La résolution étendue étroite

Nicolas Prcovic

La Résolution étendue étroite

Nicolas Prcovic
LSIS - Université d'Aix-Marseille
nicolas.prcovic@lsis.org

Résumé

La résolution étendue (ER) (c'est-à-dire, la résolution incorporant la règle d'extension) est un système de preuve plus puissant que la résolution standard (Res) car elle permet de résoudre en temps polynômial certaines classes d'instances que Res ne peut traiter qu'en temps exponentiel. Cependant, elle est très difficile à mettre en pratique car la règle d'extension accroit considérablement la taille de l'espace de recherche de la preuve. On dit qu'une résolution est étroite si l'application de la règle de résolution produit une résolvante contenant au maximum trois littéraux. Dans cet article nous présentons deux variantes de ER : la résolution étendue étroite (ER3) et la résolution à refragmentation. Ces deux systèmes de preuves p-simulent ER : toute preuve de l'une n'est que polynômialement plus longue que celle de l'autre. ER3 consiste simplement à n'autoriser que des résolutions étroites dans ER. Ceci permet une diminution exponentielle de l'espace de recherche d'une preuve dans ER. La résolution à refragmentation est une variante de ER3 qui permet une intégration facile de la résolution étendue dans n'importe quel solveur appliquant la résolution.

Abstract

Extended Resolution (ie, Resolution incorporating the extension rule) is a more powerful proof system than Resolution because it can find polynômially bounded refutations of some SAT instances where Resolution alone cannot (and at the same time, every proof with resolution is still a valid proof with extended resolution). However it is very difficult to put it into practice because the extension rule is an additionnal source of combinatorial explosion, which tends to lengthen the time to discover a refutation. We call a restriction of Resolution forbiding the production of resolvents of size greater than 3 *Narrow Resolution*. We show that Narrow Extended Resolution p-simulates (unrestricted) Extended Resolution. We thus obtain a proof system whose combinatorics is highly reduced but which is still as powerful as before. We also define *Splitting Resolution*, a way to integrate easily Narrow Extended Resolution into any resolution-based solver

1 Introduction

Les systèmes de preuve basés sur la résolution sont utilisés pour permettre de trouver la réfutation d'une formule booléenne supposée insatisfiable. Cependant, certaines familles d'instances [7, 16, 4] nécessitent de produire un nombre exponentiel de résolvantes. La résolution étendue [15] ajoute une régle supplémentaire à la résolution, consistant à ajouter les trois clauses correspondant à $x \Leftrightarrow l_1 \vee l_2$ à la formule, où x est une nouvelle variable. Dans ce cadre, Cook [5] exhibe une réfutation de longueur polynômiale du problème des pigeons. Personne n'a jamais trouvé de classe d'instances nécessitant la production d'un nombre exponentiel de résolvantes avec la résolution étendue. Mais, bien que rendant un système de preuve par résolution plus puissant, la règle d'extension introduit une source supplémentaire d'explosion combinatoire. En effet, le nombre potentiel de résolvantes que l'on peut produire n'est plus une exponentielle du nombre de variables de la formule mais du nombre total de variables, qui inclut les variables ajoutées par les extensions. Or, le nombre d'extensions que l'on peut faire n'est pas nécessairement polynômial. C'est pourquoi les rares systèmes l'incluant (GUNSAT [2], GlucosEr [1], ECL [8]) restreignent beaucoup l'application de cette règle afin que son avantage, le possible raccourcissement de la longueur de la réfutation, ne soit pas contrebalancé par son inconvénient, le traitement des clauses supplémentaires.

Rappelons qu'on dit qu'un système de preuve P *p-simule* un système de preuve Q si toute réfutation produite par P n'est que polynômialement plus longue que celle produite par Q. Dans cet article, nous proposons deux variantes de la Résolution étendue, qui la p-simulent et sont destinés à améliorer son efficacité pratique. La première variante, appelée Résolution étendue étroite, se contente de limiter l'application de la règle de résolution à la production de clauses de

taille 3 au maximum. La combinatoire de ce système de preuve se trouve dont très fortement réduit. La seconde variante, appelée Résolution à refragmentation, remplace la règle d'extension de la Résolution étendue étroite par une règle qui combine une résolution et une extension pour produire deux nouvelles clauses de taille 3 à partir de deux clauses de taille 3. Elle permet ainsi de s'intégrer facilement à n'importe quel solveur en substituant la nouvelle règle à la règle de résolution.

Après avoir rappelé quelques notions sur le problème SAT, donné les rapports entre longueur et largeur de preuve et introduit la Résolution étendue (section 2), nous démontrons que la Résolution étendue étroite p-simule la Résolution étendue (section 3) puis que c'est aussi le cas pour la Résolution à refragmentation (section 4).

2 Notions préliminaires

Une *formule booléenne*, sous sa forme dite *CNF*, est une conjonction de clauses. Une clause est une disjonction de littéraux. Un littéral est soit une variable booléenne, soit sa négation. On définit var(l) comme étant la variable du littéral l. Les variables peuvent être affectées à la valeur vrai ou faux. Une clause peut être représentée par l'ensemble de ses littéraux. Une formule SAT peut être représentée par l'ensemble de ses clauses. Un *modèle* d'une formule ϕ est une affectation de variables qui est telle que ϕ est vraie. Une formule n'ayant pas de modèle est dite *insatisfiable*. La clause vide, notée $\square$, est toujours fausse et si une formule la contient alors elle est insatisfiable.

2.1 La résolution

Les systèmes formels de preuve propositionnelle permettent de dériver d'autres formules à partir d'une formule ϕ, grâce à des règles d'inférence. En particulier, le système de Robinson [14], que nous appellerons `Res`, permet de dériver des clauses induites à partir des clauses d'une formule ϕ grâce à la règle de *résolution* :

$$\frac{C_1 \qquad C_2}{C_1 \backslash \{l\} \cup C_2 \backslash \{\bar{l}\}}$$

La clause inférée par résolution de C_1 avec C_2 sera appelé *résolvante* sur var(l) de C_1 et C_2. L'intérêt de `Res` est qu'il permet d'établir des réfutations de formules (ie, des preuves de leur insatisfiabilité) dans la mesure où une formule est insatisfiable si et seulement si il existe une suite de résolutions qui dérive la clause vide. `Res` fonctionne par saturation de l'application de sa règle : si une formule est satisfiable, `Res` le détecte quand plus aucune résolvante non redondante ne peut être dérivée.

Une dérivation est une suite d'applications de la règle de résolution permettant de dériver une clause. Elle constitue la preuve que cette clause est une conséquence logique de la formule. Elle peut se représenter grâce à un arbre binaire dont les feuilles sont des clauses de ϕ, les nœuds internes sont des résolvantes et la racine est la clause dérivée. Une dérivation de la clause vide s'appelle une *réfutation*.

2.2 Longueur et largeur de preuve

On appelle *longueur d'une preuve*, le nombre de résolvantes qu'elle contient. On appelle *taille* d'une clause le nombre de littéraux qu'elle contient. On appelle *largeur d'une preuve* la taille de la plus grande clause apparaissant dans cette preuve. Nous dirons qu'une résolution est *étroite* quand la résolvante qu'elle produit est de taille 3 au maximum.

Les relations entre longueur et largeur de preuve sont fortes. Il est clair qu'une preuve étroite est forcément courte. Plus précisément, si la largeur de la preuve est k alors la preuve ne peut contenir que de l'ordre de $\Theta(n^k)$ résolvantes différentes, où n est le nombre de variables de la formule. Si k est une constante alors la longueur de la preuve est polynômialement bornée. Dans [3], les auteurs montrent complémentairement que lorsqu'on peut produire une preuve courte, on peut faire en sorte qu'elle soit étroite. A partir de cette relation entre largeur et longueur de preuve, il est possible de définir des algorithmes de recherche de réfutations qui restreignent la largeur des résolvantes produites. Par exemple, le pré-traitement de Satz[12] (avant une résolution complète à la DPLL) consiste à effectuer toutes les résolutions étroites possibles jusqu'à saturation (ou dérivation éventuelle de la clause vide). Plus généralement, une manière de garantir une complexité polynômiale consiste à restreindre la résolution à la production de résolvantes de taille inférieure à une borne donnée. Evidemment, cela se fait au détriment de la complétude dans la mesure où il est parfois nécessaire de dériver des grandes clauses avant d'obtenir la clause vide. On peut aussi définir une méthode complète qui restreint la résolution à la dérivation de résolvantes de taille k maximum mais qui incrémente la valeur de k à chaque fois qu'il y a saturation [3]. Dans cet article, nous explorons une autre voie. Au lieu d'augmenter la valeur de k, que nous fixons une fois pour toute à 3, nous nous plaçons dans la cadre de la résolution étendue, qui va nous permettre de rajouter des clauses permettant de n'effectuer que des résolutions étroites.

2.3 La résolution étendue

La résolution étendue [15] consiste à ajouter à `Res` la *règle d'extension* en plus de la règle de résolution. Nous appellerons `ER` ce nouveau système de preuve. En toute généralité, la règle d'extension permet d'ajouter (à l'ensemble des clauses) les clauses correspondant à la formule $x \Leftrightarrow F$, où x est une nouvelle variable et F est n'importe quelle formule portant sur des variables existant déjà. A partir de maintenant, nous distinguerons les nouvelles variables introduites par des extensions, dites *variables d'extension*, des variables de la formule initiale, dites *variables d'entrée*.

On peut toujours se restreindre à ce que F soit uniquement du type $l_1 \vee l_2$ où l_1 et l_2 sont des littéraux portant sur des variables déjà présentes. En effet, il est facile de décomposer récursivement n'importe quelle formule F en une composition de disjonctions de sous-formules, éventuellement négatives. En associant chaque sous-formule à une nouvelle variable, on obtient un ensemble d'extensions de type $x_i \Leftrightarrow (l_1 \vee l_2)$ au lieu d'une seule du type $x \Leftrightarrow F$. Ce type d'extension portant sur la disjonction de deux littéraux préexistants sera appelée *extension binaire disjonctive*. Une extension $x \Leftrightarrow (l_1 \vee l_2)$ consiste à ajouter les clauses $\overline{x} \vee l_1 \vee l_2$, $x \vee \overline{l_1}$ et $x \vee \overline{l_2}$. Il faut noter qu'il est aussi parfaitement possible de se restreindre à des extensions binaires conjonctives (i.e., de type $x \Leftrightarrow (l_1 \wedge l_2)$, qui consiste à ajouter les clauses $x \vee \overline{l_1} \vee \overline{l_2}$, $\overline{x} \vee l_1$ et $\overline{x} \vee l_2$).

L'intérêt de rajouter la règle d'extension est qu'elle rend le système plus puissant dans la mesure où certains problèmes dont la preuve est de longueur nécessairement exponentielle dans `Res` ont une preuve polynômiale dans `ER`. C'est le cas du fameux problème des pigeons [7, 5]. En dehors de ce problème particulier, on peut identifier des cas plus généraux où une extension permet un raccourcissement de preuve. Considérons une preuve contenant des séquences de résolutions qui font que $C_1 \vee C$ est utilisée pour dériver $C_1 \vee C'$ et que $C_2 \vee C$ est utilisée pour dériver $C_2 \vee C'$ grâce à la même suite de clauses qui permet de passer de C à C' (C_1, C_2, C et C' sont des clauses). On peut vérifier que grâce aux clauses émanant de $x \Leftrightarrow C_1 \wedge C_2$, on dérive d'abord $x \vee C$ à partir de $C_1 \vee C$ et $C_2 \vee C$, puis on effectue *une seule fois* la séquence de résolutions sur C permettant de dériver $x \vee C'$, à partir de quoi on peut à nouveau dériver $C_1 \vee C'$ et $C_2 \vee C$ (encore grâce aux clauses émanant de la même extension). Cette idée, appelée *compression de preuve* dans [1] est mise en œuvre lors de la phase d'apprentissage d'une clause dans un solveur CDCL. Par contre, dans [8], quand une clause $\alpha \vee \beta$, où α et β sont des sous-clauses de longueur ≥ 2, est apprise par un solveur CDCL, l'extension $x \Leftrightarrow \alpha$ est ajoutée. Dans les expérimentations de cet article, α ne contient que deux littéraux, ce qui revient donc à effectuer une extension binaire disjonctive. C'est aussi ce type d'extension que nous utiliserons ici.

3 La Résolution étendue étroite

Bien que `ER` soit théoriquement plus puissant que `Res`, i.e. permet de trouver des preuves plus courtes, en pratique elle pose des difficultés qui l'ont empêché jusqu'à récemment d'être performante. Le choix des extensions à effectuer est une question difficile. Personne n'y a trouvé de réponse satisfaisante depuis plus de quarante ans. Et quand bien même les bons choix seraient faits, les résultats ne seraient pas automatiquement meilleurs. Par exemple, quand nous avons ajouté les $\Theta(n^3)$ clauses issues de la règle d'extension proposés par Cook aux $\Theta(n^2)$ clauses du problème des n pigeons, nous avons expérimenté qu'aucun type de solveurs (qu'il soit de type CDCL ou DPLL, que ce soit DP60 [6] ou de la SL-résolution [10]) ne résolvait plus rapidement (bien au contraire) le problème alors que son temps de résolution devenait théoriquement polynômial. A moins d'avoir un bon moyen de choisir les résolvantes à produire, il y a le risque de produire à peu près les mêmes résolvantes qu'auparavant et beaucoup d'autres encore.

Avec `Res`, la taille maximale d'une résolvante est n, où n est le nombre de variables de la formule. Le nombre de résolvantes possible est en $\Theta(3^n)$ car, quelle que soit la variable, elle peut apparaître dans une clause positivement, négativement ou ne pas apparaître. Dans `ER`, le nombre de résolvantes possible est en $\Theta(3^{(n+n')})$ où n' est le nombre d'extensions. Il faut remarquer que n' peut être lui-même une exponentielle de n, étant donné que chaque extension correspond à l'une des formules possibles que l'on peut construire à partir de n variables. L'espace de recherche de la preuve dans `ER` est donc exponentiellement plus grand que dans `Res`. C'est pourquoi les algorithmes voulant intégrer la règle d'extension pour essayer de produire des réfutations plus courtes ont cherché jusqu'à présent à limiter fortement le nombre d'extensions possibles. Or, dans [13], nous avions pris le parti inverse : autoriser a priori toutes les extensions possibles mais limiter l'application de la règle de résolution à des résolutions étroites (ie, la taille des résolvantes ne doit pas dépasser 3). Nous obtenions donc un nouveau système de preuve, appelé *Résolution étendue étroite* (`ER3`), qui était démontré complet et tel que toute preuve arborescente dans `ER` pouvait se réduire en temps polynômial en preuve dans `ER3` dans [13]. La question de savoir si `ER3` p-simule `ER` était laissée ouverte. La démonstration de la réponse positive (et simple, du moment que nous avons maintenant pris connaissance

d'un résultat qui nous avait échappé) à cette question est donnée plus bas dans le présent article.

L'intérêt pratique de ER3, à partir du moment où toute résolvante possède une taille limitée à 3, est que l'espace de recherche d'une preuve se trouve réduite à $\theta((n + n')^3)$. Il s'agit au pire d'une exponentielle de n, comme pour Res. Nous nous retrouvons donc à chercher des preuves potentiellement plus courte que dans Res dans un espace de recherche dont l'ordre de grandeur est à peu près du même ordre que celui de Res.

La Résolution étendue étroite ne peut s'appliquer que si les formules CNF que nous cherchons à réfuter sont des instances 3-SAT (i.e., ne contiennent que des clauses de taille 3). Dans le cas contraire, il est toujours possible de ramener une formule CNF à une instance 3-SAT équivalente du point de vue de la satisfiabilité. La technique standard consiste à remplacer une clause $C = a_1 \vee a_2 \vee ... \vee a_k$ par $k-2$ clauses ternaires $a_1 \vee a_2 \vee \overline{x_2}$, $x_2 \vee a_3 \vee \overline{x_3}$, ..., $x_{k-3} \vee a_{k-2} \vee \overline{x_{k-2}}$ et $x_{k-2} \vee a_{k-1} \vee a_k$, où les $k-3$ variables x_i sont nouvelles. Cela se fait simplement en ajoutant les extensions $x_2 \Leftrightarrow (a_1 \vee a_2)$ et $\forall i, 3 \leq i < k-1, x_i \Leftrightarrow (a_{i-1} \vee x_{i-1})$. En effet, il suffit de remarquer que les clauses ternaires ainsi ajoutées sont celles du type $x_i \vee a_{i+1} \vee \overline{x_{i+1}}$ qui remplacent C, tandis qu'en appliquant linéairement la suite de $2k-4$ résolutions entre les clauses binaires ajoutées et C, on dérive la dernière clause de remplacement $x_{k-2} \vee a_{k-1} \vee a_k$.

On peut obtenir une preuve arborescente de largeur 3 à partir d'une preuve arborescente de largeur quelconque par une suite d'*amincissements*. Un amincissement de preuve consiste à réduire d'au moins une unité le nombre de ses résolvantes de taille 4. Un amincissement se fait grâce à une extension $x \Leftrightarrow l_1 \vee l_2$ qui permet à une résolvante $l_1 \vee l_2 \vee l_3 \vee l_4$ de se réduire à $x \vee l_3 \vee l_4$. x est remplacé plus tard par $l_1 \vee l_2$ lorsque la clause qui le contient est binaire. Un exemple d'amincissements menant à une preuve étroite se trouve en figure 1.

Théorème 1 ER3 *p-simule* ER.

Preuve 1 *La proposition 4 de [13] montre qu'il existe une fonction de réduction d'une preuve arborescente dans* ER *en une preuve arborescente dans* ER3 *dont la complexité est polynômiale. Comme* ER *est un exemple de système de Frege et qu'il est montré dans [11] que les preuves des systèmes de Frege sont réductibles en temps polynômial en des preuves arborescentes, on en conclut directement que* ER3 *p-simule* ER. □

On trouvera par exemple dans [9] une explication de la façon dont une preuve dans ER peut être transformée en preuve arborescente en temps polynômial.

Conséquences théoriques et pratiques

Chercher une réfutation dans Res, c'est chercher une séquence de résolutions qui aboutit à la clause vide. Une conséquence intéressante de notre résultat est qu'on peut maintenant envisager cette recherche de réfutation tout autrement : chercher une réfutation dans ER3, c'est chercher l'ensemble des extensions telles qu'une suite de résolutions étroites (en nombre cubique) aboutit à la clause vide. Ainsi, l'ensemble des extensions d'une réfutation constitue un certificat d'insatisfiabilité dont la complexité de vérification est une fonction cubique du nombre d'extensions et de clauses d'entrée.

Si P $\neq$ co-NP, certaines instances nécessitent un temps superpolynômial pour que leur réfutation soit découverte par ER3, ce qui implique que le nombre d'extensions nécessaires est aussi superpolynômial. Comme une extension peut être vue comme un lemme permettant de renommer des morceaux de preuve identiques afin de réduire la taille de la preuve, cela signifie qu'il existerait des instances dont les réfutations ne pourraient être raccourcies que par un nombre exponentiel de lemmes. Les lemmes n'y auraient donc qu'un effet marginal voire nul. La structure des preuves serait trop irrégulière pour être suffisamment "factorisable".

D'un point de vue pratique, on peut imaginer un algorithme de recherche de réfutation basé sur ER3 et n'autorisant qu'un nombre polynômial d'extensions. Il ne serait pas complet mais il ne serait incapable de décider que des instances de toute façon trop longue à traiter par ER3. Il chercherait donc une suite de longueur polynômiale d'extensions dont il vérifierait qu'elle complète la formule initiale de telle manière qu'une suite de résolutions étroites (en nombre cubique) permet d'inférer la clause vide. Cet algorithme pourrait être systématique (rechercher toutes les suites d'extensions possibles) ou procéder par réparation (proposer une suite d'extensions initiale et la modifier de proche en proche).

Cette approche de la recherche de réfutation diffère complètement des approches traditionnelles basées sur la résolution. La résolution étendue a toujours été considérée comme principalement de la résolution avec secondairement des extensions. Or, ER3 se voit naturellement comme des extensions complétées par de la résolution étroite. De ce fait, pour définir des algorithmes basés sur ER3, nous ne pouvons pas envisager de nous appuyer sur des algorithmes efficaces existants. C'est pourquoi il nous est apparu utile de chercher une variante de ER3 qui puisse à nouveau se voir comme de la "résolution avec des extensions".

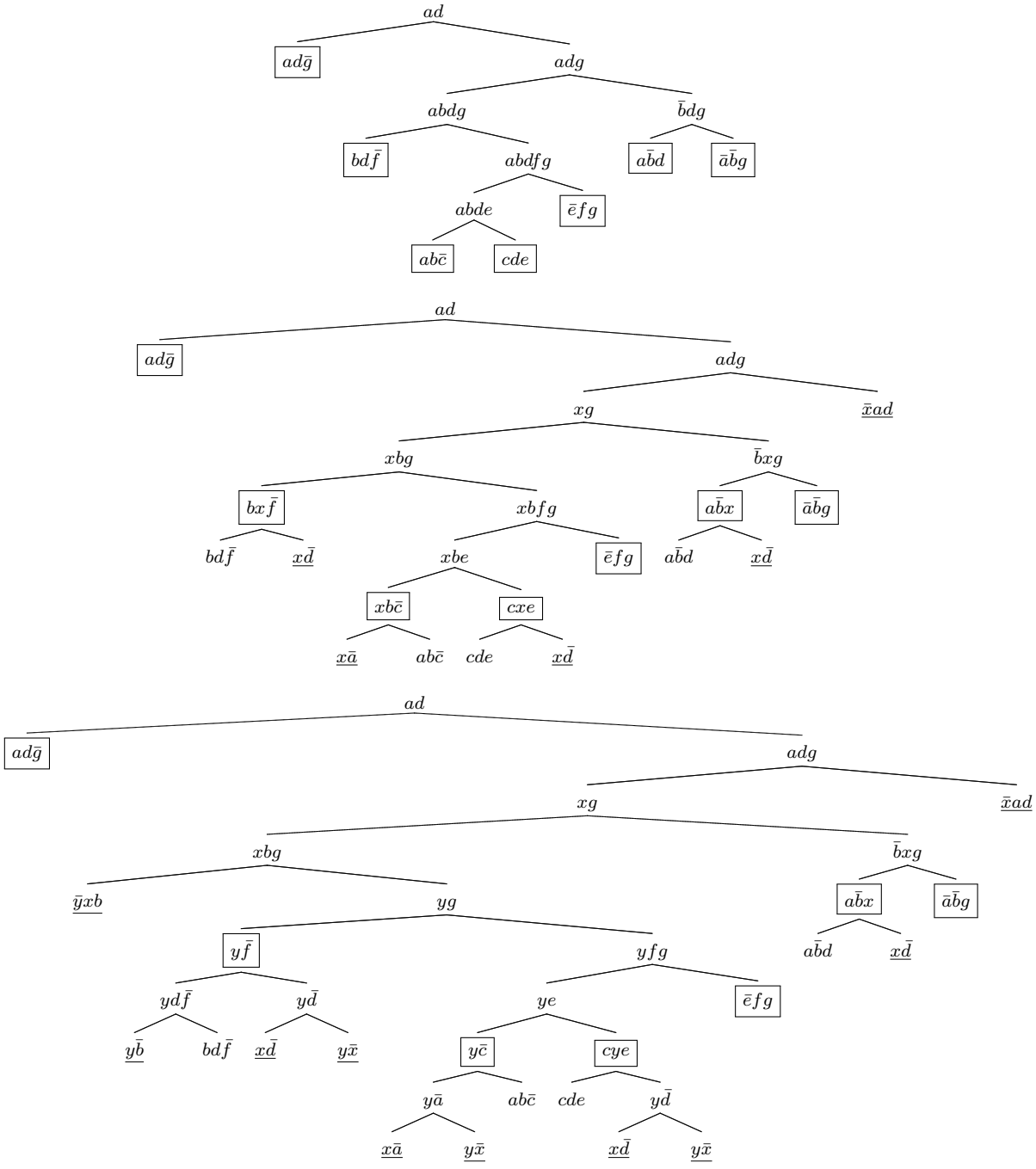

FIGURE 1 – Deux amincissements consécutifs menant à une preuve étroite. Les deux extensions sont $x \Leftrightarrow (a \vee d)$, raccourcissant $abdg$, et $y \Leftrightarrow (x \vee b)$, raccourcissant $xbfg$. Les clauses des extensions sont surlignées. Les clauses encadrées figurent les clauses d'entrée ou ce qu'elles sont devenues suite à l'intégration des extensions.

4 La Résolution à refragmentation

Voici une variante de ER3, que nous appellerons *Résolution à refragmentation*, dont la définition des règles permet une intégration facile de la résolution étendue dans n'importe quel solveur utilisant le mécanisme de la résolution. Son intérêt est de se présenter comme une autre façon de faire de la résolution. Ses deux règles sont :

- la règle de résolution étroite :
$$\frac{C_1 \qquad C_2}{C_1 \backslash \{l\} \cup C_2 \backslash \{\bar{l}\}}$$
si $|C_1 \backslash \{l\} \cup C_2 \backslash \{\bar{l}\}| \leq 3$.
- la règle de refragmentation :
$$\frac{l \vee l_1 \vee l_2 \qquad \bar{l} \vee l_3 \vee l_4}{x \vee l_1 \vee l_3,\ \bar{x} \vee l_2 \vee l_4,\ x \vee \bar{l_2},\ x \vee \bar{l_4}}$$
si l_1, l_2, l_3 et l_4 sont différents, où x est une nouvelle variable.

La règle de refragmentation permet en quelque sorte de scinder une résolvante $l_1 \vee l_2 \vee l_3 \vee l_4$ en deux clauses ternaires $x \vee l_1 \vee l_3$ et $\bar{x} \vee l_2 \vee l_4$, grâce aux clauses de l'extension $x \Leftrightarrow l_2 \vee l_4$. Elle correspond simplement à l'ajout des trois clauses de l'extension $\bar{x} \vee l_2 \vee l_4$, $x \vee \bar{l_2}$, $x \vee \bar{l_4}$ puis à une suite de trois résolutions étroites pour obtenir $x \vee l_1 \vee l_3$. C'est pourquoi nous pourrions aussi l'exprimer ainsi :

$$\frac{l \vee l_1 \vee l_2 \qquad \bar{l} \vee l_3 \vee l_4}{x \vee l_1 \vee l_3,\ x \Leftrightarrow l_2 \vee l_4}$$

Ce système de preuve n'est censé s'appliquer qu'à des instances 3-SAT. Il permet de ne générer que des clauses courtes.

Définition 1 *3-saturation d'une formule*
On appelle 3-saturation *d'une formule 3-SAT la complétion de cette formule avec toutes les résolvantes étroites possibles par application itérative de la règle de résolution étroite. Une formule est dite* 3-saturée *si la règle de résolution étroite ne peut pas (ou plus) s'appliquer.*

Comme nous l'avons vu, si n est le nombre de variables d'une formule, le nombre maximum de résolvantes est en $\Theta(n^3)$ donc la 3-saturation d'une formule 3-SAT se fait en temps polynômial.

Théorème 2 *La Résolution à refragmentation p-simule la Résolution étendue.*

Preuve 2 *Il suffit de montrer que la Résolution à refragmentation p-simule* ER3*. Or, comme les deux systèmes de preuve ne différent que par une seule règle, la règle de refragmentation remplaçant la règle d'extension, il suffit de démontrer qu'une suite polynômialement bornée d'applications de la règle de résolution étroite et de la règle de refragmentation permet d'obtenir n'importe quelle extension utile à la recherche d'une réfutation.*
Définissions le graphe de résolution (X, V) où l'ensemble des sommets X sont les clauses issus d'une instance (clauses d'entrée, résolvantes et clauses d'extension) et l'ensemble des arcs V est tel que $(C_1, C_2) \in V$ ssi on peut appliquer la règle de résolution entre C_1 et C_2. Effectuons (en temps polynômial) une 3-saturation de la formule. Dès lors, seule la règle de refragmentation peut s'appliquer. Considérons dans un premier temps le cas où le graphe de résolution est connexe. Pour chaque couple de littéraux (l_1, l_2) tels qu'ils ne portent pas sur la même variable, deux cas sont possibles : soit il existe une clause qui les contient tous les deux, soit ils n'apparaissent que séparément dans des clauses différentes. Dans le premier cas, nous considérons le cycle (qui est une chaîne revenant sur elle-même) le plus court partant et revenant à la clause. Dans le deuxième cas, considérons la chaîne la plus courte permettant de relier une clause ternaire contenant l_1 à une autre clause ternaire contenant l_2. Nous appelons distance entre l_1 et l_2 la longueur de cette chaîne (ou de ce cycle). Nous procédons par récurrence sur la longueur de la chaîne. Si la longueur est 1, les deux clauses se résolvent grâce à la règle de refragmentation et on obtient les clauses de l'extension $x \Leftrightarrow l_1 \vee l_2$. Si la longueur k est supérieur à 1, considérons les trois premiers éléments de la chaîne : $l_1 \vee a \vee b$, $\bar{b} \vee c \vee d$, $\bar{d} \vee e \vee f$. L'application de la règle de refragmentation aux deux premières clauses permet notamment d'obtenir $x \vee l_1 \vee d$. La chaîne commençant par $x \vee l_1 \vee d$ et $\bar{d} \vee e \vee f$ et suivie par les clauses restantes de la chaîne précédente est de longueur $k-1$ et permet de relier l_1 à l_2. Il suffit donc de répéter l'opération $k-1$ fois pour obtenir les clauses de $x \Leftrightarrow l_1 \vee l_2$. Vérifions maintenant que k est polynômial en n le nombre de variables de l'instance. Il faut prendre en compte le fait que des extensions ont pu compléter le graphe de résolution auparavant. La distance maximale d_1 entre deux littéraux de la formule initiale est inférieur au nombre de clauses qui est en $O(n^3)$. La distance d entre deux littéraux d'extension est inférieure à $d_1 + 2d_2$, où d_2 est la distance maximale entre un littéral d'extension et un littéral de la formule initiale. Il nous reste donc à montrer que d_2 est polynômial en n. Une extension $z \Leftrightarrow x \vee y$ ajoute trois clauses qui complètent le graphe de résolution. Etant donnée l'extension $z \Leftrightarrow x \vee y$, la distance $d(z, l)$ entre z et un littéral l de l'instance initiale est bornée par $1 + min(d(x, l), d(y, l))$. $d(z, l)$ est donc borné par le logarithme du nombre d'extensions (qui peut être une exponentielle de n) donc $d(z, l)$ est borné par un polynôme en n.
Dans le cas où le graphe de résolution n'est pas

connexe, chaque composante connexe représente une sous-formule indépendante. L'ajout d'une extension portant sur des littéraux appartenant à deux sous-formules indépendantes est inutile car la preuve la plus courte se fait en utilisant les clauses d'une seule des sous-formules. □

L'intérêt de la Résolution à refragmentation est qu'elle est facile à intégrer à un solveur. A la place d'une résolution standard, le solveur applique l'une des deux règles selon la taille attendue de la résolvante produite. Il faut cependant noter que l'application de la règle de refragmentation n'est pas déterministe une fois les deux clauses sélectionnées contrairement à la règle de résolution. En effet, à partir des clauses $l \vee l_1 \vee l_2$ et $\bar{l} \vee l_3 \vee l_4$, la règle de refragmentation peut s'appliquer de 4 façons différentes et produire au choix les clauses suivantes :

- $x \vee l_1 \vee l_3$, $\bar{x} \vee l_2 \vee l_4$, $x \vee \bar{l_2}$, $x \vee \bar{l_4}$ (extension $x \Leftrightarrow l_2 \vee l_4$)
- $x \vee l_1 \vee l_4$, $\bar{x} \vee l_2 \vee l_3$, $x \vee \bar{l_2}$, $x \vee \bar{l_3}$ (extension $x \Leftrightarrow l_2 \vee l_3$)
- $x \vee l_2 \vee l_3$, $\bar{x} \vee l_1 \vee l_4$, $x \vee \bar{l_1}$, $x \vee \bar{l_4}$ (extension $x \Leftrightarrow l_1 \vee l_4$)
- $x \vee l_2 \vee l_4$, $\bar{x} \vee l_1 \vee l_3$, $x \vee \bar{l_1}$, $x \vee \bar{l_3}$ (extension $x \Leftrightarrow l_1 \vee l_3$)

C'est à ce choix d'extension parmi 4 possibles à chaque résolution que peut se résumer l'augmentation de la combinatoire de la Résolution étendue par rapport à la Résolution standard. La Résolution à refragmentation peut par exemple facilement s'intégrer à un algorithme de type CDCL car chaque clause apprise peut s'interpréter en terme d'une suite de résolutions. Il suffit de remplacer chaque résolution par une résolution à refragmentation (à chaque fois que la résolvante serait longue) en choisissant une des 4 extensions selon une heuristique qu'il reste à trouver.

Il est à noter que la Résolution à refragmentation peut sembler ne pas véritablement différer d'une simple réécriture artificielle d'une clause quaternaire en deux clauses ternaires, à la manière dont on transforme toute instance SAT en instance 3-SAT. Ce serait vrai si la règle de refragmentation était

$$\frac{l \vee l_1 \vee l_2 \qquad \bar{l} \vee l_3 \vee l_4}{x \vee l_1 \vee l_3,\ \bar{x} \vee l_2 \vee l_4}$$

(sans les clauses $x \vee \bar{l_2}$ et $x \vee \bar{l_4}$). Il n'est pas difficile de montrer qu'avec cette règle de refragmentation "affaiblie", notre système ne serait équivalent qu'à la Résolution. Ce sont les deux clauses $x \vee \bar{l_2}$ et $x \vee \bar{l_4}$ de la règle de refragmentation qui donnent toute sa puissance à notre système : elles permettent d'obtenir d'autres clauses où x apparaît (l_2 et/ou l_4 ayant été remplacés par x).

5 Conclusion et perspectives

Nous avons montré que la Résolution étendue étroite (`ER3`), qui interdit la génération de résolvantes de longueur supérieure à 3, conserve toute la puissance de la Résolution étendue (`ER`). De plus, comme les instances de problèmes nécessitant un nombre superpolynômial d'extensions pour être résolus dans `ER3` seraient intraitables, il est raisonnable de vouloir borner polynômialement le nombre d'extensions. Il devient alors envisageable de définir des algorithmes basés uniquement sur la recherche des bonnes extensions, la suite des résolutions étroites à appliquer ensuite ne servant que de certificat polynômial d'insatisfiabilité. Ce type d'approche est nouveau car la combinatoire très élevée de `ER` ne permettait d'envisager jusqu'à présent qu'un nombre limité d'ajout d'extensions, ce qui bridait très fortement la puissance du système de preuve. Une perspective évidente de notre travail théorique est donc de définir des algorithmes (systématiques ou de réparation) efficaces basés sur cette approche. Mais comme cette approche impliquerait de définir des algorithmes entièrement nouveaux, nous avons aussi défini la Résolution à refragmentation, dans le but de permettre une intégration facile de la résolution étendue étroite dans n'importe quel algorithme utilisant la règle de résolution.

Nous avons donné les moyens théoriques de rendre l'utilisation de la Résolution étendue beaucoup plus efficace qu'auparavant. Mais il reste encore beaucoup à faire avant d'espérer obtenir un algorithme compétitif avec les meilleures techniques existantes. En effet, l'augmentation de la combinatoire de la Résolution étendue par rapport à la Résolution standard est dû au choix des extensions à faire, qui est une question difficile encore très peu traitée. La piste que nous allons suivre pour tenter de résoudre cette question est la suivante. Dans la mesure où nous devons trouver les extensions qui vont faire que la formule sera résoluble par une 3-saturation, il faut d'abord que nous puissions caractériser structurellement ce qu'est une instance résoluble par 3-saturation.

Références

[1] Gilles Audemard, George Katsirelos, and Laurent Simon. A restriction of extended resolution for clause learning sat solvers. In *AAAI*, 2010.

[2] Gilles Audemard and Laurent Simon. Gunsat : A greedy local search algorithm for unsatisfiability. In *IJCAI*, pages 2256–2261, 2007.

[3] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow - resolution made simple. *J. ACM*, 48(2) :149–169, 2001.

[4] Vasek Chvátal and Endre Szemerédi. Many hard examples for resolution. *J. ACM*, 35(4) :759–768, 1988.

[5] Stephen A. Cook. A short proof of the pigeon hole principle using extended resolution. *SIGACT News*, 8 :28–32, 1976.

[6] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3) :201–215, 1960.

[7] Armin Haken. The intractability of resolution. *Theor. Comput. Sci.*, 39 :297–308, 1985.

[8] Jinbo Huang. Extended clause learning. *Artif. Intell.*, 174(15) :1277–1284, 2010.

[9] M. Jarvisalo. On the relative efficiency of dpll and obdds with axiom and join. In *CP 2011*, pages 429–437, 2011.

[10] Robert A. Kowalski and Donald Kuehner. Linear resolution with selection function. *Artif. Intell.*, 2(3/4) :227–260, 1971.

[11] J. Krajicek. Speed-up for propositional frege systems via generalizations of proofs. *Commentationes Mathematicae Universitas Carolinae*, 30(1) :137–140, 1989.

[12] Chu Min Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *IJCAI (1)*, pages 366–371, 1997.

[13] Nicolas Prcovic. Résolution étendue et largeur de réfutation de formules sat. In *Septièmes Journées Francophones de Programmation par Contraintes*, pages 271–280, 2011.

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

[15] G. S. Tseitin. On the complexity of derivation in propositional calculus. In *Studies in Constructive Mathematics and Mathematical Logics*, pages 115–125, 1968.

[16] Alasdair Urquhart. Hard examples for resolution. *J. ACM*, 34(1) :209–219, 1987.