dots,K-1\}, \forall i \in \mathcal{D}^t :$$
$$(ls_{t,i} \vee rs_{t,i}) \implies \overline{as_{t+1,i}} \tag{58}$$
$$\overline{as_{t,i}} \implies \overline{as_{t+1,i}} \tag{59}$$

Pour les ensembles de compensation, la disponibilité dépend de leur partie restante. Plus précisément, un ensemble de compensation reste disponible tant que sa partie restante est non vide, comme indiqué dans les contraintes (60) et (61).

$$\forall t \in \{1,\dots,K-1\}, \forall k \in \mathcal{C} :$$
$$\left(\bigwedge_{x\in\mathcal{X}} \overline{pccR_{x,k,t}} \wedge \overline{nccR_{x,k,t}}\right) \implies \overline{acc_{t+1,k}} \tag{60}$$
$$\overline{acc_{t,k}} \implies \overline{acc_{t+1,k}} \tag{61}$$

Ainsi, les contraintes (52) à (61) imposent une évolution temporelle cohérente des objets de preuve. Les ensembles de compensation non sélectionnés persistent sans changement. Les clauses standards sélectionnées sont consommées. Les ensembles de compensation restent disponibles uniquement tant qu'ils contiennent encore des informations actives dans leur partie restante.

4.5.1 Condition de réfutation

Rappelons que nous traitons ici des max-réfutations. Nous devons donc garantir que la dernière clause déduite est vide. La contrainte (62) assure cette propriété en vérifiant que la dernière résolvante ne contient aucun littéral.

$$\overline{ps_{x,M+K}} \wedge \overline{ns_{x,M+K}} \qquad \forall x \in \mathcal{X} \tag{62}$$

5 Expérimentations

Nous menons des tests préliminaires afin d'évaluer la validité et l'efficacité de notre cadre pour identifier les plus courtes max-réfutations. La méthode a été implémenté en Python en utilisant la bibliothèque PySAT [15]. Nous avons encodé les contraintes de cardinalité en FNC à l'aide de réseaux de cardinalité [1]. Pour chaque valeur de K, la formule FNC correspondante Φ_K a été générée puis résolue avec le solveur SAT Cadical [3]. La recherche a été effectuée de manière incrémentale : pour une formule d'entrée donnée, nous résolvons $\Phi_1, \Phi_2, \dots$ jusqu'à trouver le premier encodage satisfiable. Cette première valeur satisfiable est notée $K^\star$. Lorsqu'une affectation satisfaisante est retournée, nous extrayons la preuve correspondante et nous la validons avec un vérificateur indépendant. Ce vérificateur vérifie les parents sélectionnés, la cohérence du pivot, la correction de chaque résolvante et la dérivation finale de la clause vide. Toutes les expérimentations ont été réalisées sur une machine équipée d'un processeur Intel Core i5 à 2.4GHz, avec 4 cœurs et 8GB de RAM.

TABLE 1 – Recherche incrémentale de la plus courte réfutation de ϕ dans l'exemple 1. La longueur minimale est $K^\star = 3$. La ligne $K = 4$ est reportée comme une exécution supplémentaire de validation.

Étape (K)	**Statut**	**Variables CNF**	**Clauses CNF**
$K = 1$	UNSAT	93	343
$K = 2$	UNSAT	177	734
$K^\star = 3$	SAT	276	1205
$K = 4$	SAT	390	1756

Nous avons d'abord considéré la formule $\phi = \{(\neg x_1 \vee x_3), (x_1), (\neg x_1 \vee x_2), (\neg x_2 \vee \neg x_3)\}$ de l'exemple 1. Les résultats du solveur au fil des itérations sont résumés dans le tableau 1. Le modèle a correctement identifié $K = 3$ comme la plus courte longueur de réfutation, avec une preuve plus courte que celles montrées dans la figure 1, tirées de [8]. Cela confirme la correction de notre sémantique d'inférence. Nous avons extrait la preuve à partir de l'affectation retournée par le solveur lors de la dernière itération afin d'obtenir la max-réfutation illustrée dans la figure 2.

Nous avons également effectué une exécution supplémentaire de validation avec $K = 4$. Cette exécution ne fait pas partie de la recherche, puisque le minimum est déjà atteint pour $K^* = 3$. Elle sert plutôt à vérifier que l'encodage peut retrouver l'une des dérivations montrées dans la figure 1, qui utilise une inférence sur une clause de compensation. Dans cette exécution, le solveur retourne SAT et produit une dérivation dans laquelle la clause de compensation $(x_1 \vee \neg x_3)$, générée en résolvant (x_1) avec $(\neg x_1 \vee x_3)$, est ensuite sélectionnée comme prémisse pour dériver $(x_2 \vee \neg x_3)$, c'est à dire qu'on obtien une preuve identique à celle illustrée à gauche dans la figure 1. Cette expérience supplémentaire confirme que note encodage permet de capturer des max-réfutations qui utilisent explicitement des inférences basées sur les clauses de compensation.

Ensuite, nous avons considéré des instances issues du principe des tiroirs. Rappelons que ce problème affirme que n pigeons doivent être placés dans $1 \geq h < n$ tiroirs sans qu'au moins deux pigeons partagent le même tiroir. Nous notons $PHP(m,h)$ la formule indiquant que m pigeons doivent être placés dans h tiroirs, chaque pigeon étant affecté à au moins un tiroir, et aucun deux pigeons ne partageant le même tiroir. Cette famille est un benchmark classique en complexité des preuves, et elle est connue pour être difficile pour les systèmes basés sur la résolution, y compris pour la max-résolution [6].

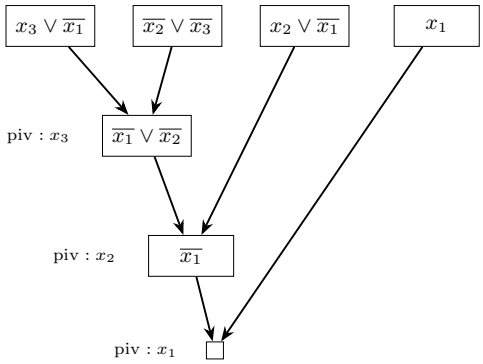

FIGURE 2 – Plus courte max-réfutation pour la formule de l'exemple 1 (les clauses de compensation sont omises, car elles ne sont pas utilisées).

TABLE 2 – Résultats expérimentaux pour l'encodage restreint vérifié de la max-résolution.

Instance	$\|V\|$	$\|C\|$	**Lit.**	$K^\star$	**Temps (s)**	**Valide**
Exemple 1	3	4	7	3	0.0	✓
$PHP(2,1)$	2	3	4	2	0.0	✓
$PHP(3,1)$	3	6	9	2	0.0	✓
$PHP(4,1)$	4	10	16	2	0.1	✓
$PHP(5,1)$	5	15	25	2	0.1	✓
$PHP(3,2)$	6	9	18	10	1800.0	✓
$PHP(4,2)$	8	16	32	10	3635.4	✓

Pour la plus petite instance du principe des tiroirs, $PHP(2,1)$, le cadre identifie correctement une plus courte réfutation de taille $K^\star = 2$. En effet, la formule contient les clauses (x_1), (x_2) et $(\neg x_1 \vee \neg x_2)$; une première étape de résolution dérive une clause unitaire négative, puis une deuxième étape dérive la clause vide. La même longueur optimale est obtenue pour $PHP(3,1)$, car deux pigeons suffisent déjà pour obtenir une contradiction lorsqu'il n'y a qu'un seul tiroir. L'instance $PHP(3,2)$ est beaucoup plus difficile. Le solveur prouve qu'aucune réfutation n'existe pour $K < 10$ dans l'espace de recherche encodé, puis trouve une réfutation correcte pour $K = 10$. À partir des clauses initiales du principe des tiroirs, le certificat extrait dérive la clause vide à travers une séquence de dix étapes de résolution. Pour $PHP(4,2)$, le cadre trouve également une réfutation de longueur $K = 10$. Bien que cette instance contienne plus de variables et de clauses, le certificat extrait utilise seulement un sous-ensemble des clauses initiales, en particulier celles liées au nombre de tiroirs h, correspondant à un cœur insatisfiable déjà présent dans la formule. Cela explique pourquoi $PHP(3,2)$ et $PHP(4,2)$ admettent toutes deux des réfutations de même longueur dans le cadre actuel fondé sur le cœur de résolution.

Les résultats complets sont résumés dans le tableau 2. Pour chaque instance, nous indiquons le nombre de variables $|V|$, le nombre de clauses $|C|$, le nombre total de littéraux, la longueur de la plus courte réfutation $K^\star$, le temps total d'exécution, ainsi que la validation ou non de la preuve extraite par le vérificateur indépendant. Dans l'ensemble, ces expériences fournissent une première preuve de concept pour le cadre proposé. Le solveur réussit à prouver des bornes inférieures pour toutes les valeurs $K < K^\star$, trouve une réfutation pour $K^\star$, et produit des preuves acceptées par un vérificateur indépendant.

6 Conclusion

Dans cet article, nous avons introduit un cadre formel basé sur SAT pour identifier les plus courtes réfutations par max-résolution. Nos résultats préliminaires confirment que le modèle est correct et qu'il est capable de calculer des max-réfutations de taille minimale pour des petites formules. Plusieurs pistes restent à explorer dans les travaux futurs. Tout d'abord, nous prévoyons d'étendre notre évaluation expérimentale en considérant un ensemble plus large d'instances, des encodages de cardinalité plus compacts et des solveurs plus compétitifs. De plus, nous souhaitons généraliser l'encodage en supprimant notre hypothèse restrictive de plus grande clause, en intégrant des règles simples supplémentaires connues pour améliorer l'efficacité de la max-résolution [19], et en étendant notre cadre au calcul des plus courtes preuves d'optimalité pour les instances MaxSAT.

Remerciements

Ce travail est partiellement soutenu par la chaire SAIPS, financée par l'université de Picardie Jules Verne, et par le projet ANR-24-CE23-6126 (BforSAT), financé par l'Agence Nationale de la Recherche.

Références

[1] Roberto Asín, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. Cardinality networks and their applications. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 167–180. Springer, 2009.

[2] Jeremias Berg, Bart Bogaerts, Jakob Nordström, Andy Oertel, Tobias Paxian, and Dieter Vandesande. Certifying without loss of generality reasoning in solution-improving maximum satisfiability. In Paul Shaw, editor, *30th International Conference on Principles and Practice of Constraint Programming, CP 2024, Girona, Spain, September 2-6, 2024*, volume 307 of *LIPIcs*, pages 4 :1–4 :28. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[3] Armin Biere, Tobias Faller, Katalin Fazekas, Mathias Fleury, Nils Froleyks, and Florian Pollitt. Cadical 2.0. In *International Conference on Computer Aided Verification*, pages 133–152. Springer, 2024.

[4] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2021.

[5] Maria Luisa Bonet and Jordi Levy. Equivalence between systems stronger than resolution. In Luca Pulina and Martina Seidl, editors, *Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference, Alghero, Italy, July 3-10, 2020, Proceedings*, volume 12178 of *LNCS*, pages 166–181. Springer, 2020.

[6] Maria Luisa Bonet, Jordi Levy, and Felip Manyà. Resolution for max-sat. *Artificial Intelligence*, 171(8-9) :606–618, 2007.

[7] Mohamed Sami Cherif, Djamal Habet, and André Abramé. Understanding the power of max-sat resolution through up-resilience. *Artif. Intell.*, 289 :103397, 2020.

[8] Mohamed Sami Cherif, Djamal Habet, and Matthieu Py. From crossing-free resolution to max-sat resolution. In Christine Solnon, editor, *28th International Conference on Principles and Practice of Constraint Programming, CP 2022, Haifa, Israel, July 31 - August 8, 2022*, volume 235 of *LIPIcs*, pages 12 :1–12 :17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[9] Sami Cherif, Heythem Sattoutah, Chu-Min Li, Corinne Lucet, and Laure Brisoux-Devendeville. Minimizing Working-Group Conflicts in Conference Session Scheduling Through Maximum Satisfiability. In Paul Shaw, editor, *30th International Conference on Principles and Practice of Constraint Programming (CP 2024)*, volume 307 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34 :1–34 :11, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[10] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, page 151–158, New York, NY, USA, 1971. Association for Computing Machinery.

[11] Ana Graça, Inês Lynce, Joao Marques-Silva, and Arlindo L Oliveira. Efficient and accurate haplotype inference by combining parsimony and pedigree information. In *Algebraic and Numeric Biology : 4th International Conference, ANB 2010, Hagenberg, Austria, July 31-August 2, 2010, Revised Selected Papers*, pages 38–56. Springer, 2012.

[12] João Guerra and Inês Lynce. Reasoning over biological networks using maximum satisfiability. In *International conference on principles and practice of constraint programming*, pages 941–956. Springer, 2012.

[13] Aarti Gupta, Malay K Ganai, and Chao Wang. Sat-based verification methods and applications in hardware verification. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems*, pages 108–143. Springer, 2006.

[14] Federico Heras and João Marques-Silva. Read-once resolution for unsatisfiability-based max-sat algorithms. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 572–577. IJCAI/AAAI, 2011.

[15] Alexey Ignatiev, António Morgado, and João Marques-Silva. Pysat : A python toolkit for prototyping with SAT oracles. In Olaf Beyersdorff and Christoph M. Wintersteiger, editors, *Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, Oxford, UK, July 9-12, 2018, Proceedings*, volume 10929 of *LNCS*, pages 428–437. Springer, 2018.

[16] Javier Larrosa and Federico Heras. Resolution in max-sat and its relation to local consistency in weighted csps. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, page 193–198, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[17] Javier Larrosa, Federico Heras, and Simon De Givry. A logical approach to efficient max-sat solving. *Artificial Intelligence*, 172(2-3) :204–233, 2008.

[18] Javier Larrosa and Emma Rollon. Augmenting the power of (partial) maxsat resolution with extension. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 1561–1568. AAAI Press, 2020.

[19] Javier Larrosa and Emma Rollon. *Towards a Better Understanding of (Partial Weighted) MaxSAT Proof Systems*, volume 12178 of *LNCS*, pages 218–232. Springer, 2020.

[20] Chu Min Li, Felip Manyà, Nouredine Ould Mohamedou, and Jordi Planes. Resolution-based lower bounds in max-sat. *Constraints An Int. J.*, 15(4) :456–484, 2010.

[21] Chu Min Li, Felip Manyà, and Jordi Planes. New inference rules for max-sat. *J. Artif. Intell. Res.*, 30 :321–359, 2007.

[22] Shoulin Li, Jordi Coll, Djamal Habet, Chu-Min Li, and Felip Manyà. *A Tableau Calculus for MaxSAT Based on Resolution*, volume 356 of *Frontiers in Artificial Intelligence and Applications*, pages 35–44. IOS Press, 2022.

[23] Carlos Mencía and Joao Marques-Silva. Computing shortest resolution proofs. In *EPIA Conference on Artificial Intelligence*, pages 539–551. Springer, 2019.

[24] Nina Narodytska and Fahiem Bacchus. Maximum satisfiability using core-guided maxsat resolution. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 2717–2723. AAAI Press, 2014.

[25] Matthieu Py, Mohamed Sami Cherif, and Djamal Habet. Towards bridging the gap between SAT and max-sat refutations. In *32nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2020, Baltimore, MD, USA, November 9-11, 2020*, pages 137–144. IEEE, 2020.

[26] Matthieu Py, Mohamed Sami Cherif, and Djamal Habet. A proof builder for max-sat. In Chu-Min Li and Felip Manyà, editors, *Theory and Applications of Satisfiability Testing - SAT 2021 - 24th International Conference, Barcelona, Spain, July 5-9, 2021, Proceedings*, volume 12831 of *LNCS*, pages 488–498. Springer, 2021.

[27] Matthieu Py, Mohamed Sami Cherif, and Djamal Habet. Proofs and certificates for max-sat. *J. Artif. Intell. Res.*, 75 :1373–1400, 2022.

[28] J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1) :23–41, January 1965.

[29] Sean Safarpour, Hratch Mangassarian, Andreas Veneris, Mark H Liffiton, and Karem A Sakallah. Improved design debugging using maximum satisfiability. In *Formal Methods in Computer Aided Design (FMCAD'07)*, pages 13–19. IEEE, 2007.

[30] Irene Unceta, Bernat Salbanya, Jordi Coll, Mateu Villaret, and Jordi Nin. Optimizing resource allocation in home care services using maxsat. *Cogn. Syst. Res.*, 88 :101291, 2024.

[31] Dieter Vandesande, Jordi Coll, and Bart Bogaerts. Certified branch-and-bound maxsat solving (extended version). *CoRR*, abs/2511.10273, 2025.

[32] Dieter Vandesande, Wolf De Wulf, and Bart Bogaerts. Qmaxsatpb : A certified maxsat solver. In Georg Gottlob, Daniela Inclezan, and Marco Maratea, editors, *Logic Programming and Nonmonotonic Reasoning - 16th International Conference, LPNMR 2022, Genova, Italy, September 5-9, 2022, Proceedings*, volume 13416 of *LNCS*, pages 429–442. Springer, 2022.