Optimisation par renommage dans la méthode de résolution

Thierry Boy de La Tour

THESE

présentée par

Thierry Boy de la Tour

pour obtenir le titre de DOCTEUR

de l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE
(Arrêté ministériel du 23 novembre 1988)

en INFORMATIQUE

Optimisations par Renommage dans la Méthode de Résolution

Date de soutenance : lundi 21 janvier 1991

Composition du jury :	Ricardo	Caferra	directeur
	Jieh	Hsiang	rapporteur
	Claude	Kirchner	rapporteur
	Michael	Rusinowitch	examinateur
	Mark	Stickel	examinateur
	Jean-Pierre	Verjus	président

Thèse préparée au sein du Laboratoire d'Informatique Fondamentale
et d'Intelligence Artificielle

Remerciements

Je dois tout d'abord remercier Jean-Pierre Verjus, qui a bien voulu nous faire l'honneur de présider le jury.

Je tiens ensuite à remercier Jieh Hsiang et Claude Kirchner, qui ont accepté le rôle difficile de rapporteur, et ce malgré des délais relativement courts, apportant ainsi à cette thèse la garantie de leur grande compétence scientifique. Jieh Hsiang a, de plus, du s'astreindre à lire mon français, langue qui n'est pas la sienne.

Je remercie particulièrement Michael Rusinowitch et Mark Stickel qui ont eu la gentillesse, en acceptant de faire partie du jury, d'apporter à ce travail le soutien de leur renommée. Je suis spécialement sensible à l'honneur que nous fait Mark Stickel en acceptant, pour la première fois, de faire partie du jury d'une thèse française, malgré le problème de la langue et la contrainte d'un déplacement conséquent.

Je remercie également les membres du projet ATINF, notamment Gilles Chaminade et Nicolas Zabel, qui m'ont aidé et soutenu dans ce travail, grâce à leur disponibilité et à leur compétence scientifique. L'excellent travail d'implémentation de Gilles Chaminade a été indispensable à la validation pratique de mes idées, et je lui dois, pour une large part, le dernier chapitre de cette thèse.

Je veux aussi remercier Philippe Jorrand, pour avoir su faire du laboratoire qu'il dirige un environnement agréable et efficace, condition indispensable à tout travail de recherche.

Je tiens enfin à remercier Ricardo Caferra, qui s'est révélé un directeur de thèse exceptionnel. Il a su offrir, sans jamais l'imposer, tout ce qui pouvait manquer à un chercheur débutant : une large connaissance de la littérature, une vue d'ensemble du sujet, des idées fécondes, l'insertion dans la communauté internationale, et a surtout su répondre, par son amitié et sa ténacité sans faille, à la fameuse "angoisse de la page blanche". Je lui dois bien plus que ce que je peux présenter ici.

Préface

On connaît l'importance de la méthode de résolution en déduction automatique, qui n'a cessé d'être étudiée et raffinée depuis son apparition en 1965, dans [Rob65], et ce malgré le développement d'autres méthodes. Celle-ci n'est cependant pas dépourvue d'inconvénients, et on pourrait voir dans le foisonnement des travaux consacrés aux stratégies de résolution un indice des limites qui lui seraient inhérentes. L'obligation qui est faite d'inférer exclusivement des clauses, si elle entraîne une grande efficacité de calcul, peut sembler trop restrictive et inhiber toute possibilité de recul par rapport au problème à résoudre. Dans le but d'approcher les extraordinaires capacités du démonstrateur humain, il semble indispensable de traiter les questions telles qu'elles sont naturellement posées. Les travaux en démonstration automatique s'articulent souvent autour de cette frontière, séparant d'une part l'utilisation de la forme clausale, parfois d'une autre forme normale (par exemple [Hsi85]), avec peu de règles d'inférence, et d'autre part l'emploi d'un grand nombre de règles d'inférence sur des formules quelconques.

Si la forme clausale ne va pas de soi, on comprend l'importance que peut avoir le problème de la transformation d'une formule quelconque en un ensemble de clauses. Or la transformation classique, celle qui est utilisée dans la plupart des démonstrateurs par résolution, présente des inconvénients majeurs, qui peuvent être résumés en disant que sa complexité en taille est, dans le pire des cas, exponentielle. Un théorème de taille modeste peut ainsi être converti en un ensemble gigantesque de clauses, qu'aucun démonstrateur ne saurait gérer. Ce fait semble bien être à même de compenser négativement l'efficacité de la résolution, et annuler ainsi le seul intérêt dont elle est universellement accréditée.

Il est alors tout à fait étonnant de constater à quel point ce problème a peu été traité dans la littérature consacrée à la résolution ; il n'est même pas évoqué dans un ouvrage aussi synthétique que [Wos88]. On ne peut pourtant prétendre obliger l'utilisateur d'un démonstrateur à énoncer ses problèmes sous forme clausale. Un démonstrateur par résolution peut également être utilisé sur des formules produites automatiquement, qui ne sont pas sous forme clausale, et qui doivent bien être transformées — particulièrement en vérification de programmes, où l'ensemble de clauses produit est souvent trop important pour être utilisable.

Plus étonnant encore est qu'une solution à ce problème existe depuis 1968 et a été depuis largement méconnue : il apparaît en effet dans [Tse68] une transformation *linéaire* en taille, pour le calcul propositionnel. On peut l'étendre facilement au

premier ordre — où elle devient quadratique —, ce qui n'a été fait que beaucoup plus tard, principalement dans [PG86] — après quelques travaux moins approfondis, et tous postérieurs à 1980. Une raison en est sans doute la difficulté de [Tse68], et aussi qu'il ne s'agit pas du résultat principal de cet article.

Cette transformation utilise une technique simple introduite en 1920 par Skolem (voir [Sko67]), qui consiste à introduire des définitions et à les utiliser — ce que nous appelons *renommage* — d'une façon systématique. Aujourd'hui encore, elle ne reçoit pas l'attention qu'elle mérite, peut-être parce que les résultats expérimentaux obtenus dans [PG86] sont mitigés. Il n'est en effet pas évident qu'une forme clausale réduite, mais contenant plus de littéraux (issus des définitions), soit plus facile à réfuter que la forme clausale classique ; on est même certain du contraire si l'on applique une stratégie inadéquate. De plus, cette forme clausale classique n'est pas toujours plus longue que celle de [PG86], car l'amélioration sur ce point ne porte que sur la complexité dans le pire des cas.

On ne peut pourtant douter de l'intérêt de cette technique, et la raison fondamentale se trouve justement dans [Tse68]. Il y est montré qu'en ajoutant à la règle de résolution, dans le calcul propositionnel, une règle dite d'*extension*, on diminue exponentiellement la complexité en taille des plus courtes preuves. Cette règle d'extension consiste à rajouter à un ensemble de clauses la définition d'un nouveau symbole. Ce n'est pas le renommage, car ce symbole est défini au moyen d'une formule quelconque : on obtient plutôt un équivalent clausal de la règle de "cut" dans les systèmes de Gentzen. La formule introduite joue le rôle d'hypothèse. Le problème de ces règles est bien connu : elles ne préservent pas la propriété de la sous-formule, ce qui contraindrait un démonstrateur qui voudrait les utiliser à des capacités analogues à celles de l'"oracle" des algorithmes non-déterministes.

Cependant, Tseitin précise que dans une preuve, toutes les applications de la règle d'extension peuvent être placées avant toutes les applications de la règle de résolution — c'est pourquoi il ne la considère pas comme une règle d'inférence. On peut donc en envisager l'application au sein du processus de transformation. On peut également en restreindre l'emploi de façon à limiter les problèmes d'oracle, la restriction la plus naturelle étant de ne définir de nouveaux symboles qu'à partir des formules existantes, ce qui permet de retrouver la propriété de la sous-formule. L'oracle doit alors choisir parmi ces sous-formules, qui sont en nombre fini. On peut enfin remplacer chaque sous-formule utilisée par le symbole défini, ce qui permet, en modifiant l'ensemble de clauses autrement que par ajout, de guider plus sûrement

le démonstrateur vers la plus courte preuve. Le résultat — on l'aura deviné — est précisément le renommage.

Nous tentons dans la suite de résoudre le problème d'oracle ainsi posé, en nous plaçant d'emblée en logique du premier ordre, bien que les résultats de [Tse68] soient restreints au calcul propositionnel.

Nous définissons dans le premier chapitre la transformation primitive que nous utiliserons dans la suite: le renommage d'une, puis d'un ensemble de sous-formules. Nous y démontrons, dans un cadre théorique général, qu'il est valide de l'utiliser à des fins de démonstration formelle. Sans employer de résultats non démontrés, ainsi que dans le reste de la thèse — sauf au chapitre 7 —, nous supposons ici connus les rudiments de la logique du premier ordre.

Dans le deuxième chapitre, c'est la mise sous forme clausale standard qui est — succinctement — définie et — abondamment — explorée. Ce chapitre n'a pas un but didactique, et ne saurait convenir à un lecteur ignorant tout de cette transformation. Nous y démontrons un ensemble de résultats qui nous seront utiles par la suite, particulièrement pour l'évaluation de certaines complexités. Certains sont connus, comme la complexité exponentielle du nombre de clauses — bien qu'il soit rare d'en trouver démonstration —, d'autres sont originaux, comme pour la skolemisation ou la longueur des clauses.

C'est au troisième chapitre que nous associons les deux transformations précédentes. Après un historique introductif à cette technique, nous donnons une preuve formelle du principal résultat de [PG86] — dont la preuve manque de rigueur, et utilise une hypothèse inutile. Dans le but de remédier à certains inconvénients de cette transformation, constatés en introduction, nous en développons ensuite une nouvelle, ce qui constitue le cœur de la thèse. Nous définissons en fait un ensemble de transformations à partir de l'idée d'essayer de minimiser le nombre de clauses. La fin de ce chapitre est consacrée aux résultats de complexité communs à toutes ces transformations.

Le quatrième chapitre contient sans doute le résultat le plus intéressant de cette thèse, et en tout cas le plus difficile à obtenir. Nous démontrons qu'un sous-ensemble particulier de nos transformations, les renommages descendants, permettent d'obtenir le nombre minimal de clauses parmi toutes les formes clausales que la technique du renommage permet d'obtenir — depuis la forme clausale standard jusqu'à celle de [PG86]. Ceci n'est cependant acquis que pour une classe restreinte de formules : celles ne contenant pas de symbole d'équivalence. Ce résultat, peu

intuitif, demande une formalisation poussée. Une version abrégée de ce chapitre, et de la deuxième moitié du précédent, se trouve dans [BdlT90].

Dans le cinquième chapitre, nous développons une autre transformation, à partir de l'idée similaire, mais plus complexe, de minimiser le nombre d'occurrences de littéraux dans la forme clausale. Nous essayons alors de calquer les résultats obtenus pour la minimisation du nombre de clauses, ce qui se fait sans trop de difficulté en ce qui concerne la complexité — qui s'améliore légèrement. Cela se révèle par contre impossible quant à ceux du quatrième chapitre, et nous n'essayons donc pas de définir de nouvelles restrictions amenant des résultats d'optimalité qui seraient sans doute encore plus difficiles à prouver que ceux déjà obtenus.

Ces transformations étant destinées à une utilisation pratique, nous en donnons au sixième chapitre une définition algorithmique répondant aux exigences naturelles d'efficacité. Une partie du chapitre est consacrée à prouver que la transformation ainsi calculée correspond à un renommage descendant, et qu'il y a donc bien optimalité du nombre de clauses. Les résultats de correction et de complexité abordés ici le sont plus complètement dans [BdlT89], mais sur un algorithme légèrement différent.

Enfin, nous tentons au septième chapitre une comparaison de ces différentes transformations quant à leur aptitude à réduire la longueur des réfutations par résolution. Aucun résultat définitif n'est acquis en ce domaine, même si nous montrons que le renommage de certaines sous-formules est forcément néfaste, et nous sommes donc réduits à établir cette comparaison sur un certain nombre de points annexes qui nous semblent significatifs, et en particulier sur des expérimentations, réalisées dans le cadre de l'ATelier d'INFérence. Ce chapitre est largement inspiré de [BdlTC90].

Chapitre 1

Expansion de formules

Le temps viendra bientôt qu'à contempler ces êtres
Tu n'auras plus de peine ; et tu en jouiras
Autant que la nature à sentir te dispose.

Dante

1.1 Notations et préliminaires

Un *langage du premier ordre* $\mathcal{L}$ est un ensemble de symboles de prédicats et de fonctions, à chacun étant attribué un entier par la fonction α, appelée *arité*[1]. Un symbole de fonction d'arité nulle est appelé *symbole de constante*, un symbole de prédicat d'arité nulle est appelé *symbole propositionnel.*

Une *$\mathcal{L}$-formule* est une chaîne de symboles construite sur le langage du premier ordre $\mathcal{L}$ en respectant l'arité, sur les symboles logiques $\neg, \forall, \exists, \wedge, \vee, \Rightarrow, \Leftrightarrow$, les paren-

[1] En théorie des langages formels, $\mathcal{L}$ est appelé une *signature*. Cette terminologie, provenant de [CK73], est justifiée par le fait que nous nous restreignons à un ensemble bien particulier de langages — les formules du premier ordre — dont chacun est alors déterminé par une signature $\mathcal{L}$.

thèses ouvrante et fermante, et un ensemble dénombrable X de symboles appelés *variables*, supposé fixé et disjoint de $\mathcal{L}$ et de l'ensemble des symboles logiques, en respectant les règles usuelles : on construit d'abord les *$\mathcal{L}$-termes* avec les symboles de fonction de $\mathcal{L}$ et les symboles de variables, puis les *$\mathcal{L}$-formules atomiques* à partir des symboles de prédicats de $\mathcal{L}$ et les $\mathcal{L}$-termes, enfin les $\mathcal{L}$-formules à partir des formules atomiques, des symboles logiques et des variables (pour les quantificateurs). Les formules sont écrites en notation infixée. Les conjonctions et disjonctions admettent un nombre quelconque d'arguments (au moins un cependant). De même les quantificateurs portent sur un nombre quelconque de variables (au moins une). De plus, nous imposons la restriction suivante : toute variable quantifiée apparaît libre et non liée dans le champ de son quantificateur, et n'apparaît nulle part ailleurs.

Les formules seront en général désignées par les lettres ψ et φ, éventuellement primées et indicées. On appelle *sous-formule* d'une formule ψ toute sous-chaîne de ψ qui est une formule, et on dira également dans ce cas que ψ est une *sur-formule* de φ. On note $SF(\psi)$ le multi-ensemble des sous-formules de ψ (deux occurrences distinctes d'une même formule dans ψ sont considérées comme des sous-formules distinctes). On note $SF^*(\psi) = SF(\psi) - \{\psi\}$. $\varphi \sqsubseteq \psi$ ou $\psi \sqsupseteq \varphi$ signifient $\varphi \in SF(\psi)$, et de même $\varphi \sqsubset \psi$ ou $\psi \sqsupset \varphi$ signifient $\varphi \in SF^*(\psi)$. On appelle *sous-formule directe* de ψ les éléments maximaux (au sens de l'ordre $\sqsubseteq$) de $SF^*(\psi)$.

Le complémentaire d'une formule φ, noté φ^c, est $\neg\varphi$ si φ n'est pas une négation, et φ' si $\varphi = \neg\varphi'$. $|\varphi|$ dénote la longueur de φ, dans laquelle tout symbole a une longueur 1, et les parenthèses ne sont pas comptées. L'ensemble des variables libres d'une formule φ est noté $FV(\varphi)$. Si $FV(\varphi) = \emptyset$, on dit que φ est *close*. On appelle *conjoint* (resp. *disjoint*) de ψ toute sous-formule $\varphi \sqsubset \psi$ telle que toute formule φ' vérifiant $\varphi \sqsubset \varphi' \sqsubseteq \psi$ est une conjonction (resp.disjonction).

Un $\mathcal{L}$-modèle est un doublet $\mathcal{M} = \langle D, \mathcal{I} \rangle$, où D est un ensemble non vide et $\mathcal{I}$ une fonction qui à chaque symbole de fonction d'arité n de $\mathcal{L}$ (symbole de constante si $n = 0$) associe une fonction de D^n vers D (un élément de D si $n = 0$), et à chaque symbole de prédicat d'arité n de $\mathcal{L}$ (symbole propositionnel si $n = 0$) associe un sous-ensemble de D^n ($\emptyset$ ou $\mathbf{1}$ si $n = 0$).

A toute fonction ν de X vers D on associe alors la fonction $\mathcal{I}_\nu$, qui associe à chaque $\mathcal{L}$-terme une valeur dans D, définie inductivement par : pour chaque symbole de constante a de $\mathcal{L}$, on a $\mathcal{I}_\nu(a) = \mathcal{I}(a)$, pour chaque symbole de variable x, on a $\mathcal{I}_\nu(x) = \nu(x)$, et pour tout symbole de fonction f de $\mathcal{L}$, pour tous $\mathcal{L}$-termes $t_1..t_{\alpha(f)}$, $\mathcal{I}_\nu(f(t_1..t_{\alpha(f)})) = \mathcal{I}(f)(\mathcal{I}_\nu(t_1)..\mathcal{I}_\nu(t_{\alpha(f)}))$.

Soit P un symbole de prédicat de $\mathcal{L}$, et $t_1..t_{\alpha(P)}$ des $\mathcal{L}$-termes, on note $\mathcal{M}, \nu \models P(t_1..t_{\alpha(P)})$ si et seulement si $\langle \mathcal{I}_\nu(t_1)..\mathcal{I}_\nu(t_{\alpha(P)})\rangle \in \mathcal{I}(P)$ ($\mathcal{I}(P) = \mathbf{1}$ si $\alpha(P) = 0$). On étend inductivement la relation à toutes les $\mathcal{L}$-formules de la façon suivante :

- $\mathcal{M}, \nu \models \neg\varphi$ si et seulement si on n'a pas $\mathcal{M}, \nu \models \varphi$
- $\mathcal{M}, \nu \models \varphi_1 \wedge .. \wedge \varphi_n$ si et seulement si $\forall i \in \{1..n\}$, on a $\mathcal{M}, \nu \models \varphi_i$
- $\mathcal{M}, \nu \models \varphi_1 \vee .. \vee \varphi_n$ si et seulement si $\exists i \in \{1..n\}$, tel que $\mathcal{M}, \nu \models \varphi_i$
- $\mathcal{M}, \nu \models \varphi \Rightarrow \varphi'$ si et seulement si on a $\mathcal{M}, \nu \models \varphi'$, ou on n'a pas $\mathcal{M}, \nu \models \varphi$
- $\mathcal{M}, \nu \models \varphi \Leftrightarrow \varphi'$ si et seulement si on a $\mathcal{M}, \nu \models \varphi$ et $\mathcal{M}, \nu \models \varphi'$, ou ni l'une ni l'autre.
- $\mathcal{M}, \nu \models \forall x_1..x_n.\varphi$ si et seulement si $\forall e_1..e_n \in D$, on a $\mathcal{M}, \nu[x_i \leftarrow e_i]_{i=1}^n \models \varphi$
- $\mathcal{M}, \nu \models \exists x_1..x_n.\varphi$ si et seulement si $\exists e_1..e_n \in D$ tel que $\mathcal{M}, \nu[x_i \leftarrow e_i]_{i=1}^n \models \varphi$

où $\nu[x_i \leftarrow e_i]_{i=1}^n$ désigne la fonction identique à ν en dehors de $\{x_1..x_n\}$, et identique à $\{\langle x_1, e_1\rangle..\langle x_n, e_n\rangle\}$ sur cet ensemble.

On note $\mathcal{M} \models \varphi$ si et seulement si on a $\mathcal{M}, \nu \models \varphi$ pour toute fonction ν. On dit d'une $\mathcal{L}$-formule φ qu'elle est *valide* si et seulement si pour tout $\mathcal{L}$-modèle $\mathcal{M}$ on a $\mathcal{M} \models \varphi$, et on note alors $\models \varphi$. On dit qu'elle est *satisfaisable* si et seulement si il existe un $\mathcal{L}$-modèle $\mathcal{M}$ et une fonction ν telle que $\mathcal{M}, \nu \models \varphi$. On montre alors aisément qu'une formule est valide si et seulement si sa négation est insatisfaisable. On constate également que si φ est close, $\mathcal{M}, \nu \models \varphi$ est indépendant de ν, donc équivalent à $\mathcal{M} \models \varphi$.

Enfin, on note $\varphi \models \varphi'$ si et seulement si pour tout modèle $\mathcal{M}$ et toute fonction ν, si $\mathcal{M}, \nu \models \varphi$ alors $\mathcal{M}, \nu \models \varphi'$.

Nous arrivons maintenant à des définitions moins usuelles, que nous introduisons au fur et à mesure de leur utilisation. En cas de problème, le lecteur est convié à utiliser l'index situé en fin de volume. La définition qui suit nous sera utile dans un argument inductif :

Définition. La *profondeur dans* ψ d'une sous-formule $\varphi \sqsubseteq \psi$ est

$$d_\psi(\varphi) = |\{\varphi' \sqsubseteq \psi / \varphi \sqsubset \varphi'\}|$$

$\diamond$

Nous utiliserons aussi la notion connue de polarité d'une sous-formule, en y apportant toutefois une légère modification pour l'équivalence :

Définition. La polarité de $\varphi \sqsubseteq \psi$ dans ψ, notée $pol(\varphi, \psi)$, est définie par :

- $pol(\varphi_i, \psi) = +1$ si $\psi = \varphi_i$, ou $\psi = \varphi_1 \wedge \ldots \wedge \varphi_n$, ou $\psi = \varphi_1 \vee \ldots \vee \varphi_n$, ou $\psi = \varphi_1 \Rightarrow \varphi_2$ et $i = 2$, ou $\psi = Q x_1..x_n.\varphi_1$ $(i = 1, Q \in \{\forall, \exists\})$.
- $pol(\varphi_1, \psi) = -1$ si $\psi = \neg\varphi_1$ ou $\psi = \varphi_1 \Rightarrow \varphi_2$.
- $pol(\varphi_i, \psi) = 0$ si $\psi = \varphi_1 \Leftrightarrow \varphi_2$.
- $\forall \varphi \sqsubseteq \varphi' \sqsubseteq \varphi'', pol(\varphi, \varphi'') = pol(\varphi, \varphi').pol(\varphi', \varphi'')$ (relation de propagation).

$\diamond$

Intuitivement, dans une formule linéaire — sans équivalence — la polarité de φ dans ψ est la parité du nombre de négations apparaissant au dessus de φ dans ψ. Cette notion s'apparente à la loi de la double négation : n négations consécutives peuvent être remplacées par n modulo 2 négation. Enfin, une notation très utile :

Définition. Soient φ,φ' deux formules ayant même variables libres $x_1..x_n$, alors :

- $\sigma^+(\varphi, \varphi')$ dénote la formule $\forall x_1..x_n.\varphi \Rightarrow \varphi'$.
- $\sigma^-(\varphi, \varphi')$ dénote la formule $\sigma^+(\varphi', \varphi)$.
- $\sigma^0(\varphi, \varphi')$ dénote la formule $\forall x_1..x_n.\varphi \Leftrightarrow \varphi'$.

$\diamond$

L'exposant du symbole σ doit être compris comme étant une polarité (abus de notation par souci d'élégance). Cette notation relie la polarité avec certaines formes d'implications, ce qui permet une formulation simple du théorème de monotonie ci-dessous.

Théorème 1.1 *Soient ψ une formule, φ, φ' deux formules telles que $\varphi \sqsubseteq \psi$ et $FV(\varphi) = FV(\varphi')$, si $s = pol(\varphi, \psi)$, alors $\sigma^s(\varphi, \varphi') \models \sigma^+(\psi, \psi[\varphi \leftarrow \varphi'])$.*

Preuve. On montre par récurrence sur la profondeur n de φ dans ψ que $\forall n \in I\!N, \forall \psi, \varphi, \varphi'$,

$$(\varphi \sqsubseteq \psi \wedge FV(\varphi) = FV(\varphi') \wedge d_\psi(\varphi) = n) \Rightarrow \sigma^{pol(\varphi,\psi)}(\varphi, \varphi') \models \sigma^+(\psi, \psi[\varphi \leftarrow \varphi'])$$

Si $n = 0$, on a $\psi = \varphi$, et le résultat est évident car $\sigma^+(\varphi, \varphi')$ et $\sigma^+(\psi, \psi[\varphi \leftarrow \varphi'])$ sont identiques.

On suppose la propriété vraie pour n, soient ψ, φ, φ' telles que $\varphi \sqsubseteq \psi, FV(\varphi) = FV(\varphi')$ et $d_\psi(\varphi) = n + 1$, et on note $s = pol(\varphi, \psi)$. Nous examinons les différents cas possibles :

- $\psi = \neg\psi'$, on a $d_{\psi'[\varphi\leftarrow\varphi']}(\varphi') = n$, donc par hypothèse de récurrence on a $\sigma^{-s}(\varphi',\varphi) \models \sigma^{+}(\psi'[\varphi \leftarrow \varphi'],\psi')$. Mais $\sigma^{-s}(\varphi',\varphi) = \sigma^{s}(\varphi,\varphi')$, et on peut énoncer la loi de contraposition de la façon suivante :

$$\sigma^{+}(\psi'[\varphi\leftarrow\varphi'],\psi') \models \sigma^{+}(\neg\psi',\neg\psi'[\varphi\leftarrow\varphi'])$$

donc $\sigma^{s}(\varphi,\varphi') \models \sigma^{+}(\psi,\psi[\varphi\leftarrow\varphi'])$.

- $\psi = \psi_1 \wedge \ldots \wedge \psi_n$, soit $i \in \{1..n\}$ tel que $\varphi \sqsubseteq \psi_i$, on a $d_{\psi_i}(\varphi) = n$, donc par hypothèse de récurrence $\sigma^{s}(\varphi,\varphi') \models \sigma^{+}(\psi_i,\psi_i[\varphi \leftarrow \varphi'])$. En utilisant la tautologie propositionnelle $\models (p \Rightarrow q) \Rightarrow ((p \wedge r) \Rightarrow (q \wedge r))$, on obtient $\sigma^{s}(\varphi,\varphi') \models \forall x_1..x_n.(\psi_1 \wedge \ldots \wedge \psi_i \wedge \ldots \wedge \psi_n) \Rightarrow (\psi_1 \wedge \ldots \wedge \psi_i[\varphi\leftarrow\varphi'] \wedge \ldots \wedge \psi_n)$ c'est à dire $\sigma^{s}(\varphi,\varphi') \models \sigma^{+}(\psi,\psi[\varphi\leftarrow\varphi'])$.

- $\psi = \psi_1 \vee \ldots \vee \psi_n$, on procède de même, en utilisant la tautologie $\models (p \Rightarrow q) \Rightarrow ((p \vee r) \Rightarrow (q \vee r))$.

- $\psi = \psi_1 \Rightarrow \psi_2$, on utilise la tautologie $\models (p \Rightarrow q) \Rightarrow ((r \Rightarrow p) \Rightarrow (r \Rightarrow q))$ si $\varphi \sqsubseteq \psi_2$, et la tautologie $\models (p \Rightarrow q) \Rightarrow ((q \Rightarrow r) \Rightarrow (p \Rightarrow r))$ si $\varphi \sqsubseteq \psi_1$ (en partant de l'hypothèse de récurrence $\sigma^{-s}(\varphi',\varphi) \models \sigma^{+}(\psi_1[\varphi \leftarrow \varphi'],\psi_1)$).

- $\psi = \forall y_1..y_p.\psi'$, soient $x_1..x_n$ les variables libres de ψ, et $z_1..z_k$ les variables libres de ψ', on a $\{z_1..z_k\} = \{x_1..x_n\} \cup \{y_1..y_p\}$. Par hypothèse de récurrence $\sigma^{s}(\varphi,\varphi') \models \sigma^{+}(\psi',\psi'[\varphi \leftarrow \varphi'])$, cette dernière formule étant $\forall z_1..z_k.\psi' \Rightarrow \psi'[\varphi\leftarrow\varphi']$. Donc :

$$\begin{aligned}
\sigma^{s}(\varphi,\varphi') &\models \forall x_1..x_n y_1..y_p.\psi' \Rightarrow \psi'[\varphi\leftarrow\varphi'] \\
&\models \forall x_1..x_n.(\exists y_1..y_p.\psi') \Rightarrow (\forall y_1..y_p.\psi'[\varphi\leftarrow\varphi']) \\
&\models \forall x_1..x_n.(\forall y_1..y_p.\psi') \Rightarrow (\forall y_1..y_p.\psi'[\varphi\leftarrow\varphi'])
\end{aligned}$$

(à partir de la tautologie $\models \forall x.P(x) \Rightarrow \exists x.P(x)$). La dernière formule est $\sigma^{+}(\psi,\psi[\varphi\leftarrow\varphi'])$.

- $\psi = \exists y_1..y_p.\psi'$, on obtient également par hypothèse de récurrence que :

$$\sigma^{s}(\varphi,\varphi') \models \forall x_1..x_n.(\exists y_1..y_p.\psi') \Rightarrow (\forall y_1..y_p.\psi'[\varphi\leftarrow\varphi'])$$

et on obtient $\sigma^{s}(\varphi,\varphi') \models \forall x_1..x_n.(\exists y_1..y_p.\psi\varphi) \Rightarrow (\exists y_1..y_p.\psi'[\varphi\leftarrow\varphi'])$.

- $\psi = \psi_1 \Leftrightarrow \psi_2$, soient $i \in \{1,2\}$ tel que $\varphi \sqsubseteq \psi_i$, et $s' = pol(\varphi, \psi_i)$, on a par hypothèse de récurrence $\sigma^{s'}(\varphi,\varphi') \models \sigma^+(\psi_i, \psi_i[\varphi \leftarrow \varphi'])$, mais également $\sigma^{s'}(\varphi',\varphi) \models \sigma^+(\psi_i[\varphi \leftarrow \varphi'], \psi_i)$, or $\sigma^0(\varphi,\varphi') \models \sigma^{s'}(\varphi,\varphi') \wedge \sigma^{s'}(\varphi',\varphi)$, donc :

$$\begin{aligned} \sigma^0(\varphi,\varphi') &\models \sigma^+(\psi_i, \psi_i[\varphi \leftarrow \varphi']) \wedge \sigma^+(\psi_i[\varphi \leftarrow \varphi'], \psi_i) \\ &\models \forall x_1..x_n.\psi_i \Leftrightarrow \psi_i[\varphi \leftarrow \varphi'] \end{aligned}$$

A partir de la tautologie $\models (p \Leftrightarrow q) \Rightarrow ((p \Leftrightarrow r) \Rightarrow (q \Leftrightarrow r))$, on conclut

$$\sigma^0(\varphi,\varphi') \models \forall x_1..x_n.\psi \Rightarrow \psi[\varphi \leftarrow \varphi']$$

La récurrence est établie, et donc $\forall \psi, \varphi, \varphi'$ tels que $\varphi \sqsubseteq \psi$ et $FV(\varphi) = FV(\varphi')$, soit $n = d_\psi(\varphi)$, d'après ce qui précède on a $\sigma^{pol(\varphi,\psi)}(\varphi,\varphi') \models \sigma^+(\psi, \psi[\varphi \leftarrow \varphi'])$ $\dashv$

1.2 Expansion de formules

Nous nous posons maintenant le problème de *l'expansion* des formules du premier ordre, c'est à dire de la façon dont on peut étendre le langage utilisé pour exprimer une formule, dans le but de transformer celle-ci. Notre but est de préserver la satisfaisabilité ou la validité entre une formule et ses expansions. Ceci nous amène à la définition suivante :

Définition. Soient $\mathcal{L}$, $\mathcal{L}'$ deux langages du premier ordre, tels que $\mathcal{L} \subset \mathcal{L}'$ ($\mathcal{L}'$ est une expansion de $\mathcal{L}$), φ une $\mathcal{L}$-formule et φ' une $\mathcal{L}'$-formule ayant mêmes variables libres. On dit que φ' est une *expansion conservative* de φ si et seulement si pour tout $\mathcal{L}$-modèle $\mathcal{M}$, il existe une expansion $\mathcal{M}'$ de $\mathcal{M}$ à $\mathcal{L}'$ telle que $\mathcal{M}' \models \sigma^0(\varphi,\varphi')$. (i.e. φ et φ' ont exactement la même valuation dans $\mathcal{M}'$). $\diamond$

Nous rappelons, suivant [CK73], que dans une expansion $\mathcal{M}'$ de $\mathcal{M}$ à $\mathcal{L}'$, les symboles interprétés dans $\mathcal{M}$ (i.e. les éléments de $\mathcal{L}$), conservent leur interprétation dans $\mathcal{M}'$, qui est un $\mathcal{L}'$-modèle. C'est donc à partir de ces notions d'expansion sur des langages et des modèles que nous avons introduit une notion similaire sur des formules. Un exemple important : on sait que si $\exists x.\varphi$ a pour variables libres $x_1..x_n$, alors $\varphi[x \leftarrow f(x_1..x_n)]$ est une expansion conservative de $\exists x.\varphi$ (où $\mathcal{L}' - \mathcal{L} = \{f\}$). Le théorème suivant montre comment l'expansion conservative permet de préserver la satisfaisabilité :

Théorème 1.2 *Soient ψ une $\mathcal{L}$-formule, avec $\varphi \sqsubseteq \psi$ et φ' une expansion conservative de φ, ψ est satisfaisable si et seulement si $\psi[\varphi \leftarrow \varphi'] \wedge \sigma^{pol(\varphi,\psi)}(\varphi',\varphi)$ est satisfaisable.*

Preuve. Soit $s = pol(\varphi, \psi)$.

Si $\psi[\varphi \leftarrow \varphi'] \wedge \sigma^s(\varphi', \varphi)$ est satisfaisable, soient $\mathcal{M}'$ un $\mathcal{L}'$-modèle et ν tels que $\mathcal{M}', \nu \models \psi[\varphi \leftarrow \varphi'] \wedge \sigma^s(\varphi', \varphi)$. On a donc $\mathcal{M}', \nu \models \sigma^s(\varphi', \varphi)$, et en appliquant le théorème 1.1, $\mathcal{M}', \nu \models \sigma^+(\psi[\varphi \leftarrow \varphi'], \psi)$. Mais $\mathcal{M}', \nu \models \psi[\varphi \leftarrow \varphi']$, donc $\mathcal{M}', \nu \models \psi$, et ψ est satisfaisable.

Réciproquement, si ψ est satisfaisable, soient $\mathcal{M}$ un $\mathcal{L}$-modèle et ν tels que $\mathcal{M}, \nu \models \psi$. Comme φ' est une expansion conservative de φ, il existe une expansion $\mathcal{M}'$ de $\mathcal{M}$ à $\mathcal{L}'$ telle que $\mathcal{M}' \models \sigma^0(\varphi, \varphi')$, et on a $\mathcal{M}', \nu \models \psi$. Comme φ et φ' ont exactement la même valuation dans $\mathcal{M}'$, elles peuvent être remplacées l'une par l'autre, donc $\mathcal{M}', \nu \models \psi[\varphi \leftarrow \varphi']$. Mais on a $\sigma^0(\varphi, \varphi') \models \sigma^s(\varphi', \varphi)$, donc $\mathcal{M}' \models \sigma^s(\varphi', \varphi)$, et donc $\psi[\varphi \leftarrow \varphi'] \wedge \sigma^s(\varphi', \varphi)$ est satisfaisable. $\dashv$

Il est facile d'en déduire un résultat dual :

Corollaire 1.3 *Soient ψ une $\mathcal{L}$-formule, avec $\varphi \sqsubseteq \psi$ et φ' une expansion conservative de φ, ψ est valide si et seulement si $\sigma^{pol(\varphi,\psi)}(\varphi, \varphi') \Rightarrow \psi[\varphi \leftarrow \varphi']$ est valide.*

Preuve. Soit $s = pol(\varphi, \psi)$, ψ est valide si et seulement si $\neg\psi$ est insatisfaisable, donc d'après le théorème 1.2, si et seulement si $\neg\psi[\varphi \leftarrow \varphi'] \wedge \sigma^{-s}(\varphi', \varphi)$ est insatisfaisable, donc si et seulement si $\neg(\neg\psi[\varphi \leftarrow \varphi'] \wedge \sigma^{-s}(\varphi', \varphi))$ est valide. Or :

$$\begin{aligned} \models \neg(\neg\psi[\varphi \leftarrow \varphi'] \wedge \sigma^{-s}(\varphi', \varphi)) &\Leftrightarrow \psi[\varphi \leftarrow \varphi'] \vee \neg\sigma^{-s}(\varphi', \varphi) \\ &\Leftrightarrow \sigma^{-s}(\varphi', \varphi) \Rightarrow \psi[\varphi \leftarrow \varphi'] \\ &\Leftrightarrow \sigma^{s}(\varphi, \varphi') \Rightarrow \psi[\varphi \leftarrow \varphi'] \end{aligned}$$

$\dashv$

De l'exemple exposé ci-dessus, nous constatons que la skolemisation est un cas particulièrement simple du fait que $\models \sigma^+(\varphi[x \leftarrow f(x_1..x_n)], \exists x.\varphi)$, de sorte que l'on peut se débarrasser de ce conjoint, c'est à dire que le théorème 1.2 prouve que si $(\exists x.\varphi) \sqsubseteq \psi$ avec $pol(\exists x.\varphi, \psi) = +1$, alors ψ est satisfaisable si et seulement si $\psi[\exists x.\varphi \leftarrow \varphi[x \leftarrow f(x_1..x_n)]]$ est satisfaisable, ce qui constitue un résultat bien connu. Dualement, $\varphi[x \leftarrow f(x_1..x_n)]$ est une expansion conservative de $\forall x.\varphi$ (si φ' est une expansion conservative de φ, alors $\neg\varphi'$ est évidemment une expansion conservative de $\neg\varphi$), et comme $\models \sigma^-(\varphi[x \leftarrow f(x_1..x_n)], \forall x.\varphi)$, on déduit du théorème 1.2 que si $(\forall x.\varphi) \sqsubseteq \psi$ avec $pol(\forall x.\varphi, \psi) = -1$, alors ψ est satisfaisable si et seulement si $\psi[\forall x.\varphi \leftarrow \varphi[x \leftarrow f(x_1..x_n)]]$ est satisfaisable.

1.3 Le renommage

Nous nous intéressons maintenant à une expansion particulière dont l'intérêt n'est pas de simplifier le terme σ, mais de simplifier l'expansion elle-même : P étant un nouveau symbole de prédicat ($\mathcal{L}' - \mathcal{L} = \{P\}$), il est facile de voir que $P(x_1..x_n)$ est une expansion conservative de toute $\mathcal{L}$-formule φ dont les variables libres sont $x_1..x_n$. Nous en déduisons, d'après le théorème 1.2, que si $\varphi \sqsubseteq \psi$ alors ψ est satisfaisable si et seulement si $\psi[\varphi \leftarrow P(x_1..x_n)] \wedge \sigma^{pol(\varphi,\psi)}(P(x_1..x_n), \varphi)$ est satisfaisable. Cette transformation constitue la base même du renommage. Elle a été introduite en 1920 par Skolem, dans la preuve du résultat concernant l'existence d'une forme normale de Skolem (voir [Sko67]). On sait aujourd'hui prouver ce résultat sans l'aide du renommage, par exemple en utilisant le théorème 1.2 et en invoquant les résultats connus concernant les formes prénexes.

Dans la suite, nous nous intéresserons à des renommages multiples — de plusieurs sous-formules simultanément — et il nous faut pour éviter toute confusion distinguer les symboles de prédicats utilisés dans les expansions de sous-formules différentes.

Définition. A chaque *occurrence* d'une sous-formule φ d'une $\mathcal{L}$-formule ψ on associe un symbole de prédicat *unique* noté SkP_φ^ψ (prédicat de Skolem), tel que $SkP_\varphi^\psi \notin \mathcal{L}$. Si ψ' est un conjoint de ψ tel que $\varphi \sqsubseteq \psi'$, on identifie $SkP_\varphi^{\psi'}$ à SkP_φ^ψ — pour des raisons techniques qui apparaîtront dans la preuve du corollaire 1.4.

Si $FV(\varphi) = \{x_1..x_n\}$, on note SkL_φ^ψ le littéral[2] $SkP_\varphi^\psi(x_1..x_n)$ (littéral de Skolem). $\diamond$

La problématique des renommages multiples nous oblige à considérer des ensembles de sous-formules, ainsi que les substitutions de ces sous-formules par les littéraux de Skolem correspondants. Les très fréquentes utilisations que nous en ferons nous imposent d'en trouver une formulation simple :

Définition. On appelle *renommage* d'une formule ψ tout sous multi-ensemble R de $SF^*(\psi)$. On note $Inf_R(\varphi)$ l'ensemble $\max\{\varphi' \in R / \varphi' \sqsubset \varphi\}$ (ensemble des éléments maximaux au sens de l'ordre $\sqsubseteq$; on obtient un renommage de φ inclus dans R). Dualement, si $\{\varphi' \in R / \varphi \sqsubseteq \varphi'\} \neq \emptyset$, on note $Sup_R(\varphi)$ l'élément minimal[3] de cet ensemble.

---

[2] Pour éviter toute ambiguïté, on peut supposer que l'ensemble des variables est totalement ordonné, et que $x_1 < \ldots < x_n$.

[3] $(SF(\psi), \sqsubseteq)$ est un sup-demi-treillis complet.

Soit R un renommage d'une formule ψ, $\psi[R]$ désigne la formule

$$\psi[\varphi \leftarrow SkL^{\psi}_{\varphi}]_{\varphi \in Inf_R(\psi)}$$

$\diamond$

Cette dernière définition ne prend son sens qu'au regard de la suivante, qui explicite la *transformation* que permet d'effectuer ce que nous avons appelé un renommage. On peut cependant déjà constater que nous n'autorisons pas le renommage de la formule ψ elle-même, et ce pour des raisons d'élégance de notation. Cette restriction n'est absolument pas gênante, et nous verrons dans les chapitres suivants qu'il serait inutile de renommer ψ.

Définition. Soient ψ une formule, $\varphi \sqsubseteq \psi$, on note $Def_{\psi}(\varphi)$ la formule $\sigma^{pol(\varphi,\psi)}(SkL^{\psi}_{\varphi}, \varphi)$. Soit R un renommage de ψ, on note $Rnm(R,\psi)$ la formule[4]

$$\psi[R] \wedge \bigwedge_{\varphi \in R} Def_{\psi}(\varphi[R])$$

$\diamond$

On vérifie aisément que cette définition correspond, lorsque R est un singleton, à la formulation qui est donnée ci-dessus du théorème 1.2 avec l'expansion considérée (expansion par littéral de skolem). On en déduit le

Corollaire 1.4 *Soient ψ une formule et R un renommage de ψ, $Rnm(R,\psi)$ est satisfaisable si et seulement si ψ est satisfaisable.*

Preuve. Si $R = \emptyset$, c'est évident car $Rnm(\emptyset,\psi) = \psi$. Sinon, montrons que $\forall \varphi \in R, \models Rnm(R,\psi) \Leftrightarrow Rnm(\{\varphi\}, Rnm(R-\{\varphi\},\psi))$. En effet:

$$\begin{aligned}
& Rnm(\{\varphi\}, Rnm(R-\{\varphi\},\psi)) \\
&= Rnm(\{\varphi\}, \psi[R-\{\varphi\}] \wedge \bigwedge_{\xi \in R-\{\varphi\}} Def_{\psi}(\xi[R-\{\varphi\}])) \\
&= (\psi[R] \wedge \bigwedge_{\xi \in R-\{\varphi\}} Def_{\psi}(\xi[R])) \wedge Def_{Rnm(R-\{\varphi\},\psi)}(\varphi[R-\{\varphi\}]) \\
&= (\psi[R] \wedge \bigwedge_{\xi \in R-\{\varphi\}} Def_{\psi}(\xi[R])) \wedge Def_{\psi}(\varphi[R])
\end{aligned}$$

car $pol(\varphi,\psi) = pol(\varphi, Rnm(R-\{\varphi\},\psi))$, $Inf_R(\varphi) = Inf_{R-\{\varphi\}}(\varphi)$ et $\psi[R-\{\varphi\}]$ est un conjoint de $Rnm(R-\{\varphi\},\psi)$. Cette dernière formule est évidemment équivalente à $Rnm(R,\psi)$.

[4] L'ordre des définitions dans la conjonction n'importe pas ; on peut convenir de les ranger dans l'ordre préfixé des sous-formules de ψ.

D'autre part, puisque SkL_{φ}^{ψ} est une expansion conservative de φ, d'après le théorème 1.2, $\forall \psi' \sqsupseteq \varphi, \psi'$ est satisfaisable si et seulement si $Rnm(\varphi, \psi')$ l'est, donc $Rnm(R, \psi)$ est satisfaisable si et seulement si $Rnm(R - \{\varphi\}, \psi)$ l'est. R étant fini, on obtient par induction que $Rnm(R, \psi)$ est satisfaisable si et seulement si $Rnm(\emptyset, \psi) = \psi$ l'est. $\dashv$

Ce théorème autorise l'utilisation de cette transformation dans les méthodes de preuve qui procèdent par réfutation, et en particulier la résolution. De même que la skolemisation, elle ne préserve pas l'équivalence. Contrairement à elle, elle n'est pas indispensable à la mise sous forme clausale. C'est cependant dans cette perspective qu'elle prend tout son intéret!

Chapitre 2

La transformation *clausale*

L'espoir luit comme un caillou dans un creux.

Verlaine

2.1 Définition des transformations

Dans la mise sous forme clausale, nous utilisons tout d'abord une linéarisation (élimination des équivalences) dépendante de la polarité (voir [HLO+80]). En effet, nous constatons que la linéarisation classique n'est pas adaptée à la mise sous forme clausale dans le cas des équivalences de polarité négative :

$$\begin{aligned} \models \neg(P \Leftrightarrow Q) &\Leftrightarrow \neg(P \Rightarrow Q \wedge Q \Rightarrow P) \\ &\Leftrightarrow (P \wedge \neg Q) \vee (Q \wedge \neg P) \\ &\Leftrightarrow (P \vee Q) \wedge (P \vee \neg P) \wedge (\neg Q \vee Q) \wedge (\neg Q \vee \neg P) \\ &\Leftrightarrow (P \vee Q) \wedge (\neg P \vee \neg Q) \end{aligned}$$

On voit que ce procédé conduit à l'apparition de clauses tautologiques, donc

inutiles. Or, lorsque P et Q sont des formules relativement complexes, ces tautologies ne sont pas détectables par les méthodes de simplification classiques. C'est en particulier vrai si ces formules contiennent des quantificateurs, puisque ceux-ci sont skolemisés. Il n'y a plus alors de simplification possible puisque la négation de la skolemisée d'une formule φ n'est pas équivalente à la skolemisée de $\neg\varphi$.

Il convient donc d'effectuer cette simplification avant la skolemisation, c'est à dire directement pendant la linéarisation. Il suffit en effet de remplacer $P \Leftrightarrow Q$ par $(P \wedge Q) \vee (\neg P \wedge \neg Q)$ lorsque cette équivalence a une polarité négative. Pour les équivalences de polarité positive, on conserve bien entendu la linéarisation classique. Enfin, on applique cette règle dans un parcours descendant, de façon à ne rencontrer que des équivalences de polarité positive ou négative. Ceci donne la définition inductive suivante :

- si φ est atomique, $lin(\varphi, s) = \varphi$.
- si $\varphi = \varphi_1 \Leftrightarrow \varphi_2$, alors

$$lin(\varphi, s) = \begin{cases} (lin(\varphi_1, -1) \Rightarrow lin(\varphi_2, +1)) \wedge (lin(\varphi_2, -1) \Rightarrow lin(\varphi_1, +1)) \text{ si } s = +1 \\ (lin(\varphi_1, -1) \wedge lin(\varphi_2, -1)) \vee (\neg lin(\varphi_1, +1) \wedge \neg lin(\varphi_2, +1)) \text{ si } s = -1 \end{cases}$$

- sinon, $lin(\varphi, s)$ est la formule φ dans laquelle chaque sous-formule directe φ' de φ est remplacée par $lin(\varphi', s.pol(\varphi', \varphi))$.

$lin(\varphi, +1)$ constitue la linéarisation dépendante de la polarité, orientée vers la mise sous forme clausale. Nous la noterons *linéaire*. $lin(\varphi, -1)$ est une linéarisation orientée vers la mise sous forme normale disjonctive.

Les transformations qui suivent la linéarisation sont, dans l'ordre, la skolemisation, la mise sous forme prénexe, et la mise sous forme normale conjonctive. Les règles de skolemisation sont décrites au chapitre 1, et sont appliquées en descendant dans la formule. Nous noterons $Sklm$ cette transformation. Elle permet d'éliminer les quantificateurs existentiels de polarité positive et les quantificateurs universels de polarité négative. L'impossibilité d'éliminer les quantificateurs de polarité nulle justifie l'étape précédente permettant d'obtenir une formule *linéaire*, c'est à dire dépourvue d'équivalence.

La mise sous forme normale prénexe peut être très simplement définie par le système de réécriture ci-dessous (bien que non confluent, ce système convient parce qu'il termine, et que nous n'utiliserons que des propriétés vérifiées par toutes les formes normales).

$$\neg\forall x_1..x_p.\varphi \rightarrow \exists x_1..x_p.\neg\varphi$$
$$\neg\exists x_1..x_p.\varphi \rightarrow \forall x_1..x_p.\neg\varphi$$
$$(\forall x_1..x_p.\varphi_1)\Rightarrow\varphi_2 \rightarrow \exists x_1..x_p.(\varphi_1\Rightarrow\varphi_2)$$
$$(\exists x_1..x_p.\varphi_1)\Rightarrow\varphi_2 \rightarrow \forall x_1..x_p.(\varphi_1\Rightarrow\varphi_2)$$
$$\varphi_1\Rightarrow(Qx_1..x_p.\varphi_2) \rightarrow Qx_1..x_p.(\varphi_1\Rightarrow\varphi_2)$$
$$\varphi_1\wedge..\wedge\varphi_{i-1}\wedge(Qx_1..x_p.\varphi_i)\wedge\varphi_{i+1}\wedge..\wedge\varphi_n \rightarrow Qx_1..x_p.(\varphi_1\wedge..\wedge\varphi_n)$$
$$\varphi_1\vee..\vee\varphi_{i-1}\vee(Qx_1..x_p.\varphi_i)\vee\varphi_{i+1}\vee..\vee\varphi_n \rightarrow Qx_1..x_p.(\varphi_1\vee..\vee\varphi_n)$$
$$\forall x_1..x_p.\forall y_1..y_m.\varphi \rightarrow \forall x_1..x_p y_1..y_m.\varphi$$
$$\exists x_1..x_p.\exists y_1..y_m.\varphi \rightarrow \exists x_1..x_p y_1..y_m.\varphi$$

où $Q \in \{\forall, \exists\}$, et les φ_i (resp. x_i) dénotent évidemment des formules (resp. des variables[1]). Nous noterons cette transformation *prénexe*. On peut également définir la mise sous forme normale conjonctive par un système de réécriture, cette fois-ci convergent :

$$\neg\neg\varphi \rightarrow \varphi$$
$$\varphi_1\Rightarrow\varphi_2 \rightarrow \neg\varphi_1\vee\varphi_2$$
$$\neg(\varphi_1\wedge..\wedge\varphi_n) \rightarrow \neg\varphi_1\vee..\vee\neg\varphi_n$$
$$\neg(\varphi_1\vee..\vee\varphi_n) \rightarrow \neg\varphi_1\wedge..\wedge\neg\varphi_n$$
$$\varphi_1\wedge..\wedge\varphi_i\wedge(\bigwedge_{j=1}^{p}\varphi'_j)\wedge\varphi_{i+1}\wedge..\wedge\varphi_n \rightarrow \varphi_1\wedge..\varphi_i\wedge\varphi'_1\wedge..\wedge\varphi'_p\wedge\varphi_{i+1}\wedge..\wedge\varphi_n$$
$$\varphi_1\vee..\vee\varphi_i\vee(\bigvee_{j=1}^{p}\varphi'_j)\vee\varphi_{i+1}\vee..\vee\varphi_n \rightarrow \varphi_1\vee..\varphi_i\vee\varphi'_1\vee..\vee\varphi'_p\vee\varphi_{i+1}\vee..\vee\varphi_n$$
$$\varphi_1\vee..\vee\varphi_i\vee(\bigwedge_{j=1}^{p}\varphi'_j)\vee\varphi_{i+1}\vee..\vee\varphi_n \rightarrow \bigwedge_{j=1}^{p}(\varphi_1\vee..\vee\varphi_i\vee\varphi'_j\vee\varphi_{i+1}\vee..\vee\varphi_n)$$

Cette dernière transformation est notée fnc. Nous pouvons donc définir la transformation sous forme clausale :

$$clausale = fnc \circ prénexe \circ Sklm \circ linéaire$$

cette transformation permet de transformer toute formule de la logique du premier ordre en une formule sous *forme clausale*, c'est à dire de la forme $\forall x_1..x_n.\bigwedge_{i=1}^{p} C_i$, où les C_i sont des *clauses*, c'est à dire de la forme $\bigvee_{j=1}^{m} L_j^i$, où les L_j^i sont des *littéraux*, c'est à dire des formules atomiques ou des négations de formules atomiques. Cette transformation préserve la satisfaisabilité, et peut donc être utilisée dans une méthode procédant par réfutation. Nous allons dans la suite montrer qu'elle est exponentielle en taille, tout en développant un ensemble d'outils qui nous seront utiles par la suite.

[1] Il n'y a pas ici de problème de capture de variables, du fait de la restriction que nous avons adoptée page 8.

2.2 Complexités en taille

2.2.1 Linéarisation

Nous commençons par une étude de la première transformation : *linéaire*. La longueur de la formule linéarisée dépend essentiellement de la disposition des équivalences dans la formule initiale.

Définition. Le *nombre d'équivalence* de ψ, noté $N_{\Leftrightarrow}(\psi)$, est le nombre d'éléments du multi-ensemble $\{\varphi_1 \Leftrightarrow \varphi_2 / \varphi_1 \Leftrightarrow \varphi_2 \sqsubseteq \psi\}$.

On appelle *profondeur en équivalence*, et on note $D_{\Leftrightarrow}$, la fonction inductivement définie sur les formules par :

- si ψ est atomique, alors $D_{\Leftrightarrow}(\psi) = 0$.
- si ψ n'est pas une équivalence, et ses sous-formules directes sont $\psi_1..\psi_n$, alors $D_{\Leftrightarrow}(\psi) = \max\{D_{\Leftrightarrow}(\psi_i) / i \in \{1..n\}\}$.
- si $\psi = \psi_1 \Leftrightarrow \psi_2$, alors $D_{\Leftrightarrow}(\psi) = 1 + \max(D_{\Leftrightarrow}(\psi_1), D_{\Leftrightarrow}(\psi_2))$.

$\diamond$

Ces deux définitions permettent de donner une borne relativement fine pour la forme linéaire :

Théorème 2.1 $\forall \psi, |\mathit{linéaire}(\psi)| \leq (|\psi| + 2N_{\Leftrightarrow}(\psi))2^{D_{\Leftrightarrow}(\psi)}$

Preuve. Par induction structurelle sur ψ :

- si ψ est atomique, $\mathit{linéaire}(\psi) = \psi$, $D_{\Leftrightarrow}(\psi) = 0$ et $N_{\Leftrightarrow}(\psi) = 0$; l'inégalité est donc évidente.
- si $\psi = \neg\psi'$, on a

$$\begin{aligned} |\mathit{linéaire}(\psi)| &= 1 + |\mathit{linéaire}(\psi')| \\ &\leq 1 + (|\psi'| + 2N_{\Leftrightarrow}(\psi'))2^{D_{\Leftrightarrow}(\psi')} \text{ par hypothèse d'induction} \\ &\leq (1 + |\psi'| + 2N_{\Leftrightarrow}(\psi'))2^{D_{\Leftrightarrow}(\psi')} = (|\psi| + 2N_{\Leftrightarrow}(\psi))2^{D_{\Leftrightarrow}(\psi)} \end{aligned}$$

- si $\psi = Qx_1..x_n.\psi'$, où $Q \in \{\forall, \exists\}$, on a

$$\begin{aligned} |\mathit{linéaire}(\psi)| &= 1 + n + |\mathit{linéaire}(\psi')| \\ &\leq 1 + n + (|\psi'| + 2N_{\Leftrightarrow}(\psi'))2^{D_{\Leftrightarrow}(\psi')} \text{ hypothèse d'induction} \\ &\leq (1 + n + |\psi'| + 2N_{\Leftrightarrow}(\psi'))2^{D_{\Leftrightarrow}(\psi')} = (|\psi| + 2N_{\Leftrightarrow}(\psi))2^{D_{\Leftrightarrow}(\psi)} \end{aligned}$$

- si $\psi = \psi_1 \wedge .. \wedge \psi_n$ (ou $\psi_1 \vee .. \vee \psi_n$, ou $\psi_1 \Rightarrow \psi_2$), alors

$$\begin{aligned} |linéaire(\psi)| &= n - 1 + \sum_{i=1}^{n} |linéaire(\psi_i)| \\ &\leq n - 1 + \sum_{i=1}^{n} (|\psi_i| + 2N_{\Leftrightarrow}(\psi_i))2^{D_{\Leftrightarrow}(\psi_i)} \\ &\leq (n - 1 + \sum_{i=1}^{n} |\psi_i| + 2\sum_{i=1}^{n} N_{\Leftrightarrow}(\psi_i))2^{\max\{D_{\Leftrightarrow}(\psi_i)/i \in \{1..n\}\}} \\ &\leq (|\psi| + 2N_{\Leftrightarrow}(\psi))2^{D_{\Leftrightarrow}(\psi)} \end{aligned}$$

- si $\psi = \psi_1 \Leftrightarrow \psi_2$, alors

$$\begin{aligned} |linéaire(\psi)| &= 2|linéaire(\psi_1)| + 2|linéaire(\psi_2)| + 5 \\ &\leq (|\psi_1| + 2N_{\Leftrightarrow}(\psi_1))2^{1+D_{\Leftrightarrow}(\psi_1)} + (|\psi_2| + 2N_{\Leftrightarrow}(\psi_2))2^{1+D_{\Leftrightarrow}(\psi_2)} + 5 \\ &\leq (|\psi_1| + |\psi_2| + 2N_{\Leftrightarrow}(\psi_1) + 2N_{\Leftrightarrow}(\psi_2) + 3)2^{D_{\Leftrightarrow}(\psi)} \\ &\leq (|\psi| + 2N_{\Leftrightarrow}(\psi))2^{D_{\Leftrightarrow}(\psi)} \end{aligned}$$

L'induction est complète. ⊣

2.2.2 Skolemisation

Nous pouvons maintenant étudier le comportement de la skolemisation sur la longueur des formules. Par rapport aux autres transformations, la croissance en taille dépend de facteurs très différents, ce qui nécessite un ensemble de définitions relativement spécifiques.

Définition. Soit ψ une formule, à chaque $\varphi \sqsubseteq \psi$ on associe l'ensemble $SV_\psi(\varphi)$ des variables skolemisables dans ψ en φ, défini par :

1. $SV_\psi(\psi) = \emptyset$.
2. $SV_\psi(\varphi) = SV_\psi(\exists x_1..x_n.\varphi) \cup \{x_1..x_n\}$ si $pol(\varphi, \psi) \geq 0$.
3. $SV_\psi(\varphi) = SV_\psi(\forall x_1..x_n.\varphi) \cup \{x_1..x_n\}$ si $pol(\varphi, \psi) \leq 0$.
4. $SV_\psi(\varphi) = SV_\psi(\varphi')$ si φ est une sous-formule directe de φ', et que les cas précédents ne s'appliquent pas.

Si φ est atomique, nous notons $SO_\psi(\varphi)$ le nombre d'occurrences dans φ des variables de $SV_\psi(\varphi)$. Sinon, soient $\varphi_1..\varphi_n$ les sous-formules directes de φ, alors $SO_\psi(\varphi) = \sum_{i=1}^{n} SO_\psi(\varphi_i)$. On note $SO(\psi)$ pour $SO_\psi(\psi)$.

$V(\psi) = \max\{|FV(\varphi)|/\varphi \sqsubseteq \psi\}$, dénote donc le nombre maximum de variables libres dans les sous-formules de ψ.

Enfin, on associe à chaque $\varphi \sqsubseteq \psi$ une sous-formule de $Sklm(\psi)$, notée $Sklm_\psi(\varphi)$, de la façon suivante : si φ est atomique, $Sklm_\psi(\varphi)$ est la formule φ dans laquelle les variables de $SV_\psi(\varphi)$ ont été instanciées par les termes de Skolem correspondants ; si φ est $\exists x_1..x_n.\varphi'$ de polarité positive dans ψ, ou $\forall x_1..x_n.\varphi'$ de polarité négative dans ψ, alors $Sklm_\psi(\varphi) = Sklm_\psi(\varphi')$; et dans les autres cas, $Sklm_\psi(\varphi)$ est la sous-formule φ dans laquelle toutes les sous-formules directes ξ de φ ont été remplacées par $Sklm_\psi(\xi)$. On a donc $Sklm_\psi(\psi) = Sklm(\psi)$. $\diamond$

Ces outils nous permettent de majorer finement la longueur d'une formule skolemisée :

Théorème 2.2 *Si ψ est une formule linéaire, alors $|Sklm(\psi)| \leq |\psi| + V(\psi)SO(\psi)$.*

Preuve. Par induction structurelle sur φ, nous montrons que $\forall \varphi \sqsubseteq \psi, |Sklm_\psi(\varphi)| \leq |\varphi| + V(\psi)SO_\psi(\varphi)$.

- si φ est une sous-formule atomique de ψ, alors $Sklm_\psi(\varphi)$ est obtenue à partir de φ en y substituant toutes les occurrences des variables de $SV_\psi(\varphi)$ par des termes de skolem, dont la longueur est majorée par $V(\psi) + 1$, donc $|Sklm_\psi(\varphi)| - |\varphi| \leq (V(\psi) + 1)SO_\psi(\varphi) - SO_\psi(\varphi) = V(\psi)SO_\psi(\varphi)$, et ce cas est terminé.

- si φ est une conjonction $\varphi_1 \wedge .. \wedge \varphi_n$ (ou une disjonction, ou une implication), alors

$$\begin{aligned} |Sklm_\psi(\varphi)| &= n - 1 + \sum_{i=1}^{n} |Sklm_\psi(\varphi_i)| \\ &\leq n - 1 + \sum_{i=1}^{n} |\varphi_i| + V(\psi) \sum_{i=1}^{n} SO_\psi(\varphi_i) \text{ hypothèse d'induction} \\ &\leq |\varphi| + V(\psi)SO_\psi(\varphi) \end{aligned}$$

- si φ est une négation $\neg\varphi'$, alors $|Sklm_\psi(\varphi)| = 1 + |Sklm_\psi(\varphi')| \leq 1 + |\varphi'| + V(\psi)SO_\psi(\varphi') = |\varphi| + V(\psi)SO_\psi(\varphi)$.

- enfin, si φ est une formule quantifiée $Qx_1..x_n.\varphi'$, on distingue deux cas :

  1. si $x_1..x_n \in SV_\psi(\varphi')$, alors $|Sklm_\psi(\varphi)| = |Sklm_\psi(\varphi')|$
  2. sinon $|Sklm_\psi(\varphi)| = n + 1 + |Sklm_\psi(\varphi')|$

Table 2.1: Nombre de clauses

ψ	$p(\psi)$	$\overline{p}(\psi)$
$\psi_1 \wedge .. \wedge \psi_n$	$\sum_{i=1}^{n} p(\psi_i)$	$\prod_{i=1}^{n} \overline{p}(\psi_i)$
$\psi_1 \vee .. \vee \psi_n$	$\prod_{i=1}^{n} p(\psi_i)$	$\sum_{i=1}^{n} \overline{p}(\psi_i)$
$\psi_1 \Rightarrow \psi_2$	$\overline{p}(\psi_1)p(\psi_2)$	$p(\psi_1) + \overline{p}(\psi_2)$
$\psi_1 \Leftrightarrow \psi_2$	$p(\psi_1)\overline{p}(\psi_2) + \overline{p}(\psi_1)p(\psi_2)$	$p(\psi_1)p(\psi_2) + \overline{p}(\psi_1)\overline{p}(\psi_2)$
$Qx_1..x_n.\psi'$	$p(\psi')$	$\overline{p}(\psi')$
$\neg\psi'$	$\overline{p}(\psi')$	$p(\psi')$
atomique	1	1

dans les deux cas, on a $|Sklm_\psi(\varphi)| \leq n + 1 + |Sklm_\psi(\varphi')| \leq n + 1 + |\varphi'| + V(\psi)SO_\psi(\varphi') = |\varphi| + V(\psi)SO_\psi(\varphi)$.

L'induction est terminée, et on peut en déduire, pour $\varphi = \psi$, que $|Sklm(\psi)| = |Sklm_\psi(\psi)| \leq |\psi| + V(\psi)SO(\psi)$ $\dashv$

La transformation *prénexe* ne pose guère de problème de complexité en taille : il est évident qu'on a $\forall\psi, |prénexe(\psi)| \leq |\psi|$.

2.2.3 Nombre de clauses

Nous allons maintenant étudier la mise sous forme clausale en distinguant d'une part le nombre de clauses, de l'autre la longueur de ces clauses.

Définition. On note $p(\psi)$ le nombre de clauses de $clausale(\psi)$, et $\overline{p}(\psi) = p(\neg\psi)$. $\diamond$

On peut calculer le nombre de clauses d'une formule sans avoir à calculer la forme clausale elle-même, en utilisant le tableau 2.1, ce qui se montre aisément : si $fnc(\psi_i) = \bigwedge_{j=1}^{n_i} C_j^i$, on a clairement $p(\psi_1 \wedge \psi_2) = p(\psi_1) + p(\psi_2)$. Dans le cas $\psi_1 \vee \psi_2$, on applique $1 + n_1$ fois la règle de distributivité (ou $1 + n_2$ fois) :

$$\bigwedge_{j=1}^{n_1} C_j^1 \vee \bigwedge_{k=1}^{n_2} C_k^2 \rightarrow \bigwedge_{j=1}^{n_1} (C_j^1 \vee \bigwedge_{k=1}^{n_2} C_k^2) \xrightarrow{*} \bigwedge_{j=1}^{n_1} \bigwedge_{k=1}^{n_2} C_j^1 \vee C_k^2$$

On a donc $p(\psi_1 \vee \psi_2) = n_1 n_2 = p(\psi_1)p(\psi_2)$. Les autres cas se déduisent de ceux-là.

En général, la transformation sous forme clausale ne se limite pas à *clausale* : on a tout intérêt à tenter des simplifications. Mais le tableau 2.1 serait alors très

Table 2.2: coefficients

φ	$a^{\psi}_{\varphi_i}$	$b^{\psi}_{\varphi_i}$
$\varphi_1 \wedge .. \wedge \varphi_n$	a^{ψ}_{φ}	$b^{\psi}_{\varphi} \prod_{j \neq i} \overline{p}(\varphi_j)$
$\varphi_1 \vee .. \vee \varphi_n$	$a^{\psi}_{\varphi} \prod_{j \neq i} p(\varphi_j)$	b^{ψ}_{φ}
$\varphi_1 \Rightarrow \varphi_2, i = 1$	b^{ψ}_{φ}	$a^{\psi}_{\varphi} p(\varphi_2)$
$\varphi_1 \Rightarrow \varphi_2, i = 2$	$a^{\psi}_{\varphi} \overline{p}(\varphi_1)$	b^{ψ}_{φ}
$\varphi_1 \Leftrightarrow \varphi_2, j = 3 - i$	$a^{\psi}_{\varphi} \overline{p}(\varphi_j) + b^{\psi}_{\varphi} p(\varphi_j)$	$a^{\psi}_{\varphi} p(\varphi_j) + b^{\psi}_{\varphi} \overline{p}(\varphi_j)$
$\neg \varphi_i$	b^{ψ}_{φ}	a^{ψ}_{φ}
$Q x_1 .. x_n . \varphi_i$	a^{ψ}_{φ}	b^{ψ}_{φ}
$\varphi_i = \psi$	1	0

différent. Ne pas effectuer de simplification induit la propriété très importante que le nombre de clauses d'une formule ne dépend *que* du nombre de clauses de ses sous-formules. C'est bien cette propriété qui est exprimée lorsque nous disons que le nombre de clauses peut se calculer *directement* à partir de la formule. On peut traduire cette propriété par la définition suivante :

Définition. $\forall \psi, \forall \varphi \sqsubseteq \psi$, on note P^{ψ}_{φ} (resp. $\overline{P}^{\psi}_{\varphi}$) la fonction de $I\!N^* \times I\!N^*$ dans $I\!N^*$ définie par $\forall \varphi', P^{\psi}_{\varphi}(p(\varphi'), \overline{p}(\varphi')) = p(\psi[\varphi \leftarrow \varphi'])$ (resp. $\overline{P}^{\psi}_{\varphi}(p(\varphi'), \overline{p}(\varphi')) = \overline{p}(\psi[\varphi \leftarrow \varphi'])$) $\diamond$

Il est clair que si cette propriété n'était pas vérifiée, P^{ψ}_{φ} ne serait pas une fonction. Cette définition nous est utile du fait que le renommage consiste bien à remplacer une sous-formule par une autre, atomique. Ainsi, $p(Rnm(\varphi, \psi)) = P^{\psi}_{\varphi}(1,1) + p(Def_{\psi}(\varphi))$. Nous utiliserons également l'équation $P^{\psi}_{\varphi}(x,y) = P^{\psi}_{\varphi'}(P^{\varphi'}_{\varphi}(x,y), \overline{P}^{\varphi'}_{\varphi}(x,y))$ où $\varphi \sqsubseteq \varphi' \sqsubseteq \psi$, dont la justification est triviale et qui justifie la définition de $\overline{P}^{\psi}_{\varphi}$ ci-dessus.

Exemple : soit $\psi = (\varphi \Rightarrow \varphi_1) \wedge \varphi_2$, on a $p(\psi) = \overline{p}(\varphi) p(\varphi_1) + p(\varphi_2)$ et $\overline{p}(\psi) = (p(\varphi) + \overline{p}(\varphi_1)) \overline{p}(\varphi_2)$, donc $P^{\psi}_{\varphi}(x,y) = p(\varphi_1)y + p(\varphi_2)$ et $\overline{P}^{\psi}_{\varphi}(x,y) = \overline{p}(\varphi_2)x + \overline{p}(\varphi_2)\overline{p}(\varphi_1)$.

Il reste bien entendu à calculer $P^{\psi}_{\varphi}(x,y)$, donc en définitive à calculer la fonction P^{ψ}_{φ} elle-même, à partir de ψ et φ. D'après le tableau 2.1, il est facile de voir que P^{ψ}_{φ} est un polynôme à coefficients entiers, et même :

Théorème 2.3 $\forall \psi, \forall \varphi \sqsubseteq \psi, \exists abc \in I\!N, \forall xy \in I\!N^*, P^{\psi}_{\varphi}(x,y) = ax + by + c$

Preuve. Par récurrence sur $d_\psi(\varphi)$. Si $d_\psi(\varphi) = 0$, alors $\psi = \varphi$, et $P^\psi_\varphi(x,y) = x$, donc $a = 1$ et $b = c = 0$. Supposons la propriété vérifiée si $d_\psi(\varphi) = n$, et considérons deux formules $\varphi \sqsubseteq \psi$ telles que $d_\psi(\varphi) = n+1$. Soit φ' la plus petite sur-formule de φ dans ψ, on a $d_\psi(\varphi') = n$, donc $\exists abc \in I\!N, \forall xy \in I\!N^*, P^\psi_{\varphi'}(x,y) = ax + by + c$. On a donc $P^\psi_\varphi(x,y) = P^\psi_{\varphi'}(P^{\varphi'}_\varphi(x,y), \overline{P}^{\varphi'}_\varphi(x,y)) = aP^{\varphi'}_\varphi(x,y) + b\overline{P}^{\varphi'}_\varphi(x,y) + c$. Il suffit donc de prouver que $P^{\varphi'}_\varphi(x,y)$ et $\overline{P}^{\varphi'}_\varphi(x,y)$ sont des polynômes de degré 1 en x, y. Or ceci est évident du fait que dans le tableau 2.1, chaque ligne est un polynôme de degré 1 en $p(\psi_i), \overline{p}(\psi_i)$. $\dashv$

Il faut noter ici que ce théorème n'est vrai que si l'on applique la linéarisation dépendante de la polarité. Dans le cas contraire, le degré de ces polynômes n'est pas borné — ce qui complique énormément les choses, voir [BdlT89]. Ensuite, on remarque que les coefficients de x et y dans $P^\psi_\varphi(x,y)$ ne dépendent pas de c. Si on note a^ψ_φ (resp. b^ψ_φ) le coefficient de x (resp. de y) dans $P^\psi_\varphi(x,y)$, ceux-ci peuvent donc être calculés à partir de $a^\psi_{\varphi'}$ et $b^\psi_{\varphi'}$, où φ' est comme dans la preuve ci-dessus la plus petite sur-formule de φ dans ψ. Les formules correspondantes sont données dans le tableau 2.2. On ne donnera que la justification pour $\varphi = \varphi_1 \Leftrightarrow \varphi_2, i = 1$:

$$\begin{aligned} P^\psi_{\varphi_1}(x,y) &= P^\psi_\varphi(P^\varphi_{\varphi_1}(x,y), \overline{P}^\varphi_{\varphi_1}(x,y)) \\ &= a^\psi_\varphi(\overline{p}(\varphi_2)x + p(\varphi_2)y) + b^\psi_\varphi(p(\varphi_2)x + \overline{p}(\varphi_2)y) + c \\ &= (a^\psi_\varphi\overline{p}(\varphi_2) + b^\psi_\varphi p(\varphi_2))x + (a^\psi_\varphi p(\varphi_2) + b^\psi_\varphi\overline{p}(\varphi_2))y + c \end{aligned}$$

Ces formules nous seront utile d'un point de vue algorithmique, au chapitre 6. Pour l'heure, nous revenons au tableau 2.1 pour constater le caractère exponentiel de la mise sous forme normale conjonctive : soit ψ_n la formule $\bigvee_{i=1}^n (P_i \wedge Q_i)$, on a $|\psi_n| = 4n-1$, et $p(\psi_n) = 2^n$, donc $|fnc(\psi_n)| \geq 2^n$. La complexité en taille, de même qu'en nombre de clauses, de cette transformation est donc au moins exponentielle dans le pire des cas. On montre d'abord qu'en ce qui concerne le nombre de clauses, cette complexité est exactement exponentielle :

Théorème 2.4 $\forall \psi, \max(p(\psi), \overline{p}(\psi)) \leq 2^{|SF(\psi)|}$

Preuve. Par induction structurelle sur ψ :

- si ψ est une formule atomique, on a $|SF(\psi)| = 1$, et d'après le tableau 2.1, $p(\psi) = \overline{p}(\psi) = 1$, l'inégalité est donc trivialement vérifiée.

- si $\psi = \neg\psi'$ (ou si $\psi = Qx_1..x_n.\psi'$), on a :

$$\begin{aligned}\max(p(\psi),\overline{p}(\psi)) &= \max(p(\psi'),\overline{p}(\psi')) \\ &\leq 2^{|SF(\psi')|} \text{ par hypothèse d'induction} \\ &\leq 2^{|SF(\psi)|} \text{ car } |SF(\psi')| \leq |SF(\psi)|\end{aligned}$$

- si $\psi = \psi_1 \wedge .. \wedge \psi_n$ (de même si $\psi = \psi_1 \vee .. \vee \psi_n$ ou $\psi = \psi_1 \Rightarrow \psi_2$), on a :

$$\begin{aligned}p(\psi) &= \sum_{i=1}^{n} p(\psi_i) \leq \sum_{i=1}^{n} 2^{|SF(\psi_i)|} \text{ par hypothèse d'induction} \\ &\leq \prod_{i=1}^{n} 2^{|SF(\psi_i)|} \leq 2^{\sum_{i=1}^{n}|SF(\psi_i)|} \leq 2^{|SF(\psi)|}\end{aligned}$$

et de même $\overline{p}(\psi) = \prod_{i=1}^{n} \overline{p}(\psi_i) \leq 2^{|SF(\psi)|}$

- si $\psi = \psi_1 \Leftrightarrow \psi_2$, on a :

$$\begin{aligned}\max(p(\psi),\overline{p}(\psi)) &= \max(p(\psi_1)\overline{p}(\psi_2) + \overline{p}(\psi_1)p(\psi_2), p(\psi_1)p(\psi_2) + \overline{p}(\psi_1)\overline{p}(\psi_2)) \\ &\leq 2^{|SF(\psi_1)|+|SF(\psi_2)|} + 2^{|SF(\psi_1)|+|SF(\psi_2)|} \text{ par h.i.} \\ &\leq 2^{|SF(\psi_1)|+|SF(\psi_2)|+1} \\ &\leq 2^{|SF(\psi)|}\end{aligned}$$

L'induction est complète. ⊣

2.2.4 Longueur des clauses

Quant à la longueur même de la forme clausale, dont on sait qu'elle est au moins exponentielle, nous allons en obtenir une borne en majorant la *longueur* des clauses.

Définition. Soient $C_i, i = 1..n$ les clauses de $fnc(prénexe(linéaire(\psi)))$, et $L^i_j, j = 1..p_i$ les littéraux de C_i, on note $h(\psi) = \max_{i=1}^{n} \sum_{j=1}^{p_i} |L^i_j|$, et $\overline{h}(\psi) = h(\neg\psi)$ ◇

$h(\psi)$ n'est pas exactement la longueur maximale des clauses : on ne compte pas les disjonctions. Mais comme il n'y a pas plus de symboles de disjonction que de littéraux, la longueur maximale des clauses est bornée par $2h(\psi)$. L'avantage est que la fonction h est très facile à calculer à partir de la formule ψ. Il suffit pour cela d'appliquer les règles du tableau 2.3, qui s'établissent très facilement : si $fnc(\psi_i) = \bigwedge_{j=1}^{n_i} C^i_j$, on a évidemment $h(\psi_1 \wedge \psi_2) = \max(h(\psi_1), h(\psi_2))$, et comme

Table 2.3: Longueur maximale des clauses

ψ	$h(\psi)$	$\overline{h}(\psi)$
$\psi_1 \wedge .. \wedge \psi_n$	$\max_{i=1}^{n} h(\psi_i)$	$\sum_{i=1}^{n} \overline{h}(\psi_i)$
$\psi_1 \vee .. \vee \psi_n$	$\sum_{i=1}^{n} h(\psi_i)$	$\max_{i=1}^{n} \overline{h}(\psi_i)$
$\psi_1 \Rightarrow \psi_2$	$\overline{h}(\psi_1) + h(\psi_2)$	$\max(h(\psi_1), \overline{h}(\psi_2))$
$\psi_1 \Leftrightarrow \psi_2$	$\max(h(\psi_1) + \overline{h}(\psi_2), \overline{h}(\psi_1) + h(\psi_2))$	$\max(h(\psi_1) + h(\psi_2), \overline{h}(\psi_1) + \overline{h}(\psi_2))$
$Qx_1..x_n.\psi'$	$h(\psi')$	$\overline{h}(\psi')$
$\neg\psi'$	$\overline{h}(\psi')$	$h(\psi')$
littéral	$\lvert\psi\rvert$	$\lvert\psi^c\rvert$

$fnc(\psi_1 \vee \psi_2) = \bigwedge_{j=1}^{n_1} \bigwedge_{k=1}^{n_2} C_j^1 \vee C_k^2$, et qu'on sait que $\exists j \in \{1..n_1\}/h(C_j^1) = h(\psi_1), \exists k \in \{1..n_2\}/h(C_k^2) = h(\psi_2)$, on a $h(C_j^1 \vee C_k^2) = h(\psi_1) + h(\psi_2)$, et bien sûr $\forall j', k', h(C_{j'}^1 \vee C_{k'}^2) \leq h(C_{j'}^1) + h(C_{k'}^2) \leq h(\psi_1) + h(\psi_2)$, donc $h(\psi_1 \vee \psi_2) = h(\psi_1) + h(\psi_2)$. On voit ici l'intérêt de ne pas "compter" dans h le nombre de symboles de disjonctions dans les clauses.

De même si nous avions tenu compte de la skolemisation il n'aurait pas été possible de calculer $h(\psi)$ uniquement en fonction des $h(\psi_i)$ (ou $\overline{h}(\psi_i)$), car la longueur après skolemisation ne dépend plus seulement des longueurs après skolemisation des sous-formules, mais également du nombre d'occurrence des variables skolemisées. Nous nous contentons donc d'une formule prénexe dont la matrice est sous forme normale conjonctive; le préfixe n'est pas nécessairement universel. On peut alors borner la fonction h par la longueur de la formule initiale :

Théorème 2.5 $\forall \psi, h(\psi) \leq |\psi| \wedge \overline{h}(\psi) \leq |\psi^c|$

Preuve. Par induction structurelle sur ψ.

- si ψ est atomique, les inégalités sont triviales d'après le tableau 2.3.
- si $\psi = \neg\psi'$, on a

$$\begin{aligned} h(\psi) = \overline{h}(\psi') &\leq |\psi'^c| \text{ (par hypothèse d'induction)} \\ &\leq |\psi'| + 1 = |\psi| \\ \overline{h}(\psi) = h(\psi') &\leq |\psi'| = |\psi^c| \end{aligned}$$

- si $\psi = Qx_1..x_n.\psi'$, alors

$$\begin{aligned} h(\psi) &= h(\psi') \leq |\psi'| \leq |\psi| \\ \overline{h}(\psi) &= \overline{h}(\psi') \leq |\psi'^c| \leq |\psi^c| \end{aligned}$$

- si $\psi = \psi_1 \wedge .. \wedge \psi_n$, on a

$$\begin{aligned} h(\psi) &= \max_{i=1}^{n} h(\psi_i) \leq \max_{i=1}^{n} |\psi_i| \leq |\psi| \\ \overline{h}(\psi) &= \sum_{i=1}^{n} \overline{h}(\psi_i) \leq \sum_{i=1}^{n} |\psi_i^c| \leq n + \sum_{i=1}^{n} |\psi_i| \leq |\psi| + 1 = |\psi^c| \end{aligned}$$

- si $\psi = \psi_1 \vee .. \vee \psi_n$, de même

$$\begin{aligned} h(\psi) &= \sum_{i=1}^{n} h(\psi_i) \leq \sum_{i=1}^{n} |\psi_i| \leq |\psi| \\ \overline{h}(\psi) &= \max_{i=1}^{n} \overline{h}(\psi_i) \leq \max_{i=1}^{n} |\psi_i^c| \leq 1 + \max_{i=1}^{n} |\psi_i| \leq 1 + |\psi| = |\psi^c| \end{aligned}$$

- si $\psi = \psi_1 \Rightarrow \psi_2$, alors

$$\begin{aligned} h(\psi) &= \overline{h}(\psi_1) + h(\psi_2) \leq |\psi_1^c| + |\psi_2| \leq 1 + |\psi_1| + |\psi_2| = |\psi| \\ \overline{h}(\psi) &= \max(h(\psi_1), \overline{h}(\psi_2)) \leq 1 + |\psi_1| + |\psi_2| \leq |\psi^c| \end{aligned}$$

- si $\psi = \psi_1 \Leftrightarrow \psi_2$,

$$\begin{aligned} h(\psi) = \max(h(\psi_1) + \overline{h}(\psi_2), \overline{h}(\psi_1) + h(\psi_2)) &\leq \max(|\psi_1| + |\psi_2^c|, |\psi_1^c| + |\psi_2|) \\ &\leq 1 + |\psi_1| + |\psi_2| = |\psi| \\ \overline{h}(\psi) = \max(h(\psi_1) + h(\psi_2), \overline{h}(\psi_1) + \overline{h}(\psi_2)) &\leq \max(|\psi_1| + |\psi_2|, |\psi_1^c| + |\psi_2^c|) \\ &\leq 2 + |\psi_1| + |\psi_2| = |\psi^c| \end{aligned}$$

L'induction est complète. ⊣

2.2.5 La forme clausale

Pour conclure ce chapitre, nous pouvons évaluer la complexité en taille de la transformation *clausale* :

Théorème 2.6 $\forall \psi, |clausale(\psi)|$ *est bornée par* $O((1 + V(\psi))|\psi|2^{|SF(\psi)|})$.

Preuve. Soient $\psi' = linéaire(\psi)$ et $\psi'' = Sklm(\psi')$. On a $|clausale(\psi)| \leq 2p(\psi)h(\psi'') + 1 + V(prénexe(\psi''))$, le dernier terme correspondant à la longueur du préfixe. Il est facile de voir que $h(\psi'') \leq (1 + V(\psi'))h(\psi')$, car chaque clause C de $fnc(prénexe(\psi''))$ correspond à une clause C' de $fnc(prénexe(\psi'))$, dans laquelle certaines variables ont pu être remplacées par des termes de longueur au plus $1 + V(\psi')$, donc $h(C) \leq (1 + V(\psi'))h(C')$.

On a évidemment $h(\psi') = h(\psi) \leq |\psi|$ (théorème 2.5) et $V(\psi') = V(\psi)$, on obtient donc en utilisant également le théorème 2.4 : $|\mathit{clausale}(\psi)| \leq 2(1 + V(\psi))|\psi|2^{|SF(\psi)|} + 1 + V(\mathit{prénexe}(\psi''))$.

Par ailleurs, on a $V(\mathit{prénexe}(\psi'')) \leq |\mathit{prénexe}(\psi'')| \leq |\psi''| \leq |\psi'| + V(\psi')SO(\psi') \leq (1 + V(\psi'))|\psi'| \leq (1 + V(\psi))(|\psi| + 2N_{\Leftrightarrow}(\psi))2^{D_{\Leftrightarrow}(\psi)} \leq (1 + V(\psi))|\psi|2^{1+D_{\Leftrightarrow}(\psi)}$, car $2N_{\Leftrightarrow}(\psi) \leq |\psi|$. On a évidemment $1 + D_{\Leftrightarrow}(\psi) \leq |SF(\psi)|$, et le préfixe est donc également borné en $O((1 + V(\psi))|\psi|2^{|SF(\psi)|})$ ⊣

Dans le cas du calcul propositionnel, cette borne se ramène à $O(|\psi|2^{|\psi|})$. En reprenant l'exemple $\psi_n = \bigvee_{i=1}^{n}(P_i \wedge Q_i)$, où $|\psi_n| = 4n-1$ et $p(\psi_n) = 2^n$, le nombre de littéraux dans $fnc(\psi_n)$ est $p(\psi_n)\sum_{i=1}^{n}\frac{2}{2} = n2^n$. La complexité en taille de *clausale* dans le pire des cas, dans le calcul propositionnel, est donc exactement $O(|\psi|2^{|\psi|})$.

En logique du premier ordre, cette borne est intéressante dans la mesure où elle montre que la longueur des termes n'intervient pas dans le comportement exponentiel de la transformation : c'est le nombre de clauses qui en est responsable.

Chapitre 3

Forme clausale et renommages

3.1 Introduction

Et sous les arbres pleins d'une gente musique,
Notre entretien était souvent métaphysique.

Verlaine

L'utilisation de la technique du renommage pour la mise sous forme clausale est apparue dans [Tse68], mais restreinte au calcul propositionnel, la transformation présentée étant approximativement $fnc(Rnm(SF(\psi), \psi))$. Sa principale propriété est d'être *linéaire* en taille, ce qui constitue une importante amélioration de la transformation fnc, qui est exponentielle en taille, du moins dans le pire des cas. Plus tard sont apparues des généralisations à la logique du premier ordre, en particulier dans [Ede84] et [PG86]. La transformation subit cependant deux modifications importantes.

La première concerne la façon dont les définitions sont formulées : dans [Tse68] et [Ede84], elles le sont exclusivement avec des équivalences. Ce n'est que dans

[PG86] qu'apparaît la simplification consistant à ne conserver qu'une implication pour les sous-formules de polarité non nulle. Elle est cependant déjà mentionnée dans [Ede84]. La deuxième consiste à *restreindre* le renommage. Dans [Tse68], on renomme absolument toutes les sous-formules. Dans [Ede84], les formules atomiques ne sont pas renommées. La restriction s'étend aux négations dans [PG86], dans lequel la transformation prend le nom de "structure preserving".

Définition. Nous noterons $struct\_pres(\psi)$ le multi-ensemble $SF(\psi)$ soustrait des formules atomiques et des négations. $\diamond$

Ces modifications sont incontestablement des améliorations de la transformation initiale : tout en conservant la complexité en taille, les formes clausales obtenues sont plus simples, c'est à dire essentiellement plus facile à réfuter. Le but n'est évidemment pas *uniquement* d'obtenir une forme clausale linéaire (ou quadratique dans le premier ordre), encore faut-il que celle-ci ne soit pas plus difficile à réfuter que la forme clausale standard. Or il n'est pas du tout évident que ce soit le cas.

Une première raison tient tout simplement au fait que malgré le mauvais comportement asymptotique de la transformation standard *clausale*, il n'est pas rare que $clausale(\psi)$ soit plus concise, plus simple que $clausale \circ Rnm(struct\_pres(\psi), \psi)$. C'est en particulier vrai si ψ est déjà une formule clausale, auquel cas $struct\_pres(\psi)$ contient chaque clause (chaque conjoint) de ψ, et la forme clausale résultante est donc plus longue que ψ.

Mais la raison principale est qu'il n'existe pas de relation (connue) entre la longueur d'une forme clausale insatisfaisable et celle de sa réfutation : une formule clausale relativement petite peut être beaucoup plus difficile à réfuter qu'une autre, plus longue. Cela dépend également de la stratégie utilisée. Or, dans le cas qui nous occupe, il est possible d'établir une comparaison qui ne souffre aucune exception : si une stratégie d'ordonnancement privilégiant les littéraux de Skolem est utilisée pour réfuter $clausale \circ Rnm(R, \psi)$, alors la réfutation est plus longue que celle de $clausale(\psi)$ avec la même stratégie.

Un tel résultat n'est absolument pas difficile à obtenir : il suffit de constater qu'une telle stratégie, travaillant à éliminer les littéraux de Skolem, va produire toutes les clauses de $clausale(\psi)$. Dans le cas du renommage *struct_pres*, ces littéraux permettaient d'éviter toute application de la loi de distributivité (cause du comportement exponentiel de *clausale*). En résolvant d'abord sur les littéraux de Skolem, c'est la résolution qui produit, clause par clause, ce qu'aurait produit l'application de la loi de distributivité s'il n'y avait pas eu renommage.

Ceci ne constitue évidemment pas une raison suffisante pour invalider le renommage ; nous en concluons simplement qu'il convient de résoudre le plus tard possible sur les littéraux de Skolem. C'est ce qui est fait dans [PG86], grâce à la "lock-resolution", avec des indices tels que les littéraux de Skolem sont résolus d'autant plus tard qu'ils correspondent à des sous-formules plus grandes.

Ce faisant, il n'est plus de certitude quant aux difficultés relatives de réfutation. Si le comportement linéaire de la transformation structure preserving est certainement intéressant, du moins intuitivement, on aurait bien du mal à en *démontrer* l'intérêt d'une façon générale. De même que toute stratégie de résolution connue. C'est en définitive sur des expérimentations que repose l'évidence d'un tel intérêt.

Les expérimentations présentées dans [PG86] montrent que le renommage *struct_pres* permet parfois de réduire considérablement la durée de réfutation, mais également qu'il peut l'accroître, ce qui montre que ce renommage est perfectible, et que l'excellente complexité en taille de cette transformation ne constitue pas un critère déterminant d'un "bon" renommage. Il semble cependant que la concision de la forme clausale soit un facteur très important d'efficacité des réfutations, et qu'il est donc essentiel d'éviter le comportement exponentiel de la transformation *clausale*. Ce critère étant vérifié par le renommage *struct_pres*, il ne permet pas d'en expliquer les échecs. Ceux-ci semblent plus justement imputables à la confusion qui est faite entre les formules dont la forme clausale est effectivement exponentielle et les autres ; il n'est certainement pas pertinent de renommer une formule dont la forme clausale est déjà concise. En particulier, les formules clausales elle-même ne doivent pas être renommées, et toute mise sous forme clausale (ou sous une forme quelconque) doit être projective, ce qui n'est pas le cas de la transformation structure preserving.

D'un point de vue plus général (et plus proche des préoccupations de l'Intelligence Artificielle), il nous semble qu'une bonne transformation par renommage doit elle-même être projective. Nous aimerions en effet atteindre, par renommage, une formulation simple (quoique cela puisse signifier) d'un problème, et donc disposer d'une transformation projective vers la classe des formules simples. Dans notre cas, cette notion de simplicité est, au minimum, relative à la complexité de *clausale* ; la restriction de *clausale* à la classe des formules simples doit être polynômiale.

L'algorithme de renommage lui-même doit évidemment répondre à des critères d'efficacité . Si nous n'en imposions pas, la réponse aux problèmes précédents serait

triviale, et débile ; il suffirait en effet de calculer tous les renommages possibles et de réfuter toutes les formules clausales obtenues pour connaître la plus courte. Ce n'est donc que sous une condition d'efficacité que les questions précédentes prennent un sens. Nous réclamerons donc d'un bon renommage qu'il soit calculable en temps polynômial de la taille de la formule.

3.2 Le renommage *struct_pres*

Nous avons déjà, en introduction, défini le renommage ***struct_pres*** et énoncé la propriété qui en fait l'intérêt, à savoir que ce renommage permet d'obtenir une transformation sous forme clausale dont la complexité en taille est linéaire dans le calcul propositionnel, quadratique dans la logique du premier ordre. On peut trouver une preuve de ce résultat dans [PG86], quoique fort succincte, et basée sur le fait que le champ des quantificateurs existentiels est renommé, suggérant donc que le renommage de ces sous-formules est nécessaire pour obtenir une telle complexité, ce qui est faux (il suffit en effet d'utiliser le point 3 du lemme 3.1 ci-dessous).

Mais s'il nous faut ici redémontrer ce résultat, ce n'est pas uniquement pour en améliorer la preuve, c'est également parce que notre formalisme est différent ; nous avons en effet adopté une syntaxe autorisant l'emploi des conjonctions et disjonctions avec un nombre quelconque d'arguments, alors que dans [PG86], celui-ci est fixé à deux ; en conséquence, le nombre de littéraux par clause y est borné par une constante, ce qui n'est pas vrai pour $clausale(Rnm(struct\_pres(\psi), \psi))$. Notre formalisme étant donc plus général (également par les quantificateurs, ce qui est moins important ici), il nous faut étendre cette preuve, et nous en profitons pour en donner une formulation complète, ce qui nécessite quelques résultats préliminaires :

Lemme 3.1 *Pour toute formule ψ et tout renommage R de ψ, on a :*

1. $V(Rnm(R,\psi)) = V(\psi)$
2. $SV_\psi(\varphi) = \emptyset \Rightarrow SO_\psi(\varphi) = SO(\varphi)$
3. $SO(Rnm(R,\psi)) \leq SO(\psi)$
4. $|SF(\psi[R])| + \sum_{\varphi \in R} |SF(\varphi[R])| = |SF(\psi)| + |R|$
5. $|\psi[R]| + \sum_{\varphi \in R} |\varphi[R]| \leq |\psi| + (1 + V(\psi))|R|$
6. $|Rnm(R,\psi)| \leq |\psi| + (5 + 3V(\psi))|R|$

Preuve.

1. trivial : on ne modifie pas les variables libres de ψ par renommage, et les nouvelles sous-formules ont les mêmes variables libres que les sous-formules renommées.

2. évident : pour toute sous-formule atomique ξ de ψ, on a $SV_\varphi(\xi) = SV_\psi(\xi)$ d'après la définition, donc $SO_\varphi(\xi) = SO_\psi(\xi)$. On obtient donc par induction $SO_\psi(\varphi) = SO_\varphi(\varphi)$.

3. se montre facilement à partir de $\forall \varphi \sqsubseteq \psi, SO(Rnm(\varphi,\psi)) \leq SO(\psi)$, qui s'établit comme suit : on a $SO(Rnm(\varphi,\psi)) = SO(\psi[\varphi]) + SO(Def_\psi(\varphi))$. Pour chaque sous-formule atomique ξ de ψ telle que $\xi \not\sqsubseteq \varphi$, on a $SO_{\psi[\varphi]}(\xi) = SO_\psi(\xi)$, et on peut donc éliminer ces termes de chaque côté de l'inégalité à démontrer. On obtient alors l'inégalité $SO_{\psi[\varphi]}(SkL^\psi_\varphi) + SO(Def_\psi(\varphi)) \leq SO_\psi(\varphi)$, qui reste à montrer.

   Pour toute sous-formule atomique $\xi \sqsubseteq Def_\psi(\varphi)$, soit $\xi = SkL^\psi_\varphi$, et $SO_{Def_\psi(\varphi)}(\xi) = 0$, soit $\xi \sqsubseteq \varphi$, et on a $SV_{Def_\psi(\varphi)}(\xi) = SV_\psi(\xi) - SV_\psi(\varphi)$, car les variables libres de φ sont quantifiées universellement dans $Def_\psi(\varphi)$, et n'y sont donc pas skolemisables. On en déduit que $SO_{Def_\psi(\varphi)}(\xi) = SO_\psi(\xi) - n_\xi$, où n_ξ est le nombre d'occurrences dans ξ des variables de $SV_\psi(\varphi)$. Donc

$$SO(Def_\psi(\varphi)) = \sum_{\substack{\xi \sqsubseteq \varphi \\ \xi \text{ atomique}}} (SO_\psi(\xi) - n_\xi) = SO_\psi(\varphi) - \sum_{\substack{\xi \sqsubseteq \varphi \\ \xi \text{ atomique}}} n_\xi$$

   Il reste donc à prouver que $SO_{\psi[\varphi]}(SkL^\psi_\varphi) \leq \sum_{\substack{\xi \sqsubseteq \varphi \\ \xi \text{ atomique}}} n_\xi$

   ce qui est évident puisqu'il n'y a dans SkL^ψ_φ qu'une seule occurrence de chacune des variables de $SV_\psi(\varphi)$ qui apparaît effectivement dans φ (i.e. qui est libre dans φ).

4. se montre facilement à partir de $\forall \varphi \sqsubseteq \psi, |SF(\psi[\varphi])| + |SF(\varphi)| = 1 + |SF(\psi)|$, ce qui est évident puisqu'en renommant φ, on rajoute la sous-formule SkL^ψ_φ.

5. se montre à partir de $\forall \varphi \sqsubseteq \psi, |\psi[\varphi]| + |\varphi| \leq |\psi| + 1 + V(\psi)$, ce qui est évident, car $|SkL^\psi_\varphi| \leq 1 + V(\psi)$.

6. $$\begin{aligned} |Rnm(R,\psi)| &= |\psi[R]| + |R| + \sum_{\varphi \in R} |Def_\psi(\varphi[R])| \\ &\leq |\psi[R]| + \sum_{\varphi \in R} |\varphi[R]| + \sum_{\varphi \in R} (3 + 2V(\psi)) + |R| \\ &\leq |\psi| + (1 + V(\psi))|R| + (3 + 2V(\psi))|R| + |R| \\ &\leq |\psi| + (5 + 3V(\psi))|R| \end{aligned}$$

⊣

Théorème 3.2 $\forall \psi, |clausale(Rnm(struct\_pres(\psi), \psi))|$ *est bornée par* $O((1 + V(\psi))|\psi|)$, *et* $p(Rnm(struct\_pres(\psi), \psi))$ *est borné par* $O(|SF(\psi)|)$.

Preuve. On pose $R = struct\_pres(\psi)$. Pour mettre $Rnm(R, \psi)$ sous forme clausale, il suffit de mettre chaque définition $Def_\psi(\varphi[R]), \varphi \in R$ ainsi que $\psi[R]$ sous forme clausale, puis de mettre le résultat sous forme prénexe et de concaténer les conjonctions. Ces deux dernières opérations font décroître la longueur, et on a donc :

$$|clausale(Rnm(R,\psi))| \leq |clausale(\psi[R])| + \sum_{\varphi \in R} |clausale(Def_\psi(\varphi[R]))|$$

ainsi que $p(Rnm(R,\psi)) = p(\psi[R]) + \sum_{\varphi \in R} p(Def_\psi(\varphi[R]))$. Nous allons donc borner chaque $|clausale(Def_\psi(\varphi[R]))|$ et $p(Def_\psi(\varphi[R]))$, en fonction de la nature de φ. Ces bornes seront (trivialement) valables pour $|clausale(\psi[R])|$ et $p(\psi[R])$. Nous constatons auparavant que $|Def_\psi(\varphi[R])| \leq 3 + 2V(\psi) + |\varphi[R]|$, inégalité que nous utiliserons dans tous les cas de figure. Nous examinons en détail le cas $pol(\varphi, \psi) = 1$, $Def_\psi(\varphi[R])$ étant donc de la forme $\forall x_1..x_n.SkL^\psi_\varphi \Rightarrow \varphi[R]$.

- si φ est une conjonction, alors $\varphi[R] = \varphi_1 \wedge .. \wedge \varphi_m$, avec $\forall i \in \{1..m\}, p(\varphi_i) = 1$. On a donc $clausale(Def_\psi(\varphi[R])) = \forall x_1..x_n. \bigwedge_{i=1}^m \neg SkL^\psi_\varphi \vee fnc(\varphi_i)$. Sachant que calculer $fnc(\varphi_i)$ consiste simplement à appliquer un certain nombre de fois la loi de la double négation, on a $|fnc(\varphi_i)| \leq |\varphi_i|$, et on a donc :

  $$\begin{aligned} |clausale(Def_\psi(\varphi[R]))| &\leq |Def_\psi(\varphi[R])| + 2m - 1 + (m-1)(1 + V(\psi)) \\ &\leq 1 + V(\psi) + |\varphi[R]| + (3 + V(\psi))|SF(\varphi[R])| \end{aligned}$$

  Quant au nombre de clauses, on a ici $p(Def_\psi(\varphi[R])) = m \leq |SF(\varphi[R])|$.

- si φ est une disjonction, alors $\varphi[R] = \varphi_1 \vee .. \vee \varphi_m$, avec $\forall i \in \{1..m\}, p(\varphi_i) = 1$. On a donc $clausale(Def_\psi(\varphi[R])) = \forall x_1..x_n. \neg SkL^\psi_\varphi \vee fnc(\varphi_1) \vee .. \vee fnc(\varphi_m)$. On a toujours $|fnc(\varphi_i)| \leq |\varphi_i|$, donc $|clausale(Def_\psi(\varphi[R]))| \leq 1 + |Def_\psi(\varphi[R])| \leq 4 + 2V(\psi) + |\varphi[R]|$. De plus $p(Def_\psi(\varphi[R])) = 1$.

- si φ est une implication, $\varphi[R] = \varphi_1 \Rightarrow \varphi_2$ avec $\overline{p}(\varphi_1) = p(\varphi_2) = 1$, et on a $clausale(Def_\psi(\varphi[R])) = clausale(Def_\psi(\neg\varphi_1 \vee \varphi_2))$, donc d'après ce qui précède $|clausale(Def_\psi(\varphi[R]))| \leq 4 + 2V(\psi) + |\neg\varphi_1 \vee \varphi_2| \leq 5 + 2V(\psi) + |\varphi[R]|$. On a $p(Def_\psi(\varphi[R])) = 1$.

- si φ est une équivalence, $\varphi[R] = \varphi_1 \Leftrightarrow \varphi_2$, alors $clausale(Def_\psi(\varphi[R])) = \forall x_1..x_n.[\neg SkL^\psi_\varphi \vee fnc(\neg\varphi_1) \vee fnc(\varphi_2)] \wedge [\neg SkL^\psi_\varphi \vee fnc(\varphi_1) \vee fnc(\neg\varphi_2)]$. Donc $|Def_\psi(\varphi[R])| \leq 10 + 3V(\psi) + 2|\varphi[R]|$ et $p(Def_\psi(\varphi[R])) = 2 \leq |SF(\varphi[R])|$.

- si φ est une universelle, on a $\varphi[R] = \forall y_1..y_m.\varphi'$ avec $p(\varphi') = 1$. Donc $clausale(Def_\psi(\varphi[R])) = \forall x_1..x_n y_1..y_m.\neg SkL^\psi_\varphi \vee fnc(\varphi')$, et donc $|clausale(Def_\psi(\varphi[R]))| \leq |Def_\psi(\varphi[R])| \leq 3 + 2V(\psi) + |\varphi[R]|$.

- si φ est une existentielle, on a $\varphi[R] = \exists y_1..y_m.\varphi'$ avec $p(\varphi') = 1$. Donc $clausale(Def_\psi(\varphi[R])) = \forall x_1..x_n.\neg SkL^\psi_\varphi \vee Sklm_{\varphi[R]}(fnc(\varphi'))$. D'après le théorème 2.2, on a $|Sklm_{\varphi[R]}(fnc(\varphi'))| \leq |fnc(\varphi')| + V(\varphi[R])SO_{\varphi[R]}(fnc(\varphi')) \leq |\varphi[R]| + V(\psi)SO(Def_\psi(\varphi[R]))$. On en déduit

$$\begin{aligned} |clausale(Def_\psi(\varphi[R]))| &\leq 1 + |Def_\psi(\varphi[R])| - |\varphi[R]| + |Sklm_{\varphi[R]}(fnc(\varphi'))| \\ &\leq 1 + |Def_\psi(\varphi[R])| + V(\psi)SO(Def_\psi(\varphi[R])) \\ &\leq 4 + 2V(\psi) + |\varphi[R]| + V(\psi)SO(Def_\psi(\varphi[R])) \end{aligned}$$

Nous examinons succinctement le cas $pol(\varphi, \psi) = -1$, très similaire au précédent. On a $Def_\psi(\varphi[R]) = \forall x_1..x_n.\varphi[R] \Rightarrow SkL^\psi_\varphi$.

- si φ est une conjonction, on a $|Def_\psi(\varphi[R])| \leq 3 + 2V(\psi) + |\varphi[R]| + |SF(\varphi[R])|$ et $p(Def_\psi(\varphi[R])) = 1$.

- si φ est une disjonction, on a $|Def_\psi(\varphi[R])| \leq 1 + V(\psi) + |\varphi[R]| + (3 + V(\psi))|SF(\varphi[R])|$ et $p(Def_\psi(\varphi[R])) \leq |SF(\varphi[R])|$.

- si φ est une implication, on a $|Def_\psi(\varphi[R])| \leq 6 + 3V(\psi) + |\varphi[R]|$ et $p(Def_\psi(\varphi[R])) = 2$.

- si φ est une équivalence, on a $|Def_\psi(\varphi[R])| \leq 8 + 3V(\psi) + 2|\varphi[R]|$ et $p(Def_\psi(\varphi[R])) = 2$.

- si φ est une universelle, on a $|Def_\psi(\varphi[R])| \leq 4 + 2V(\psi) + |\varphi[R]| + V(\psi)SO(Def_\psi(\varphi[R]))$ et $p(Def_\psi(\varphi[R])) = 1$.

- si φ est une existentielle, on a $|Def_\psi(\varphi[R])| \leq 3 + 2V(\psi) + |\varphi[R]|$ et $p(Def_\psi(\varphi[R])) = 1$

Enfin, le cas $pol(\varphi, \psi) = 0$ se ramène dans une certaine mesure aux deux précédents, en constatant que la mise sous forme clausale de $Def_\psi(\varphi[R])$ correspond à la mise sous forme clausale de deux définitions, l'une positive et l'autre négative. Si $Def_\psi(\varphi[R]) = \forall x_1..x_n.SkL^\psi_\varphi \Leftrightarrow \varphi[R]$, il est clair que $|clausale(Def_\psi(\varphi[R]))| \leq |clausale(\forall x_1..x_n.SkL^\psi_\varphi \Rightarrow \varphi[R])| + 1 + |clausale(\forall x_1..x_n.\varphi[R] \Rightarrow SkL^\psi_\varphi)|$, et il suffit donc d'ajouter les bornes obtenues dans les deux cas précédents (de même pour le nombre de clauses). Dans le cas des formules quantifiées, il convient de constater que :

$$\begin{cases} SO(\forall x_1..x_n.SkL^\psi_\varphi \Rightarrow \varphi[R]) & \leq & SO(Def_\psi(\varphi[R])) \\ SO(\forall x_1..x_n.\varphi[R] \Rightarrow SkL^\psi_\varphi) & \leq & SO(Def_\psi(\varphi[R])) \end{cases}$$

En définitive, et en tenant compte du fait que $1 \leq |SF(\varphi[R])| \leq |\varphi[R]|$, on obtient pour $|clausale(Def_\psi(\varphi[R]))|$ une borne en $O(|\varphi[R]| + V(\psi)SO(Def_\psi(\varphi[R])) + V(\psi)|SF(\varphi[R])|)$. On obtient donc de même pour $|clausale(\psi[R])|$ une borne en $O(|\psi[R]| + V(\psi)SO(\psi[R]) + V(\psi)|\psi[R]|)$. On a également une borne pour $p(Def_\psi(\varphi[R]))$ en $O(|SF(\varphi[R])|)$, et pour $p(\psi[R])$ en $O(|SF(\psi[R])|)$.

D'après le lemme 3.1, on a :

$$|\psi[R]| + \sum_{\varphi \in R} |\varphi[R]| \leq |\psi| + (1 + V(\psi))|R| \leq |\psi| + (1 + V(\psi))|SF(\psi)|$$

De même, on a $|SF(\psi[R])| + \sum_{\varphi \in R} |SF(\varphi[R])| = |SF(\psi)| + |R| \leq 2|SF(\psi)|$. Enfin, on a :

$$\begin{aligned} SO(Rnm(R,\psi)) &= SO_{Rnm(R,\psi)}(\psi[R]) + \sum_{\varphi \in R} SO_{Rnm(R,\psi)}(Def_\psi(\varphi[R])) \\ &= SO(\psi[R]) + \sum_{\varphi \in R} SO(Def_\psi(\varphi[R])) \text{ (lemme 3.1.2)} \\ &\leq SO(\psi) \text{ (théorème 3.1.3)} \end{aligned}$$

donc $|clausale(Rnm(R,\psi))|$ est borné en $O(|\psi| + V(\psi)|SF(\psi)| + V(\psi)SO(\psi))$. En bornant $|SF(\psi)|$ et $SO(\psi)$ par $|\psi|$, on obtient la borne souhaitée[1]. On obtient également, d'après ce qui précède, la borne pour $p(Rnm(R,\psi))$. ⊣

[1] Celle-ci est cependant plus fine ; elle montre que la longueur de la formule, pour ce qui est des symboles constants, n'intervient que linéairement dans la longueur de la forme clausale structure preserving.

3.3 Les renommages R_{inf}

3.3.1 Définition et propriétés

Le principe de renommage que nous adoptons ici est fondé sur une généralisation des restrictions du renommage appliquées dans [PG86]. On constate en effet que renommer une formule atomique ou une négation (pour autant que toutes les autres sous-formules soient renommées) résulte en un accroissement du nombre de clauses, ce qui suggère de renommer de façon à minimiser le nombre de clauses. D'après les résultats du chapitre 2, il suffit de s'intéresser au nombre de clauses pour éviter le comportement exponentiel de la transformation *clausale*. Il est tout aussi évident qu'on obtient ainsi un renommage projectif, appliquant sur l'ensemble des formules telles que tout renommage fait croître le nombre de clauses. Enfin, il est évident d'après le chapitre 2 qu'il est très facile de calculer effectivement le nombre de clauses d'une formule, et nous obtiendrons au chapitre 6 un algorithme très efficace pour calculer ces renommages.

Définition. On appelle *bénéfice en nombre de clauses* d'un renommage R de ψ, et on note $B(R,\psi) = p(\psi) - p(Rnm(R,\psi))$. $\forall \varphi \sqsubset \psi, B(\varphi,\psi)$ dénote $B(\{\varphi\},\psi)$, et $B(\varphi,\psi[R])$ dénote $B(\varphi[R],\psi[R])$ si $\varphi[R] \sqsubseteq \psi[R]$, et $B(SkL^{\psi}_{\varphi},\psi[R])$ si $\varphi \in Inf_R(\psi)$. $\diamond$

D'un point de vue algorithmique, nous verrons que les sous-formules de ψ seront renommées les unes après les autres, et non pas simultanément. Nous devrions donc étudier les suites (sans répétitions) plutôt que les ensembles de sous-formules, et restreindre notre étude aux suites $(\varphi_1..\varphi_n)$ telles que chaque renommage d'une sous-formule fait décroître le nombre de clauses :

Définition. Une suite $(\varphi_1..\varphi_n)$ de sous-formules de ψ, sans répétition, est dite *p-décroissante* si et seulement si $\forall i \in \{1..n\}, B(\varphi_i, Rnm(\{\varphi_1..\varphi_{i-1}\},\psi)) \geq 0$. On appelle *renommages p-décroissants* les renommages R correspondants, c'est à dire tels qu'il existe une suite p-décroissante $(\varphi_1..\varphi_n)$ telle que $R = \{\varphi_1..\varphi_n\}$. $\diamond$

Nous ne calculerons donc effectivement que des renommages p-décroissants, qui sont bien entendu des renommages à bénéfice positif. La réciproque est fausse : il existe des renommages à bénéfice positif qui ne sont pas p-décroissants ; il suffit qu'ils contiennent par exemple une formule atomique. On peut donc se demander si cette restriction n'est pas gênante, et si les meilleurs renommages, ceux qui minimisent le nombre de clauses, ne sont pas parmi ces renommages inaccessibles. Le

théorème suivant, qui exprime une propriété de monotonie, et que nous nommerons le *théorème fondamental de monotonie* (tfm en abrégé), permet de répondre à cette question :

Théorème 3.3 *Soient ψ une formule, R et R' deux renommages tels que $R \subset R'$, alors $\forall \varphi \sqsubset \psi, B(\varphi, Rnm(R,\psi)) \geq B(\varphi, Rnm(R',\psi))$.*

Preuve. On montre d'abord que $\forall \varphi' \sqsubset \psi, B(\varphi,\psi) \geq B(\varphi, Rnm(\varphi',\psi))$. On a $B(\varphi,\psi) = P^{\psi}_{\varphi}(p(\varphi),\overline{p}(\varphi)) - P^{\psi}_{\varphi}(1,1) - p(Def_{\psi}(\varphi))$. Il nous faut pour calculer $B(\varphi, Rnm(\varphi',\psi))$ distinguer trois cas :

1. si φ et φ' sont disjointes, $B(\varphi, Rnm(\varphi',\psi)) = B(\varphi,\psi[\varphi']) = P^{\psi[\varphi']}_{\varphi}(p(\varphi),\overline{p}(\varphi)) - P^{\psi[\varphi']}_{\varphi}(1,1) - p(Def_{\psi}(\varphi))$, mais les coefficients du polynôme $P^{\psi[\varphi']}_{\varphi}$ sont inférieurs à ceux de P^{ψ}_{φ} (car $p(\psi[\varphi']) \leq p(\psi)$), donc $B(\varphi,\psi[\varphi']) \leq B(\varphi,\psi)$ (puisque $p(\varphi),\overline{p}(\varphi),p(\varphi'),\overline{p}(\varphi') \geq 1$, et les coefficients de $P^{\psi[\varphi']}_{\varphi}$ sont positifs).

2. si $\varphi \sqsubseteq \varphi'$: $B(\varphi, Rnm(\varphi',\psi)) = B(\varphi, Def_{\psi}(\varphi')) = P^{Def_{\psi}(\varphi')}_{\varphi}(p(\varphi),\overline{p}(\varphi)) - P^{Def_{\psi}(\varphi')}_{\varphi}(1,1) - p(Def_{\psi}(\varphi))$. Mais les coefficients de $P^{Def_{\psi}(\varphi')}_{\varphi}$ sont inférieurs à ceux de P^{ψ}_{φ}, car $P^{\psi}_{\varphi}(x,y) = P^{\psi}_{\varphi'}(P^{\varphi'}_{\varphi}(x,y),\overline{P}^{\varphi'}_{\varphi}(x,y))$ et $P^{Def_{\psi}(\varphi')}_{\varphi}(x,y) = P^{Def_{\psi}(\varphi')}_{\varphi'}(P^{\varphi'}_{\varphi}(x,y),\overline{P}^{\varphi'}_{\varphi}(x,y))$, et les coefficients de $P^{Def_{\psi}(\varphi')}_{\varphi'}$sont évidemment inférieurs à ceux de $P^{\psi}_{\varphi'}$. Donc $B(\varphi, Def_{\psi}(\varphi')) \leq B(\varphi,\psi)$.

3. si $\varphi' \sqsubseteq \varphi$: on a $B(\varphi,\psi) + B(\varphi', Rnm(\varphi,\psi)) = B(\varphi',\psi) + B(\varphi, Rnm(\varphi',\psi))$, donc $B(\varphi, Rnm(\varphi',\psi)) \leq B(\varphi,\psi)$ si et seulement si $B(\varphi', Rnm(\varphi,\psi)) \leq B(\varphi',\psi)$, ce qui a été démontré au cas précédent, en inversant φ et φ'.

Soit $\{\varphi_1..\varphi_n\} = R' - R$, on a donc $\forall i \in \{1..n-1\}, B(\varphi, Rnm(R \cup \{\varphi_1..\varphi_i\},\psi)) \geq B(\varphi, Rnm(\varphi_{i+1}, Rnm(R \cup \{\varphi_1..\varphi_i\},\psi))) = B(\varphi, Rnm(R \cup \{\varphi_1..\varphi_{i+1}\},\psi))$. On en déduit que $B(\varphi, Rnm(R,\psi)) \geq B(\varphi, Rnm(R \cup \{\varphi_1..\varphi_n\},\psi)) = B(\varphi, Rnm(R',\psi))$. ⊣

Nous pouvons en déduire qu'il convient de ne s'intéresser qu'aux renommages p-décroissants :

Théorème 3.4 *Soient ψ une formule et R un renommage non p-décroissant de ψ, il existe un renommage $R' \subset R$, p-décroissant tel que $p(Rnm(R',\psi)) < p(Rnm(R,\psi))$.*

Preuve. Soit $\{\varphi_1..\varphi_n\} = R$, et $i \in \{1..n\}$ tel que $B(\varphi_i, Rnm(\{\varphi_1..\varphi_{i-1}\}, \psi)) < 0$, alors d'après le théorème fondamental $B(\varphi_i, Rnm(R - \{\varphi_i\}, \psi)) < 0$, et donc $p(Rnm(R - \{\varphi_i\}, \psi)) < p(Rnm(\varphi_i, Rnm(R - \{\varphi_i\}, \psi))) = p(Rnm(R, \psi))$. On a donc construit un renommage $R' \subset R, R' \neq R$ tel que $p(Rnm(R', \psi)) < p(Rnm(R, \psi))$.

Or $(SF(\psi), \subset)$ est un ordre bien fondé (car $SF(\psi)$ est fini). On peut donc en déduire par induction nœtherienne qu'il existe un renommage R' p-décroissant tel que $p(Rnm(R', \psi)) < p(Rnm(R, \psi))$. $\dashv$

Par ailleurs, nous nous intéressons évidemment aux renommages p-décroissants *maximaux*, c'est à dire tels qu'il ne soit pas possible de faire décroître le nombre de clauses en y rajoutant des sous-formules.

Définition. Un renommage R de ψ est *complet* dans ψ si et seulement si $\forall \varphi \sqsubset \psi, B(\varphi, Rnm(R, \psi)) < 0$. R est *optimal* dans ψ si et seulement si $\forall R', p(Rnm(R, \psi)) \leq p(Rnm(R', \psi))$. Nous appelons *renommage* R_{inf} tout renommage p-décroissant et complet dans ψ (ce sont les éléments maximaux de l'ensemble des renommages p-décroissants dans $(SF(\psi), \subset)$). $\diamond$

L'emploi du terme R_{inf} dans cette définition se justifie par le fait que ces renommages s'obtiennent par la procédure suivante :

$R_{inf}(\psi) =$ **debut**
$R := \emptyset$;
tant_que $\exists \varphi \sqsubset \psi, B(\varphi, Rnm(R, \psi)) \geq 0$ **faire** $R := R \cup \{\varphi\}$;
retour(R)
fin

Cette procédure permet effectivement d'obtenir tous les renommages R_{inf}, selon la stratégie de choix de la sous-formule φ dans le test (qui n'est pas spécifiée ici ; il s'agit donc d'une procédure non-déterministe). Il est également facile de voir qu'on n'obtient que ces renommages : $R_{inf}(\psi)$ est évidemment un renommage p-décroissant, et il est complet car la procédure ne s'arrête que s'il ne reste aucune sous-formule de bénéfice positif.

Enfin, on voit dans cette procédure pourquoi les renommages doivent être effectués en séquence : les bénéfices dépendent des renommages déjà effectués. Si l'on ne tenait pas compte de ceux-ci, on pourrait faire *croître* le nombre de clauses.

Une autre conséquence importante du théorème fondamental de monotonie concerne les renommages complets :

Théorème 3.5 *$\forall\psi$, soient R et R' des renommages de ψ tels que R est complet dans ψ et $R \subset R'$, alors $p(Rnm(R,\psi)) \leq p(Rnm(R',\psi))$.*

Preuve. Soient $\{\varphi_1..\varphi_n\} = R' - R$ et $\psi' = Rnm(R,\psi)$, comme R est complet dans ψ, on a $\forall i \in \{1..n\}, B(\varphi_i,\psi') < 0$, donc $B(\varphi_i, Rnm(\{\varphi_1..\varphi_{i-1}\},\psi')) < 0$ d'après le tfm, et donc $p(Rnm(\varphi_1..\varphi_{i-1},\psi')) < p(Rnm(\{\varphi_1..\varphi_i\},\psi'))$. On en déduit par récurrence que $p(Rnm(R,\psi)) = p(\psi') \leq p(Rnm(\{\varphi_1..\varphi_n\},\psi')) = p(Rnm(R',\psi))$. ⊣

Ceci montre que l'on peut obtenir des renommages optimaux avec la procédure R_{inf}. Nous verrons au chapitre 6 qu'elle peut se calculer très efficacement, pour autant que la stratégie adoptée soit simple.

Or le renommage obtenu, ainsi que le nombre de clauses, dépendent de cette stratégie. Le problème se pose donc de savoir quelle est la meilleure stratégie, tout en restant raisonnable : il est exclu d'essayer tous les renommages R_{inf} possibles. Ce sera le sujet du chapitre 4.

Cependant, il est auparavant possible d'énoncer des propriétés indépendantes de la stratégie, c'est à dire vérifiées par tous les renommages R_{inf}. Si on ne peut obtenir l'optimalité avec une stratégie quelconque, on peut cependant établir des comparaisons avec certains renommages. En particulier, il est évident que l'on obtient avec R_{inf} moins de clauses qu'avec le renommage vide, c'est à dire qu'avec la transformation *clausale*. Nous allons également montrer qu'on obtient moins de clauses qu'avec le renommage *struct_pres*, ce qui apparaît comme conséquence du théorème 3.5.

Cependant, on n'a pas nécessairement $R_{inf}(\psi) \subset struct\_pres(\psi)$, car $R_{inf}(\psi)$ peut contenir des négations, alors que $struct\_pres(\psi)$ ne contient que les formules niées, si celles ci ne sont pas elle-même des négations. Plus précisément :

Définition. La fonction SN (Sans Négation), est définie inductivement par $SN(\neg\varphi) = SN(\varphi)$ et $SN(\varphi) = \varphi$ si φ n'est pas une négation. ◇

$struct\_pres(\psi)$ contient donc les sous-formules $SN(\varphi)$ non atomiques. Nous montrons dans le lemme suivant que $R_{inf}(\psi)$ ne contient qu'une sous-formule parmi une chaîne de négations $\neg\neg..\neg\varphi$.

Lemme 3.6 *$\forall\psi, \forall\varphi, \varphi' \sqsubseteq \psi$ telles que $SN(\varphi) = SN(\varphi')$, on a :*

1. $p(\psi[\varphi]) = p(\psi[\varphi'])$
2. $p(Def_\psi(\varphi)) = p(Def_\psi(\varphi'))$
3. $p(Rnm(\varphi,\psi)) = p(Rnm(\varphi',\psi))$
4. $\varphi \neq \varphi' \Rightarrow \varphi \notin R_{inf}(\psi) \vee \varphi' \notin R_{inf}(\psi)$

Preuve. On remarque d'abord que puisque $SN(\varphi) = SN(\varphi')$, on a $\varphi \sqsubseteq \varphi'$ ou $\varphi' \sqsubseteq \varphi$.

1. on peut supposer que $\varphi \sqsubseteq \varphi'$, et donc $p(\psi[\varphi]) = P^{\psi}_{\varphi'}(p(\varphi'[\varphi]), \overline{p}(\varphi'[\varphi])) = P^{\psi}_{\varphi'}(1,1) = p(\psi[\varphi'])$.

2. si $pol(\varphi,\psi) = pol(\varphi',\psi)$, alors $p(\varphi) = p(\varphi')$ et $\overline{p}(\varphi) = \overline{p}(\varphi')$, donc $p(Def_\psi(\varphi)) = p(Def_\psi(\varphi'))$. Sinon, on peut supposer que $pol(\varphi,\psi) = 1, pol(\varphi',\psi) = -1$, et on a $p(Def_\psi(\varphi)) = p(\varphi) = \overline{p}(\varphi') = p(Def_\psi(\varphi'))$.

3. $$\begin{aligned} p(Rnm(\varphi,\psi)) &= p(\psi[\varphi]) + p(Def_\psi(\varphi)) \\ &= p(\psi[\varphi']) + p(Def_\psi(\varphi')) = p(Rnm(\varphi',\psi)) \end{aligned}$$

4. si $\varphi \neq \varphi'$, supposons que $\varphi \in R_{inf}(\psi)$ et $\varphi' \in R_{inf}(\psi)$, soit $(\varphi_1..\varphi_n)$ une suite p-décroissante telle que $\{\varphi_1..\varphi_n\} = R_{inf}(\psi)$, on a donc $\exists i,j/\varphi = \varphi_i \wedge \varphi' = \varphi_j$. On peut supposer que $i < j$, et on a alors $B(\varphi_j, Rnm(\{\varphi_1..\varphi_{j-1}\},\psi)) \leq B(\varphi', Rnm(\varphi,\psi))$ d'après le tfm. On a deux possibilités :

   (a) si $\varphi \sqsubset \varphi'$, alors $B(\varphi', Rnm(\varphi,\psi)) = B(\varphi',\psi[\varphi]) = p(\psi[\varphi]) - p(\psi[\varphi']) - p(Def_\psi(\varphi')) < 0$ car $p(\psi[\varphi]) = p(\psi[\varphi'])$.

   (b) si $\varphi' \sqsubset \varphi$, alors $B(\varphi', Rnm(\varphi,\psi)) = B(\varphi', Def_\psi(\varphi)) = p(Def_\psi(\varphi)) - p(Def_\psi(\varphi)[\varphi']) - p(Def_\psi(\varphi')) < 0$ car $p(Def_\psi(\varphi)) = p(Def_\psi(\varphi'))$.

   on a donc $B(\varphi_j, Rnm(\{\varphi_1..\varphi_{j-1}\},\psi)) < 0$, ce qui est impossible.

⊣

Il est alors possible de prouver le résultat annoncé :

Théorème 3.7 $\forall\psi, p(Rnm(R_{inf}(\psi),\psi)) \leq p(Rnm(struct\_pres(\psi),\psi))$

Preuve. Soit $R = struct\_pres(\psi) - \{SN(\xi)/\neg\xi \in R_{inf}(\psi)\} \cup \{\neg\xi/\neg\xi \in R_{inf}(\psi)\}$, montrons que $R_{inf}(\psi) \subset R$.

$\forall \varphi \in R_{inf}(\psi)$, si φ est une négation, alors $\varphi \in R$ par construction, sinon $\varphi \in struct\_pres(\psi)$, car $R_{inf}(\psi)$ ne contient pas de formule atomique (dont le bénéfice est toujours négatif). Si $\exists \neg\xi \in R_{inf}(\psi)$ tel que $SN(\xi) = \varphi$, on a $\neg\xi \neq \varphi$ et $SN(\neg\xi) = \varphi = SN(\varphi)$, ce qui est impossible d'après le lemme 3.6.4, puisqu'on a $\varphi \in R_{inf}(\psi)$ et $\neg\xi \in R_{inf}(\psi)$. Donc $\varphi \notin \{SN(\xi)/\neg\xi \in R_{inf}(\psi)\}$, et donc $\varphi \in R$.

La restriction de SN à $R - struct\_pres(\psi)$ applique dans $struct\_pres(\psi) - R$: $\forall \varphi \in R - struct\_pres(\psi)$, on a par construction $\varphi \in \{\neg\xi/\neg\xi \in R_{inf}(\psi)\}$, donc $\varphi \in R_{inf}(\psi)$ et $SN(\varphi) \notin R$, alors que $SN(\varphi) \in struct\_pres(\psi)$. Cette fonction est en fait bijective : elle est injective d'après le lemme 3.6.4, car $\forall \varphi, \varphi' \in R - struct\_pres(\psi)$ tels que $\varphi \neq \varphi'$, comme $\varphi, \varphi' \in R_{inf}(\psi)$, on a $SN(\varphi) \neq SN(\varphi')$. Elle est également surjective : $\forall \varphi \in struct\_pres(\psi) - R$, par construction $\varphi \in \{SN(\xi)/\neg\xi \in R_{inf}(\psi)\}$, donc $\exists \neg\xi \in R_{inf}(\psi)$ tel que $\varphi = SN(\xi) = SN(\neg\xi)$, et on a $\neg\xi \in R - struct\_pres(\psi)$.

Soient $\{\varphi_1..\varphi_n\} = R - struct\_pres(\psi)$ et $\psi' = Rnm(R \cap struct\_pres(\psi), \psi)$, on a donc

$$p(Rnm(R, \psi)) = p(Rnm(R - struct\_pres(\psi), \psi'))$$

$$\begin{aligned} \text{et } p(Rnm(struct\_pres(\psi), \psi)) &= p(Rnm(struct\_pres(\psi) - R, \psi')) \\ &= p(Rnm(\{SN(\varphi_1)..SN(\varphi_n)\}, \psi')) \end{aligned}$$

Or d'après le lemme 3.6.3, on a $\forall i \in \{1..n\}$,

$$\begin{aligned} p(Rnm(\{\varphi_1..\varphi_{i-1}\}, \psi')) &= p(Rnm(\{SN(\varphi_1)..SN(\varphi_{i-1})\}, \psi')) \\ \Rightarrow p(Rnm(\{\varphi_1..\varphi_i\}, \psi')) &= p(Rnm(\{SN(\varphi_1)..SN(\varphi_i)\}, \psi')) \end{aligned}$$

Par récurrence, on obtient donc $p(Rnm(R, \psi)) = p(Rnm(struct\_pres(\psi), \psi))$. Par ailleurs, $R_{inf}(\psi)$ est complet dans ψ, donc d'après le théorème 3.5, $p(Rnm(R_{inf}(\psi), \psi)) \leq p(Rnm(R, \psi))$. $\dashv$

3.3.2 Complexité

Le théorème 3.7 n'a pas pour seul intérêt d'établir l'un des avantages théoriques des renommages R_{inf} sur *struct_pres* ; il permet également d'en déduire la complexité en nombre de clauses :

Corollaire 3.8 *$p(Rnm(R_{inf}(\psi), \psi))$ est borné en $O(|SF(\psi)|)$.*

Preuve. Conséquence triviale des théorèmes 3.2 et 3.7. $\dashv$

Il est alors évident que les renommages R_{inf} permettent d'éviter le comportement exponentiel de la transformation *clausale*, répondant ainsi à l'impératif de concision

que nous nous sommes fixé. Le théorème 3.7 suggère même que les renommages R_{inf} améliorent la concision par rapport au renommage *struct_pres*. Or ceci est faux, comme on peut s'en rendre compte très facilement au vu de l'exemple suivant : considérons la suite de formule $(F_n)_{n\in\mathbb{N}}$ où $F_n = (P_1 \wedge \ldots \wedge P_n) \vee Q_1 \vee \ldots \vee Q_n$. On a $p(F_n) = n$, et aucune sous-formule n'a un bénéfice positif : n est le nombre optimal de clauses. Mais la forme clausale est :

$$fnc(F_n) = (P_1 \vee Q_1 \vee \ldots \vee Q_n) \wedge (P_2 \vee Q_1 \vee \ldots \vee Q_n) \wedge \ldots \wedge (P_n \vee Q_1 \vee \ldots \vee Q_n)$$

dont la longueur est en $O(n^2)$. Ceci prouve qu'*indépendamment de la stratégie*, la complexité en taille dans le pire des cas est au moins $O(|\psi|^2)$ dans le calcul propositionnel. Nous allons montrer que cet exemple constitue effectivement le pire des cas en bornant la complexité en taille de la transformation $clausale(Rnm(R_{inf}(\psi), \psi))$ par $O((1 + V(\psi))|\psi||SF(\psi)|)$, ce qui ne découle pas trivialement des résultats du chapitre 2 parce qu'il nous faut ici tenir compte du fait que la skolemisation a lieu après la linéarisation, transformation qui est exponentielle dans le pire des cas, ce qui peut rendre la skolemisation exponentielle :

Lemme 3.9 $\forall \psi, SO(linéaire(\psi)) \leq SO(\psi) 2^{D_\forall(\psi)}$

Preuve. Toute sous-formule atomique φ de $linéaire(\psi)$ est une sous-formule de ψ, et on a $SV_{linéaire(\psi)}(\varphi) \subset SV_\psi(\varphi)$, car $SV_\psi(\varphi)$ contient toutes les variables quantifiées au dessus de φ par un quantificateur de polarité nulle, alors que $SV_{linéaire(\psi)}(\varphi)$ n'en contient que certaines (selon la polarité du quantificateur), et donc $SO_{linéaire(\psi)}(\varphi) \leq SO_\psi(\varphi)$. Par ailleurs, chaque sous-formule atomique de ψ apparaît au plus $2^{D_\forall(\psi)}$ fois dans $linéaire(\psi)$, donc

$$SO(linéaire(\psi)) = \sum_{\substack{\varphi \sqsubseteq linéaire(\psi) \\ \varphi \text{ atomique}}} SO_{linéaire(\psi)}(\varphi) \leq \sum_{\substack{\varphi \sqsubseteq \psi \\ \varphi \text{ atomique}}} SO_\psi(\varphi) 2^{D_\forall(\psi)} = SO(\psi) 2^{D_\forall(\psi)}$$

⊣

Or la linéarisation ne peut évidemment pas être exponentielle sur les formules $Rnm(R_{inf}(\psi), \psi)$, sinon le nombre de clauses serait exponentiel, et nous en déduisons que :

Lemme 3.10 $\exists c \in I\!N / \forall \psi, D_{\Leftrightarrow}(Rnm(R_{inf}(\psi), \psi)) \leq c$

Preuve. Supposons que $\forall n \in I\!N, \exists \psi_n, D_{\Leftrightarrow}(Rnm(R_{inf}(\psi_n), \psi_n)) \geq n$, alors $p(Rnm(R_{inf}(\psi_n), \psi_n)) \geq p(\psi'_n)$, où $\psi'_n = A_n \Leftrightarrow (A_{n-1} \Leftrightarrow ..(A_1 \Leftrightarrow A_0)..)$, puisque cette formule atteint évidemment le minimum du nombre de clauses des formules dont la profondeur en équivalences est n. Or $p(\psi'_n) = \overline{p}(\psi'_n) = p(\psi'_{n-1}) + \overline{p}(\psi'_{n-1})$, et $p(\psi'_0) = \overline{p}(\psi'_0) = 1 = 2^0$, donc $p(\psi'_n) = 2^n$. On a donc exhibé une suite $(\psi_n)_{n \in I\!N}$ telle que $\forall n \in I\!N, p(Rnm(R_{inf}(\psi_n), \psi_n)) \geq 2^n$, ce qui est en contradiction avec le corollaire 3.8. $\dashv$

Ce lemme se généralise trivialement à toutes les procédures de renommage qui permettent d'éviter le comportement exponentiel de *clausale*. Nous pouvons alors évaluer la complexité en taille de la mise sous forme clausale de ces formules, et ceci indépendamment de la stratégie utilisée dans R_{inf} :

Théorème 3.11

$$\forall \psi, |clausale(Rnm(R_{inf}(\psi), \psi))| \text{ est bornée par } O((1 + V(\psi))|\psi||SF(\psi)|).$$

Preuve. Soient $\psi' = Rnm(R_{inf}(\psi), \psi)$ et $\psi'' = prénexe(Sklm(linéaire(\psi')))$, on a $|fnc(\psi'')| \leq 2p(\psi'')h(\psi'') + 1 + V(\psi'')$. On sait que $p(\psi'') = p(\psi')$ est borné en $O(|SF(\psi)|)$, et $h(\psi'') \leq |\psi''|, V(\psi'') \leq |\psi''|$, donc $|fnc(\psi'')|$ est bornée en $O(|SF(\psi)||\psi''|)$. On a :

$$\begin{aligned}
|\psi''| &\leq |Sklm(linéaire(\psi'))| \\
&\leq |linéaire(\psi')| + V(linéaire(\psi'))SO(linéaire(\psi')) \\
&\leq (|\psi'| + 2N_{\Leftrightarrow}(\psi'))2^{D_{\Leftrightarrow}(\psi')} + V(\psi')SO(\psi')2^{D_{\Leftrightarrow}(\psi')} \\
&\leq [|\psi| + (5 + 3V(\psi))|SF(\psi)| + 2N_{\Leftrightarrow}(\psi') + V(\psi)SO(\psi)]2^c
\end{aligned}$$

(en utilisant le lemme 3.1) et on a $N_{\Leftrightarrow}(Rnm(R_{inf}(\psi), \psi)) = N_{\Leftrightarrow}(\psi) + |\{\varphi \in R_{inf}(\psi) / pol(\varphi, \psi) = 0\}| \leq 2|SF(\psi)|$. Donc :

$$|\psi''| \leq [|\psi| + (7 + 3V(\psi))|SF(\psi)| + V(\psi)SO(\psi)]2^c$$

avec $SO(\psi) \leq |\psi|$ et $|SF(\psi)| \leq |\psi|$, on voit que $|\psi''|$ est bornée[2] par $O((1 + V(\psi))|\psi|)$, et donc $|fnc(\psi'')|$ est bornée en $O((1 + V(\psi))|\psi||SF(\psi)|)$. $\dashv$

[2] De même que pour la transformation structure-preserving, la borne pour $|\psi''|$ est plus fine que $O((1 + V(\psi))|\psi|)$, et montre que les symboles constants dans ψ ne peuvent intervenir que linéairement dans $fnc(\psi'')$.

Chapitre 4

Les renommages descendants

Entend comme brame,
Près des acacias,
En avril la rame
Viride du pois.

Rimbaud

4.1 Stratégies de renommage

Parmi les renommages p-décroissants on peut distinguer ceux qui sont obtenus par la procédure R_{inf} à partir de certaines stratégies de choix des sous-formules à soumettre au test de décroissance du nombre de clauses. Nous nous intéressons bien entendu à des stratégies simples, c'est à dire capable de conduire à des algorithmes efficaces. Or le calcul du bénéfice d'une sous-formule nécessite non seulement celui du nombre de clauses de cette sous-formule, mais également le calcul des coefficients du polynôme. Si on veut ne parcourir qu'une fois la formule pour y effectuer ces

tests, il est nécessaire de disposer du nombre de clauses en attribut à chaque sous-formule, et de mettre à jour ces attributs à chaque renommage effectué. Ces mises à jour ne peuvent se faire efficacement qu'en adoptant une stratégie (ou parcours) en profondeur d'abord.

Ce choix étant fixé, il reste deux possibilités : les stratégies ascendantes et descendantes. Les propriétés qui nous intéressent sont bien entendu celles qui permettent une comparaison de leur nombre de clauses respectifs. Or, il est relativement facile de trouver des exemples de formules pour lesquelles les stratégies descendantes sont meilleures que les stratégies ascendantes :

Exemple 1 Considérons la formule $\psi = \varphi \vee \varphi_1$ avec $\varphi = A \wedge (B \vee \varphi_2)$, et $p(\varphi_1) \geq 2$, $p(\varphi_2) \geq 2$. On a $a^{\psi}_{\varphi_2} = p(\varphi_1)$, donc $B(\varphi_2, \psi) \geq 0$. Mais $p(\varphi[\varphi_2]) = 2$, donc de même $B(\varphi, \psi[\varphi_2]) \geq 0$, et un renommage ascendant contient donc φ et φ_2. Par contre, on a $B(\varphi, \psi) \geq 0$, mais $B(\varphi_2, Def_{\psi}(\varphi)) < 0$, donc un renommage descendant ne renomme pas φ_2, et donne moins de clauses.

Il est par contre plus difficile d'en trouver où l'inverse est vrai :

Exemple 2 Considérons la formule $\psi = \varphi \Leftrightarrow (A \wedge B)$ avec $\varphi = \varphi' \vee (A' \wedge B')$. On a $B(\varphi, \psi) = \overline{p}(\varphi') - 2$ et $B(\varphi', Def_{\psi}(\varphi)) = p(\varphi') - 3$. Si donc $p(\varphi')$ et $\overline{p}(\varphi')$ sont suffisamment grands, nous obtenons par une stratégie descendante un renommage contenant φ et φ'. Cependant, $B(\varphi[\varphi'], \psi[\varphi']) = -1$, et donc $B(\varphi', \psi) > B(\{\varphi, \varphi'\}, \psi)$, et une stratégie ascendante nous donne un renommage contenant φ' mais pas φ, meilleur que celui obtenu par stratégie descendante.

Nous en déduisons que :

Proposition 4.1 *La classe des stratégies descendantes et celle des stratégies ascendantes ne sont pas comparables en nombre de clauses, et ne sont donc pas optimales.*

Cependant, la facilité avec laquelle on trouve des exemples de formules qui vont à l'avantage des stratégies descendantes suggère que cette dernière doit être meilleure sur une classe importante de formules, et la présence d'une équivalence dans l'exemple 2 qu'il convient de s'intéresser aux formules linéaires. Nous allons dans la suite démontrer une propriété bien plus forte qu'une simple comparaison entre les stratégies ascendantes et descendantes, à savoir que ces dernières permettent d'obtenir, dans le cas linéaire, *le nombre optimal de clauses parmi tous les renommages possibles.*

Dans un premier temps, nous établirons cette propriété non pas sur les renommages obtenus par stratégies descendantes, mais sur ce que nous appellerons les renommages descendants, dont la définition est plus simple. C'est au cours de

l'étude algorithmique, au chapitre 6, où l'on définira une stratégie descendante particulière, que nous établirons la correspondance entre ces deux notions. Travailler directement sur des renommages présente cet avantage de permettre de faire abstraction de l'*ordre* dans lequel sont produites les sous-formules au cours du calcul de $R_{inf}(\psi)$. Cette abstraction nécessite la notion d'indépendance suivante :

Définition. Un renommage R de $\varphi \sqsubseteq \psi$ est *libre* dans ψ si et seulement si $\forall(\varphi_i)_{i=1}^n, R = \{\varphi_1..\varphi_n\} \Rightarrow \forall i \in \{1..n\}, B(\varphi_i, Rnm(\varphi_1..\varphi_{i-1}, \psi)) \geq 0$. $\diamond$

Un renommage est donc libre s'il peut être obtenu par la procédure R_{inf} dans n'importe quel ordre (selon la stratégie employée). En utilisant le théorème fondamental de monotonie, on peut immédiatement en donner une caractérisation beaucoup plus simple :

Théorème 4.2 *Un renommage R de φ est libre dans ψ si et seulement si $\forall \varphi' \in R, B(\varphi', Rnm(R - \{\varphi'\}, \psi)) \geq 0$.*

Preuve. Cette condition est évidemment nécessaire : il suffit de considérer une suite $(\varphi_i)_{i=1}^n$ telle que $\varphi_n = \varphi$. Elle est également suffisante : $\forall i \in \{1..n\}$, on a par tfm $B(\varphi_i, Rnm(\varphi_1..\varphi_{i-1}, \psi)) \geq B(\varphi_i, Rnm(R - \{\varphi_i\}, \psi)) \geq 0$. $\dashv$

C'est évidemment une condition très forte sur les renommages ; en particulier, elle est nécessaire à l'optimalité :

Corollaire 4.3 *Tout renommage optimal de ψ est libre dans ψ.*

Preuve. Si R n'est pas libre, $\exists \varphi \in R / B(\varphi, Rnm(R - \{\varphi\}, \psi)) < 0$, donc $p(Rnm(R - \{\varphi\}, \psi)) < p(Rnm(R, \psi))$, et R n'est pas optimal. $\dashv$

Quant à la réciproque, il est très difficile de savoir si elle est vraie ou non ; existe-t-il des renommages libres non optimaux? On peut raisonnablement conjecturer qu'il n'en existe pas, du moins dans le cas des formules linéaires, ce qui fournirait un test simple d'optimalité, mais pas directement une méthode pour construire des renommages optimaux. Nous allons donc nous restreindre à une classe plus "constructive" de renommages libres :

Définition. Un renommage R de φ est *descendant* dans ψ si et seulement si R est libre dans ψ et $\forall \varphi' \sqsubseteq \varphi$, si $Sup_R(\varphi')$ existe, alors $B(\varphi', Def_\psi(Sup_R(\varphi')[R - SF(\varphi')])) < 0$ sinon $B(\varphi', \psi[R - SF(\varphi')]) < 0$ (que nous appelons *condition de saturation*). $\diamond$

On voit dans l'expression de cette condition de saturation l'intérêt de faire abstraction de l'ordre : on suppose ici que tous les renommages disjoints de φ' ont eu

lieu *avant* les renommages en φ', ce qui ne sera *effectivement* le cas, dans un calcul $R_{inf}(\psi)$, que pour un nombre restreint de sous-formules. Cette simplification est essentielle pour la suite, et il nous faudra montrer au chapitre 6 que le renommage obtenu avec un parcours descendant est libre.

Nous pouvons déjà énoncer une propriété élémentaire des renommages descendants :

Théorème 4.4 *Soient $\varphi \sqsubset \psi$ et R un renommage de φ descendant dans ψ, alors*

$$\varphi \in R \Leftrightarrow B(\varphi, \psi) \geq 0$$

Preuve. Si $\varphi \notin R$, $Sup_R(\varphi)$ n'existe pas, et on a $B(\varphi, \psi[R - SF(\varphi)]) = B(\varphi, \psi) < 0$ d'après la condition de saturation (avec $\varphi' = \varphi$).

Si $\varphi \in R$, on a d'après le théorème fondamental de monotonie $B(\varphi, \psi) \geq B(\varphi, Rnm(R - \{\varphi\}, \psi)) \geq 0$ car R est libre dans ψ. $\dashv$

4.2 Invariance des renommages descendants

Nous nous limitons dans la suite, et jusqu'à la fin de ce chapitre, aux *formules linéaires sous forme normale négative*, c'est à dire sans implication et telles que les négations n'apparaissent que devant des formules atomiques. Cette dernière restriction est destinée à simplifier les preuves de certains résultats, car cela évite de considérer tous les cas possibles de polarités, qui mèneraient à d'inutiles répétitions. On peut également considérer que les polynômes P_φ^ψ ne dépendent plus que d'une variable, ce qui allège les notations. De plus, nous utiliserons systématiquement le fait qu'on a alors $\forall \varphi' \sqsubseteq \varphi \sqsubset \psi, P_{\varphi'}^{Rnm(\varphi,\psi)} = P_{\varphi'}^{Def_\psi(\varphi)} = P_{\varphi'}^{\varphi}$, ce qui donne en particulier $B(\varphi', Def_\psi(\varphi)) = B(\varphi', \varphi)$, d'où une simplification de la condition de saturation.

Enfin, on peut remarquer qu'il n'est pas absolument nécessaire d'établir ces résultats pour toutes les formules linéaires, puisque il suffit d'effectuer une mise sous forme normale négative avant le renommage, ce qui ne pose pas de problèmes puisque cette transformation a une complexité linéaire en taille et en temps. Il faut cependant rappeler que les résultats que nous obtiendrons seront valides pour toutes les formules linéaires.

Avant de nous attaquer à l'optimalité, il nous faut d'abord démontrer une propriété d'invariance, à savoir que de tous les renommages descendants on obtient le

même nombre de clauses. Nous en déduirons ainsi l'optimalité de *tous* les renommages descendants, ce qui nous permettra de choisir une stratégie descendante simple, et d'obtenir à la fois l'efficacité, la facilité de programmation et la possibilité de prouver sans trop de difficulté que cette stratégie permet bien de calculer un renommage descendant.

La preuve de cette propriété d'invariance n'est cependant pas directe, et il nous faut établir certains faits concernant les renommages descendants, avant d'y parvenir. En particulier, le théorème suivant nous permettra d'utiliser l'induction structurelle sur les formules :

Théorème 4.5 *Soient $\varphi \sqsubseteq \psi, R$ un renommage de φ descendant dans ψ, $\varphi' \sqsubseteq \varphi$, $R' = R \cap SF^*(\varphi')$,et $\psi' = Sup_R(\varphi')[R - R']$ si $Sup_R(\varphi')$ existe, $\psi' = \psi[R - R']$ sinon, alors R' est un renommage de φ' descendant dans ψ'.*

Preuve. On montre d'abord que R' est libre dans ψ' : $\forall \xi \in R'$, soit $e = B(\xi, Rnm(R' - \{\xi\}, \psi'))$, on distingue deux cas.

1. si $\xi \notin Inf_{R'}(\varphi')$, alors $\exists \xi' \in R' / \xi \in Inf_{R'}(\xi')$ (car $\xi \in R'$). Donc

$$\begin{aligned} e &= B(\xi, \xi'[R' - \{\xi\}]) \\ &= B(\xi, \xi'[R - \{\xi\}]) \qquad \text{car } \xi' \sqsubseteq \varphi' \\ &= B(\xi, Rnm(R - \{\xi\}, \psi)) \qquad \text{car } \xi \in Inf_{R'}(\xi') = Inf_R(\xi') \\ &\geq 0 \qquad \text{car } R \text{ est libre dans } \psi \end{aligned}$$

2. si $\xi \in Inf_{R'}(\varphi')$, alors $e = B(\xi, \psi'[R' - \{\xi\}])$ (car $\varphi' \notin R'$). Donc, si $Sup_R(\varphi')$ existe, on a

$$\begin{aligned} e &= B(\xi, Sup_R(\varphi')[R - R'][R' - \{\xi\}]) \\ &= B(\xi, Sup_R(\varphi')[R - \{\xi\}]) \qquad \text{car } \xi \in R' \subset R \\ &= B(\xi, Rnm(R - \{\xi\}, \psi)) \qquad \text{car } Sup_R(\xi) = Sup_R(\varphi') \end{aligned}$$

   Sinon, on a de même $e = B(\xi, \psi[R - \{\xi\}]) = B(\xi, Rnm(R - \{\xi\}, \psi))$ (car $Sup_R(\xi)$ n'existe pas non plus). Dans les deux cas, du fait que R est libre dans ψ, on a $e \geq 0$.

Il faut ensuite établir la condition de saturation : $\forall \xi \sqsubseteq \varphi'$, si $Sup_{R'}(\xi)$ existe, alors $Sup_R(\xi) = Sup_{R'}(\xi)$, donc

$$\begin{aligned} B(\xi, Sup_{R'}(\xi)[R' - SF(\xi)]) &= B(\xi, Sup_R(\xi)[R - SF(\xi)]) \qquad \text{car } Sup_R(\xi) \sqsubseteq \varphi' \\ &< 0 \qquad \text{car } R \text{ est descendant dans } \psi \end{aligned}$$

Sinon, soit $e = B(\xi, \psi'[R' - SF(\xi)])$, si $Sup_R(\xi)$ existe, alors $Sup_R(\xi) = Sup_R(\varphi')$ et on a

$$\begin{aligned} e &= B(\xi, Sup_R(\xi)[R - R'][R' - SF(\xi)]) \\ &= B(\xi, Sup_R(\xi)[R - SF(\xi)]) \qquad \text{car } (R - R') \cap SF(\xi) = \emptyset \\ &< 0 \qquad \text{car } R \text{ est descendant dans } \psi \end{aligned}$$

Sinon, on a $\psi' = \psi[R - R']$, et on procède de même, en remplaçant $Sup_R(\xi)$ par ψ. $\dashv$

Il nous faut également établir des propriétés particulières concernant les conjonctions et les disjonctions. Quant aux conjonctions, l'invariance n'est pas très difficile à obtenir ; elle provient du fait que le bénéfice d'une sous-formule ne dépend pas de ses conjoints, ce que nous exprimons de la façon suivante :

Lemme 4.6 *Soient $\varphi \sqsubseteq \psi$, avec $\varphi = \varphi_1 \wedge .. \wedge \varphi_n$, $k \in \{1..n\}$, $\varphi' \in SF(\varphi_k)$ et $\varphi'_1 .. \varphi'_n$ des formules quelconques, on note $\psi' = \psi[\varphi_i \leftarrow \varphi'_i]_{i \neq k}$, alors $B(\varphi', \psi) = B(\varphi', \psi')$.*

Preuve. Soient $a = a^{\psi}_{\varphi}$, $a' = a^{\varphi_k}_{\varphi'}$, b, b' tels que $P^{\psi}_{\varphi} = \lambda x.ax + b$ et $P^{\varphi_k}_{\varphi'} = \lambda x.a'x + b'$, $c = \sum_{i \neq k} p(\varphi_i)$ et $c' = \sum_{i \neq k} p(\varphi'_i)$, on a $P^{\psi}_{\varphi'} = \lambda x.a(a'x + b' + c) + b$ et $P^{\psi'}_{\varphi'} = \lambda x.a(a'x + b' + c') + b$, donc $B(\varphi', \psi) = (aa' - 1)(p(\varphi') - 1) - 1 = B(\varphi', \psi')$. $\dashv$

A partir de ce lemme on peut établir un résultat analogue au théorème 4.5, mais plus simple car limité aux conjonctions :

Lemme 4.7 *Soient $\varphi \sqsubseteq \psi$, avec $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_n$, $k \in \{1..n\}$, R un renommage de φ descendant dans ψ, alors $R' = R \cap SF(\varphi_k)$ est descendant dans ψ.*

Preuve. R' est libre dans ψ : $\forall \varphi' \in R'$,

$$\begin{aligned} B(\varphi', Rnm(R' - \{\varphi'\}, \psi)) &\geq B(\varphi', Rnm(R - \{\varphi'\}, \psi)) \qquad \text{d'après tfm} \\ &\geq 0 \qquad \text{car } R \text{ est libre dans } \psi \end{aligned}$$

Montrons que la condition de saturation est vérifiée : $\forall \varphi' \sqsubseteq \varphi_k$, si $Sup_{R'}(\varphi')$ existe, alors

$$\begin{aligned} & B(\varphi', Sup_{R'}(\varphi')[R' - SF(\varphi')]) \\ &= B(\varphi', Sup_R(\varphi')[R - SF(\varphi')]) \qquad \text{car } Sup_R(\varphi') = Sup_{R'}(\varphi') \in R' \\ &< 0 \qquad \text{car } R \text{ est descendant dans } \psi \end{aligned}$$

Sinon

$$
\begin{aligned}
& B(\varphi', \psi[R' - SF(\varphi')]) \\
&\quad = B(\varphi', \psi[R - SF(\varphi')]) \qquad \text{d'après le lemme 4.6, car } (R - R') \cap SF(\varphi') = \emptyset \\
&\quad < 0 \qquad \text{car } R \text{ est descendant dans } \psi
\end{aligned}
$$

⊣

Par contre, il est indifférent d'instancier ψ en des sous-formules disjointes, pour autant que ces sous-formules soient réunies par une conjonction. Cette indépendance vaut pour les bénéfices, comme il est exprimé au lemme 4.6, et donc également pour les renommages descendants :

Lemme 4.8 *Soient $\varphi \sqsubseteq \psi$ avec $\varphi = \varphi_1 \wedge .. \wedge \varphi_n$, $k \in \{1..n\}$, R un renommage de φ_k descendant dans ψ, et $R' \subset \bigcup_{i \neq k} SF(\varphi_i)$, alors R est descendant dans $\psi' = \psi[R']$.*

Preuve. Soient $\varphi'_i = \varphi_i[R']$, on a $\psi' = \psi[\varphi_i \leftarrow \varphi'_i]_{i \neq k}$, donc d'après le lemme 4.6 on a $\forall \varphi' \in R$,

$$
\begin{aligned}
& B(\varphi', Rnm(R - \{\varphi'\}, \psi')) \\
&\quad = B(\varphi', Rnm(R - \{\varphi'\}, \psi)) \\
&\qquad\quad \text{car } Rnm(R - \{\varphi'\}, \psi[R']) = Rnm(R - \{\varphi'\}, \psi)[R'] \\
&\quad \geq 0 \qquad \text{car } R \text{ est libre dans } \psi
\end{aligned}
$$

De plus, $\forall \varphi' \sqsubseteq \varphi_k$, si $Sup_R(\varphi')$ existe, la condition de saturation est la même pour ψ' et pour ψ, donc est acquise pour ψ', et sinon

$$
\begin{aligned}
B(\varphi', \psi'[R - SF(\varphi')]) &= B(\varphi', \psi[R - SF(\varphi')]) \qquad \text{toujours d'après le lemme 4.6} \\
&< 0 \qquad \text{car } R \text{ est descendant dans } \psi
\end{aligned}
$$

⊣

En ce qui concerne les disjonctions, il n'y a pas de notion équivalente d'indépendance des bénéfices, et l'invariance sera plus difficile à obtenir. Il nous faut pour cela considérer la *façon* dont sont effectués les renommages dans une disjonction, ce qui se décrit, du moins pour les disjonctions qui n'apparaissent pas trop profondément dans la formule (d'où la condition sur a^{ψ}_{φ}), comme suit :

Lemme 4.9 *Soient $\varphi = \varphi_1 \vee \ldots \vee \varphi_n \sqsubseteq \psi$ tel que $a^{\psi}_{\varphi} = 1$, R un renommage de φ descendant dans ψ, et $S = \{i / p(\varphi_i) > 1\}$, il existe un élément k de S tel que $\varphi_k \notin R$, et $\forall i \in S - \{k\}, \varphi_i \in R$.*

Preuve. On suppose d'abord que $\forall k \in S, \varphi_k \in R$, donc $\forall i \in S$

$$\begin{aligned}
& B(\varphi_i, Rnm(R - \{\varphi_i\}, \psi)) \\
&= B(\varphi_i, \psi[R - \{\varphi_i\}]) \\
&\qquad \text{car } a^{\psi}_{\varphi} = 1, \text{ donc } B(\varphi, \psi) < 0, \text{ donc } \varphi \notin R \text{ (théorème 4.4)} \\
&= p(\psi[R - \{\varphi_i\}]) - p(\psi[R]) - p(\varphi_i[R]) \\
&= p(\varphi[R - \{\varphi_i\}]) - p(\varphi[R]) - p(\varphi_i[R]) \qquad \text{car } a^{\psi}_{\varphi} = 1 \\
&= p(\varphi_i[R]) - 1 - p(\varphi_i[R]) \\
&< 0
\end{aligned}$$

ce qui est impossible car R est libre dans ψ. Donc $\exists k \in S / \varphi_k \notin R$. Supposons maintenant que $\exists i \in S - \{k\} / \varphi_i \notin R$, on a alors

$$\begin{aligned}
B(\varphi_i, \psi[R - SF(\varphi_i)]) &= p(\varphi[R - SF(\varphi_i)]) - p(\varphi[R \cup \{\varphi_i\}]) - p(\varphi_i) \\
&\geq p(\varphi_k)p(\varphi_i) - p(\varphi_k) - p(\varphi_i) \\
&\geq 0 \qquad \text{car } p(\varphi_k) > 1 \text{ et } p(\varphi_i) > 1
\end{aligned}$$

ce qui est en contradiction avec la condition de saturation. On en déduit que $\forall i \in S - \{k\} / \varphi_i \in R$. $\dashv$

D'après ce lemme, il est clair que l'hypothèse faite sur a^{ψ}_{φ}, vérifiée à la racine, va se propager en descendant dans la formule, autant dans les conjonctions que dans les disjonctions. On pourra donc l'utiliser de façon plus suivie, en particulier grâce au lemme suivant.

Lemme 4.10 *Soient $\varphi \sqsubseteq \psi$ avec $a^{\psi}_{\varphi} = 1$, R un renommage de φ descendant dans ψ, alors R est descendant dans φ, et $B(R, \varphi) = B(R, \psi)$.*

Preuve. On montre d'abord que R est libre dans φ : $\forall \varphi' \in R$, soit $e = B(\varphi', Rnm(R - \{\varphi'\}, \varphi))$, si $\varphi' \notin Inf_R(\varphi)$ alors $\exists \xi \in R / \varphi' \in Inf_R(\xi)$ et $e = B(\varphi', \xi[R - \{\varphi'\}]) = B(\varphi', Rnm(R - \{\varphi'\}, \psi)) \geq 0$, sinon

$$\begin{aligned}
e &= B(\varphi', \varphi[R - \{\varphi'\}]) \\
&= p(\varphi[R - \{\varphi'\}]) - p(\varphi[R]) - p(\varphi'[R]) \\
&= p(\psi[R - \{\varphi'\}]) - p(\psi[R]) - p(\varphi'[R]) \qquad \text{car } a^{\psi}_{\varphi} = 1 \\
&= B(\varphi', \psi[R - \{\varphi'\}]) \\
&= B(\varphi', Rnm(R - \{\varphi'\}, \psi)) \\
&\geq 0 \qquad \text{car } R \text{ est libre dans } \psi
\end{aligned}$$

On établit ensuite la condition de saturation : $\forall \varphi' \sqsubseteq \varphi$, si $Sup_R(\varphi')$ existe, la condition est la même pour φ et pour ψ, et sinon

$$\begin{aligned} B(\varphi', \varphi[R - SF(\varphi')]) &= p(\varphi[R - SF(\varphi')]) - p(\varphi[R \cup \{\varphi'\}]) - p(\varphi') \\ &= p(\psi[R - SF(\varphi')]) - p(\psi[R \cup \{\varphi'\}]) - p(\varphi') \\ &= B(\varphi', \psi[R - SF(\varphi')]) \\ &< 0 \end{aligned}$$

Enfin, on a

$$\begin{aligned} B(R, \varphi) &= p(\varphi) - p(Rnm(R, \varphi)) \\ &= p(\varphi) - p(\varphi[R]) - \sum_{\varphi' \in R} p(\varphi'[R]) \\ &= p(\psi) - p(\psi[R]) - \sum_{\varphi' \in R} p(\varphi'[R]) \\ &= B(R, \psi) \end{aligned}$$

⊣

Nous pouvons maintenant démontrer l'invariance du nombre de clauses des renommages descendants.

Théorème 4.11 *Soient $\varphi \sqsubseteq \psi$, R et R' deux renommages de φ descendants dans ψ, on a $p(Rnm(R, \psi)) = p(Rnm(R', \psi))$.*

Preuve. On procède par induction structurelle sur φ, en montrant que l'égalité est vérifiée quelque soient ψ (sur-formule de φ), R et R'.

Si φ est un littéral, on a $R = R' = \emptyset$, donc $p(Rnm(R, \psi)) = p(Rnm(R', \psi)) = p(\psi)$.

Sinon, soient $\psi \sqsupseteq \varphi$, R et R' deux renommages de φ descendants dans ψ, on suppose d'abord que $B(\varphi, \psi) < 0$, et on a donc par le théorème 4.4, $\varphi \notin R$ et $\varphi \notin R'$. On a trois cas possibles :

1. $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_n$.

   Soient $R_k = R \cap SF(\varphi_k)$ et $R'_k = R' \cap SF(\varphi_k)$ pour $k \in \{1..n\}$. D'après le lemme 4.7, R_k et R'_k sont descendants dans ψ. Soit

$$R''_k = \bigcup_{i=1}^{k-1} R'_i \cup \bigcup_{i=k+1}^{n} R_i$$

on a $R''_k \subset \bigcup_{i \neq k} SF(\varphi_i)$, donc d'après le lemme 4.8, R_k et R'_k sont descendants dans $\psi[R''_k]$. Par hypothèse d'induction, on a $p(Rnm(R_k, \psi[R''_k])) = p(Rnm(R'_k, \psi[R''_k]))$, c'est à dire

$$P^{\psi}_{\varphi}(\sum_{i=1}^{k-1} p(\varphi_i[R'_i]) + \sum_{i=k}^{n} p(\varphi_i[R_i])) + r_k$$

$$= P^{\psi}_{\varphi}(\sum_{i=1}^{k} p(\varphi_i[R'_i]) + \sum_{i=k+1}^{n} p(\varphi_i[R_i])) + r'_k$$

$$\text{où } r_k = \sum_{\xi \in R_k} p(\xi[R_k]) \text{ et } r'_k = \sum_{\xi \in R'_k} p(\xi[R'_k]).$$

Or on a $p(Rnm(R, \psi)) = P^{\psi}_{\varphi}(\sum_{i=1}^{n} p(\varphi_i[R_i])) + \sum_{i=1}^{n} r_i$, on peut donc montrer par récurrence sur k que $\forall k \in \{1..n+1\}$,

$$p(Rnm(R, \psi)) = P^{\psi}_{\varphi}(\sum_{i=1}^{k-1} p(\varphi_i[R'_i]) + \sum_{i=k}^{n} p(\varphi_i[R_i])) + \sum_{i=1}^{k-1} r'_i + \sum_{i=k}^{n} r_i.$$

En particulier pour $k = n+1$, on a $p(Rnm(R, \psi)) = P^{\psi}_{\varphi}(\sum_{i=1}^{n} p(\varphi_i[R'_i])) + \sum_{i=1}^{n} r'_i = p(Rnm(R', \psi))$.

2. $\varphi = \varphi_1 \vee \ldots \vee \varphi_n$.

Si $p(\varphi) = 1$, on a $\forall \varphi' \sqsubseteq \varphi, B(\varphi', \psi) < 0$, donc $R = R' = \emptyset$, d'où l'égalité. Sinon, comme $B(\varphi, \psi) < 0$, on a $a^{\psi}_{\varphi} = 1$. Soit donc $S = \{i / p(\varphi_i) > 1\}$, d'après le lemme 4.9, $\exists k, k' \in S$ tels que $\varphi_k \notin R, \varphi_{k'} \notin R'$ et $\forall i \in S, \varphi_i \in R$ si $i \neq k$ et $\varphi_i \in R'$ si $i \neq k'$.

$\forall i \in S$, soient $R_i = R \cap SF^*(\varphi_i)$ et $R'_i = R' \cap SF^*(\varphi_i)$. Si $i \in S - \{k, k'\}$, d'après le théorème 4.5, R_i et R'_i sont descendants dans $Sup_R(\varphi_i)[R - R_i] = Sup_{R'}(\varphi_i)[R' - R'_i] = \varphi_i$ (car $\varphi_i \in R \cap R'$), et donc par hypothèse d'induction on a $p(Rnm(R_i, \varphi_i)) = p(Rnm(R'_i, \varphi_i))$.

Par ailleurs, pour $i \in S - \{k\}$, on a

$$\begin{aligned} p(Rnm(R_i, \varphi_i)) &= p(\varphi_i[R_i]) + \sum_{\xi \in R_i} p(Def_{\varphi_i}(\xi[R_i])) \\ &= \sum_{\xi \in R \cup \{\varphi_i\}} p(Def_{\psi}(\xi[R_i])) \end{aligned}$$

et on a évidemment $R = R_k \cup \bigcup_{i \in S - \{k\}} (R_i \cup \{\varphi_i\})$, donc on peut en déduire (de même pour R') que :

$$\begin{cases} p(Rnm(R, \psi)) = p(Rnm(R_k, \psi[R - R_k])) + \sum_{i \in S - \{k\}} p(Rnm(R_i, \varphi_i)) \\ p(Rnm(R', \psi)) = p(Rnm(R'_{k'}, \psi[R' - R'_{k'}])) + \sum_{i \in S - \{k'\}} p(Rnm(R'_i, \varphi_i)) \end{cases}$$

On obtient donc, d'après l'hypothèse d'induction exprimée ci-dessus pour $i \in S - \{k, k'\}$, l'égalité

$$p(Rnm(R,\psi)) - p(Rnm(R',\psi)) = \\ p(Rnm(R_k, \psi[R - R_k])) + p(Rnm(R_{k'}, \varphi_{k'})) - p(Rnm(R'_{k'}, \psi[R' - R'_{k'}])) \\ -p(Rnm(R'_k, \varphi_k))$$

En notant $\psi' = \psi[\varphi_i]_{i \in S-\{k,k'\}}$, puisque $\psi[R - R_k] = \psi'[\varphi_{k'}]$ et $\psi[R' - R'_{k'}] = \psi'[\varphi_k]$, on obtient

$$p(Rnm(R,\psi)) - p(Rnm(R',\psi)) = \\ p(Rnm(R_k, \psi'[\varphi_{k'}])) - p(Rnm(R'_k, \varphi_k)) - \\ [p(Rnm(R'_{k'}, \psi'[\varphi_k])) - p(Rnm(R_{k'}, \varphi_{k'}))]$$

Cependant, il existe b tel que $P_{\varphi_k}^{\psi'[\varphi_{k'}]} = P_{\varphi_{k'}}^{\psi'[\varphi_k]} = P_{\varphi}^{\psi} = \lambda x.x + b$, tandis que d'après le théorème 4.5, R_k (resp. $R'_{k'}$) est descendant dans $\psi[R - R_k] = \psi'[\varphi_{k'}]$ (resp. dans $\psi'[\varphi_k]$), donc d'après le lemme 4.10, R_k (resp. $R'_{k'}$) est descendant dans φ_k (resp. $\varphi_{k'}$), et $B(R_k, \psi'[\varphi_{k'}]) = B(R_k, \varphi_k)$ (resp. $B(R'_{k'}, \psi'[\varphi_k]) = B(R'_{k'}, \varphi_{k'})$), c'est à dire

$$\begin{cases} p(\psi'[\varphi_{k'}]) - p(Rnm(R_k, \psi'[\varphi_{k'}])) = p(\varphi_k) - p(Rnm(R_k, \varphi_k)) \\ p(\psi'[\varphi_k]) - p(Rnm(R'_{k'}, \psi'[\varphi_k])) = p(\varphi_{k'}) - p(Rnm(R'_{k'}, \varphi_{k'})) \end{cases}$$

Or $p(\psi'[\varphi_{k'}]) = p(\varphi_k) + b$ et $p(\psi'[\varphi_k]) = p(\varphi_{k'}) + b$, donc $p(Rnm(R_k, \psi'[\varphi_{k'}])) = p(Rnm(R_k, \varphi_k)) + b$ et $p(Rnm(R'_{k'}, \psi'[\varphi_k])) = p(Rnm(R'_{k'}, \varphi_{k'})) + b$. On obtient donc

$$p(Rnm(R,\psi)) - p(Rnm(R',\psi)) = p(Rnm(R_k, \varphi_k)) - p(Rnm(R'_k, \varphi_k)) - \\ [p(Rnm(R'_{k'}, \varphi_{k'})) - p(Rnm(R_{k'}, \varphi_{k'}))]$$

D'après le théorème 4.5, R'_k (resp. $R_{k'}$) est descendant dans $Sup_{R'}(\varphi_k)[R - R'_k] = \varphi_k$ (resp. dans $\varphi_{k'}$), donc par hypothèse d'induction, $p(Rnm(R_k, \varphi_k)) = p(Rnm(R'_k, \varphi_k))$ (resp. $p(Rnm(R'_{k'}, \varphi_{k'})) = p(Rnm(R_{k'}, \varphi_{k'}))$). On en déduit que $p(Rnm(R,\psi)) = p(Rnm(R',\psi))$.

3. $\varphi = Qx_1..x_n.\varphi'$, où $Q \in \{\forall, \exists\}$. Si $p(\varphi) = 1$, voir le cas précédent, sinon, on a de même $a_{\varphi}^{\psi} = 1$, donc $a_{\varphi'}^{\psi} = 1$. De plus, $R = R \cap SF(\varphi')$ (resp. $R' = R' \cap SF(\varphi')$), donc d'après le théorème 4.5, R (resp. R') est un renommage de φ' descendant dans ψ. Par hypothèse d'induction, on a donc $p(Rnm(R,\psi)) = p(Rnm(R',\psi))$.

On suppose maintenant que $B(\varphi,\psi) \geq 0$, alors d'après le théorème 4.4 $\varphi \in R \cap R'$, donc d'après le théorème 4.5, $R - \{\varphi\}$ et $R' - \{\varphi\}$ sont des renommages de φ descendants dans φ ($Sup_R(\varphi) = Sup_{R'}(\varphi) = \varphi$). On a $B(\varphi,\varphi) < 0$, donc d'après ce qui précède, $p(Rnm(R - \{\varphi\},\varphi)) = p(Rnm(R' - \{\varphi\},\varphi))$. Or $p(Rnm(R,\psi)) = p(\psi[\varphi]) + p(Rnm(R - \{\varphi\},\varphi))$, et de même pour R', donc $p(Rnm(R,\psi)) = p(Rnm(R',\psi))$. L'induction est complète, et donc l'égalité est vraie quelque soit φ. $\dashv$

Il est à noter que ce théorème *ne se généralise pas aux formules non linéaires.* En effet, l'exemple présenté au début de ce chapitre montre qu'en inversant l'ordre des appels récursifs, donc la stratégie, on peut obtenir un meilleur renommage (à savoir celui qui est obtenu avec la stratégie ascendante). Il n'est donc pas exclu qu'une stratégie descendante soit optimale pour toutes les formules, mais il est certain que dans celle-ci, l'ordre des appels récursifs sur les sous-formules dépendrait de ces sous-formules. Il est donc très probable qu'un tel algorithme aurait une complexité supérieure à celui présenté plus loin (chapitre 6), qui correspond à la stratégie descendante, en profondeur d'abord et gauche-droite.

4.3 Optimalité des renommages descendants

Nous pouvons maintenant aborder le problème de l'optimalité du nombre de clauses des renommages descendants, qui se trouve heureusement simplifié par le résultat d'invariance que nous venons d'obtenir. Il nous suffit en effet d'établir *l'existence* d'un renommage descendant optimal pour assurer l'optimalité de *tous* les renommages descendants. Dans la suite, nous travaillerons principalement sur les renommages optimaux, et y découvrirons un renommage descendant en utilisant une technique inspirée des ordres bien fondés, mais généralisée aux préordres pour des raisons de simplicité. Il est en effet facile de définir un préordre sur les renommages qui pourra être aisément rapproché de la propriété de saturation des renommages descendants :

Définition.

i) la profondeur dans ψ d'un renommage R de ψ est

$$d_\psi(R) = \sum_{\varphi \in R} d_\psi(\varphi)$$

ii) soient R et R' deux renommages de ψ, on note $R \preceq R'$ si et seulement si $d_\psi(R) \leq d_\psi(R')$.

$\diamond$

Ce n'est pas par abus de notation que ψ n'est pas spécifiée dans $R \preceq R'$; en effet, si R et R' sont également des renommages de ψ', on a aussi $d_{\psi'}(R) \leq d_{\psi'}(R')$. L'important est de se référer à la même sur-formule des éléments de R et R'. La relation $\preceq$ est évidemment un préordre, et nous utiliserons principalement la propriété suivante :

Théorème 4.12 *Il n'existe pas de suite infinie $(R_i)_{i\in\mathbb{N}}$ de renommages de ψ telle que $\forall i \in \mathbb{N}, R_{i+1} \preceq R_i \wedge R_i \not\preceq R_{i+1}$.*

Preuve. Supposons qu'il en existe une, la suite d'entiers naturels $(d_\psi(R_i))_{i\in\mathbb{N}}$ serait alors infinie et strictement décroissante, ce qui est impossible. $\dashv$

Cette propriété est, pour les préordres, l'analogue de l'axiome de bonne fondation: on peut en fait montrer que l'ordre induit par $\preceq$ est bien fondé. Il n'est pas nécessaire de faire ce pas et de travailler sur des classes d'équivalence de renommages. L'optimalité s'obtient en fait relativement simplement, grâce au théorème 4.14, qui utilise comme lemme une propriété intuitivement évidente : on ne modifie pas les bénéfices en renommant une sous-formule φ telle que $p(\varphi) = 1$.

Lemme 4.13 $\forall \varphi \sqsubseteq \psi, \forall \varphi' \sqsubseteq \psi, (\varphi' \not\sqsubseteq \varphi \wedge p(\varphi) = 1) \Rightarrow B(\varphi', \psi[\varphi]) = B(\varphi', \psi)$

Preuve. Si $\varphi \sqsubseteq \varphi'$, alors $p(\varphi'[\varphi]) = p(\varphi')$, car $p(\varphi) = 1$, et $P_{\varphi'}^{\psi[\varphi]} = P_{\varphi'}^{\psi}$, et on a donc

$$\begin{aligned} B(\varphi', \psi[\varphi]) &= P_{\varphi'}^{\psi[\varphi]}(p(\varphi'[\varphi])) - P_{\varphi'}^{\psi[\varphi]}(1) - p(\varphi'[\varphi]) \\ &= P_{\varphi'}^{\psi}(p(\varphi')) - P_{\varphi'}^{\psi}(1) - p(\varphi') \\ &= B(\varphi', \psi) \end{aligned}$$

Sinon, φ et φ' sont disjointes, et donc $p(\varphi'[\varphi]) = p(\varphi')$. On a également $P_{\varphi'}^{\psi[\varphi]} = P_{\varphi'}^{\psi}$, puisque $\forall \varphi'', p(\psi[\varphi][\varphi' \leftarrow \varphi'']) = p(\psi[\varphi' \leftarrow \varphi''])$ pour la même raison que ci-dessus, à savoir que $p(\varphi) = 1$. On a donc de même $B(\varphi', \psi[\varphi]) = B(\varphi', \psi)$. $\dashv$

L'idée de cette décroissance bien fondée est assez naturelle si l'on considère la relation de saturation descendante ; il est clair qu'un renommage descendant est en quelque sorte un minimum pour $\preceq$. On peut "remonter" les sous-formules renommées qui constituent des exceptions à la condition de saturation. Qui plus est, cette opération peut se faire en préservant l'optimalité et la complétude :

Théorème 4.14 *Soit R un renommage de ψ optimal, complet et non descendant dans ψ, il existe un renommage R' de ψ optimal et complet dans ψ tel que $R' \preceq R$ et $R \not\preceq R'$.*

Preuve. R est optimal, donc libre dans ψ (corollaire 4.3), mais non descendant dans ψ, c'est donc que la condition de saturation n'est pas vérifiée : $\exists \varphi \sqsubseteq \psi / B(\varphi, \psi'[R - SF(\varphi)]) \geq 0$, où $\psi' = Sup_R(\varphi)$ si défini, et $\psi' = \psi$ sinon. On a $\varphi \neq \psi'$, sinon le bénéfice serait négatif, donc $\varphi \notin R$. Par ailleurs, si $Inf_R(\varphi) = \emptyset$, alors $B(\varphi, \psi'[R]) \geq 0$, et donc $B(\varphi, Rnm(R, \psi)) \geq 0$, en contradiction avec la complétude de R dans ψ, donc $Inf_R(\varphi) \neq \emptyset$. Soient $\varphi' \in Inf_R(\varphi)$ et $R' = R \cup \{\varphi\} - \{\varphi'\}$. On a $\varphi' \sqsubset \varphi$, donc $R' \preceq R$ et $R \not\preceq R'$.

Comme $B(\varphi, \psi'[R - SF(\varphi)]) \geq 0$, alors $a_\varphi^{\psi'[R]} > 1$. Mais $B(\varphi, \psi'[R]) < 0$, car R est complet, donc $p(\varphi[R]) = 1$, et donc $P_{\varphi'}^{\varphi[R]} = \lambda x.x$. Or

$$\begin{aligned} & p(Rnm(R, \psi)) - p(Rnm(R', \psi)) \\ = \; & p(\psi'[R]) - p(\psi'[R']) + p(\varphi'[R]) - p(\varphi[R']) \\ = \; & P_\varphi^{\psi'[R]}(P_{\varphi'}^{\varphi[R]}(1)) - P_\varphi^{\psi'[R]}(1) + p(\varphi'[R]) - P_{\varphi'}^{\varphi[R]}(p(\varphi'[R])) \\ & \text{car } Inf_R(\varphi') = Inf_{R'}(\varphi'),\ P_\varphi^{\psi'[R']} = P_\varphi^{\psi'[R]} \text{ et } P_{\varphi'}^{\varphi[R']} = P_{\varphi'}^{\varphi[R]} \end{aligned}$$

On en conclut que $p(Rnm(R, \psi)) = p(Rnm(R', \psi))$, donc R' est optimal dans ψ.

Il reste à montrer que R' est complet : $\forall \xi \sqsubseteq \psi$, soit $e = B(\xi, Rnm(R', \psi))$, on distingue trois cas :

1. $Sup_{R'}(\xi) \notin \{\varphi, \psi'\}$ si $Sup_{R'}(\varphi)$ existe, et $\psi \neq \psi'$ sinon. Si $Sup_R(\xi)$ existe, alors $Sup_R(\xi) = Sup_{R'}(\xi)$ (qu'on note ψ''), sinon $Sup_{R'}(\xi)$ n'existe pas non plus, et on note $\psi'' = \psi$. On a $\psi''[R] = \psi''[R']$, donc $e = B(\xi, Rnm(R, \psi)) < 0$ car R est complet dans ψ.

2. $Sup_{R'}(\xi) = \psi'$ s'il existe, et $\psi = \psi'$ sinon. On a $e = B(\xi, \psi'[R']) = B(\xi, \psi'[R \cup \{\varphi\}])$. Mais $\xi[R] \not\sqsubseteq \varphi[R]$ (sinon on aurait $Sup_{R'}(\xi) = \varphi$), et $p(\varphi[R]) = 1$, donc d'après le lemme 4.13 on a $e = B(\xi, \psi'[R])$. Mais $Sup_R(\xi) = \psi'$ si $Sup_{R'}(\xi)$ existe, et $\psi' = \psi$ sinon, donc $e = B(\xi, Rnm(R, \psi)) < 0$.

3. $Sup_{R'}(\xi) = \varphi$, donc $e = B(\xi, \varphi[R'])$. On a deux cas :

   (a) $Sup_R(\xi) = \varphi'$, alors

$$\begin{aligned} e &= p(\varphi[R']) - p(\varphi[R' \cup \{\xi\}]) - p(\xi[R']) \\ &= P_{\varphi'}^{\varphi[R']}(p(\varphi'[R'])) - P_{\varphi'}^{\varphi[R']}(p(\varphi'[R' \cup \{\xi\}])) - p(\xi[R']) \end{aligned}$$

$$
\begin{aligned}
&= p(\varphi'[R']) - p(\varphi'[R' \cup \{\xi\}]) - p(\xi[R']) \quad \text{car } P_{\varphi'}^{\varphi[R']} = P_{\varphi'}^{\varphi[R]} = \lambda x.x \\
&= B(\xi, \varphi'[R']) \\
&= B(\xi, \varphi'[R]) \qquad \text{car } Inf_{R'}(\xi) = Inf_R(\xi) \\
&= B(\xi, Rnm(R, \psi)) \\
&< 0
\end{aligned}
$$

(b) $Sup_R(\xi) = \psi'$. Si $\varphi' \sqsubseteq \xi$, on a $P_\xi^{\varphi[R]} = \lambda x.x$, et $e = B(\xi, \varphi[R])$, donc $e < 0$. Par contre, si $\varphi' \not\sqsubseteq \xi$, on a $\xi[R'] = \xi[R] \sqsubseteq \varphi[R]$, or $p(\varphi[R]) = 1$, donc $p(\xi[R']) = 1$, et donc $e < 0$.

⊣

Il faut encore assurer la base de l'induction nœtherienne, qui s'obtient elle-même par un argument du même type :

Lemme 4.15 *Il existe un renommage de ψ complet et optimal dans ψ.*

Preuve. Il existe évidemment un renommage R optimal dans ψ. Si R n'est pas complet dans ψ, alors $\exists \varphi \sqsubseteq \psi / B(\varphi, Rnm(R, \psi)) \geq 0$. Donc $\varphi \notin R$, et comme R est optimal, on a $B(\varphi, Rnm(R, \psi)) = 0$, et $R \cup \{\varphi\}$ est donc optimal. Comme $SF(\psi)$ est fini, l'ordre $\supset$ sur $SF(\psi)$ est bien fondé, et on a $R \cup \{\varphi\} \supset R$. Il existe donc un renommage complet et optimal de ψ. ⊣

Il ne reste dès lors plus qu'à conclure :

Théorème 4.16 *Il existe un renommage de ψ descendant et optimal dans ψ.*

Preuve. D'après le lemme 4.15, il existe un renommage R_0 de ψ complet et optimal dans ψ. A partir de tout renommage R_i de ψ complet, optimal et non descendant dans ψ, on peut d'après le théorème 4.14 construire un renommage R_{i+1} de ψ complet et optimal dans ψ tel que $R_{i+1} \preceq R_i \wedge R_i \not\preceq R_{i+1}$. On construit donc une suite de renommages complets et optimaux dans ψ. Si $\forall i, R_i$ n'est pas descendant dans ψ, alors cette suite est infinie, ce qui est en contradiction avec le théorème 4.12. Donc $\exists i$ tel que R_i est descendant dans ψ, et bien entendu R_i est optimal dans ψ. ⊣

Corollaire 4.17 *Tout renommage de ψ descendant dans ψ est optimal dans ψ.*

Preuve. Il existe un renommage R descendant et optimal dans ψ. Soit R' un renommage descendant dans ψ quelconque, on a d'après le théorème 4.11 $p(Rnm(R, \psi)) = p(Rnm(R', \psi))$, donc R' est optimal dans ψ. ⊣

Chapitre 5

Minimisation du nombre d'occurrences de littéraux

Tel qu'au fil des glaïeuls le vol des libellules

Rimbaud

5.1 Calcul du nombre de littéraux

La propriété d'optimalité démontrée précédemment ne concerne que le nombre de clauses : la transformation sous forme clausale correspondante étant quadratique en taille, elle ne peut évidemment pas être optimale en taille, puisque la transformation structure-preserving est linéaire. Celle-ci n'est cependant pas optimale non plus ; de nombreuses formules admettent une forme clausale plus courte avec la transformation précédente. On peut alors se demander s'il est possible de calculer un renommage qui permette d'obtenir une forme clausale toujours plus courte que celle obtenue avec le renommage *struct_pres*, et qui serait donc linéaire en taille

dans le calcul propositionnel. Nous imposons bien entendu que ce renommage soit calculable en temps polynômial, car la solution à ce problème est évidente et inutile en dehors de cette contrainte.

Il nous est donc interdit de calculer effectivement une forme clausale avant renommage, ce qui rend difficile le calcul de la longueur d'une forme clausale. En particulier, dans le premier ordre, c'est la skolemisation qui pose problème : la longueur dépend du nombre d'occurrences des variables skolemisables, et celui-ci est modifié par les renommages. Cela implique une structure de données complexe, et une formalisation relativement lourde. Nous nous limiterons donc à un critère qui permet comme pour le nombre de clauses un traitement purement arithmétique, et qui sera suffisant dans le calcul propositionnel : *le nombre de littéraux.*

Définition. On note $q(\psi)$ le nombre d'occurrences de littéraux dans $clausale(\psi)$, et $\overline{q}(\psi) = q(\neg\psi)$. $\diamond$

De même que pour le nombre de clauses, la fonction q est purement additive sur les conjonctions, ce qui est évident. Par contre, dans le cas des disjonctions, on ne peut calculer $q(\psi_1 \vee .. \vee \psi_n)$ uniquement en fonction des $q(\psi_i)$; il faut également tenir compte des $p(\psi_i)$. Les formules de la table 5.1 s'établissent trivialement à partir de la formule pour la disjonction, que nous justifions d'abord pour $n = 2$: soient $fnc(\psi_1) = \bigwedge_{i=1}^{n_1} C_i^1$ avec $n_1 = p(\psi_1)$, et $fnc(\psi_2) = \bigwedge_{j=1}^{n_2} C_j^2$ avec $n_2 = p(\psi_2)$. On a $fnc(\psi_1 \vee \psi_2) = \bigwedge_{i=1}^{n_1} \bigwedge_{j=1}^{n_2} C_i^1 \vee C_j^2$, donc :

$$\begin{aligned}
q(\psi_1 \vee \psi_2) &= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} q(C_i^1 \vee C_j^2) \\
&= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} q(C_i^1) + q(C_j^2) \text{ (car } C_i^1 \vee C_j^2 \text{ est une clause)} \\
&= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} q(C_i^1) + \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} q(C_j^2) \\
&= \sum_{i=1}^{n_1} n_2 q(C_i^1) + n_1 \sum_{j=1}^{n_2} q(C_j^2) \\
&= p(\psi_2)q(\psi_1) + p(\psi_1)q(\psi_2)
\end{aligned}$$

La formule générale se déduit par induction : supposons qu'elle soit correcte pour $\psi = \psi_1 \vee .. \vee \psi_n$; i.e. $q(\psi_1 \vee .. \vee \psi_n) = \prod_{i=1}^{n} p(\psi_i) \sum_{i=1}^{n} \frac{q(\psi_i)}{p(\psi_i)}$, soit $\psi = \psi_1 \vee .. \vee \psi_{n+1}$, on a alors :

$$q(\psi) = q((\psi_1 \vee .. \vee \psi_n) \vee \psi_{n+1})$$

Table 5.1: Nombre de littéraux

ψ	$q(\psi)$	$\overline{q}(\psi)$
$\psi_1 \wedge .. \wedge \psi_n$	$\sum_{i=1}^{n} q(\psi_i)$	$\overline{p}(\psi) \sum_{i=1}^{n} \frac{\overline{q}(\psi_i)}{\overline{p}(\varphi_i)}$
$\psi_1 \vee .. \vee \psi_n$	$p(\psi) \sum_{i=1}^{n} \frac{q(\psi_i)}{p(\psi_i)}$	$\sum_{i=1}^{n} \overline{q}(\psi_i)$
$\psi_1 \Rightarrow \psi_2$	$\overline{q}(\psi_1)p(\psi_2) + q(\psi_2)\overline{p}(\psi_1)$	$q(\psi_1) + \overline{q}(\psi_2)$
$\psi_1 \Leftrightarrow \psi_2$	$q(\psi_1)\overline{p}(\psi_2) + q(\psi_2)\overline{p}(\psi_1)$ $+\overline{q}(\psi_1)p(\psi_2) + \overline{q}(\psi_2)p(\psi_1)$	$q(\psi_1)p(\psi_2) + q(\psi_2)p(\psi_1)$ $+\overline{q}(\psi_1)\overline{p}(\psi_2) + \overline{q}(\psi_2)\overline{p}(\psi_1)$
$Qx_1..x_n.\psi'$	$q(\psi')$	$\overline{q}(\psi')$
$\neg\psi'$	$\overline{q}(\psi')$	$q(\psi')$
atomique	1	1

Table 5.2: Coefficients

φ	$a^{\psi}_{\varphi_i}$	$b^{\psi}_{\varphi_i}$
$\varphi_1 \wedge .. \wedge \varphi_n$	a^{ψ}_{φ}	$(b^{\psi}_{\varphi} + d^{\psi}_{\varphi} \sum_{j\neq i} \frac{\overline{q}(\varphi_j)}{\overline{p}(\varphi_j)}) \prod_{j\neq i} \overline{p}(\varphi_j)$
$\varphi_1 \vee .. \vee \varphi_n$	$(a^{\psi}_{\varphi} + c^{\psi}_{\varphi} \sum_{j\neq i} \frac{q(\varphi_j)}{p(\varphi_j)}) \prod_{j\neq i} p(\varphi_j)$	b^{ψ}_{φ}
$\varphi_1 \Rightarrow \varphi_2$ et $i = 1$	b^{ψ}_{φ}	$a^{\psi}_{\varphi}p(\varphi_2) + c^{\psi}_{\varphi}q(\varphi_2)$
$\varphi_1 \Rightarrow \varphi_2$ et $i = 2$	$a^{\psi}_{\varphi}\overline{p}(\varphi_1) + c^{\psi}_{\varphi}\overline{q}(\varphi_1)$	b^{ψ}_{φ}
$\varphi_1 \Leftrightarrow \varphi_2, j \neq i$	$a^{\psi}_{\varphi}\overline{p}(\varphi_j) + b^{\psi}_{\varphi}p(\varphi_j)$ $+c^{\psi}_{\varphi}\overline{q}(\varphi_j) + d^{\psi}_{\varphi}q(\varphi_j)$	$a^{\psi}_{\varphi}p(\varphi_j) + b^{\psi}_{\varphi}\overline{p}(\varphi_j)$ $+c^{\psi}_{\varphi}q(\varphi_j) + d^{\psi}_{\varphi}\overline{q}(\varphi_j)$
$\neg\varphi_1$	b^{ψ}_{φ}	a^{ψ}_{φ}
$Qx_1..x_n.\varphi_1$	a^{ψ}_{φ}	b^{ψ}_{φ}

φ	$c^{\psi}_{\varphi_i}$	$d^{\psi}_{\varphi_i}$
$\varphi_1 \wedge .. \wedge \varphi_n$	c^{ψ}_{φ}	$d^{\psi}_{\varphi} \prod_{j\neq i} \overline{p}(\varphi_j)$
$\varphi_1 \vee .. \vee \varphi_n$	$c^{\psi}_{\varphi} \prod_{j\neq i} p(\varphi_j)$	d^{ψ}_{φ}
$\varphi_1 \Rightarrow \varphi_2$ et $i = 1$	d^{ψ}_{φ}	$c^{\psi}_{\varphi}p(\varphi_2)$
$\varphi_1 \Rightarrow \varphi_2$ et $i = 2$	$c^{\psi}_{\varphi}\overline{p}(\varphi_1)$	d^{ψ}_{φ}
$\varphi_1 \Leftrightarrow \varphi_2, j \neq i$	$c^{\psi}_{\varphi}\overline{p}(\varphi_j) + d^{\psi}_{\varphi}p(\varphi_j)$	$c^{\psi}_{\varphi}p(\varphi_j) + d^{\psi}_{\varphi}\overline{p}(\varphi_j)$
$\neg\varphi_1$	d^{ψ}_{φ}	c^{ψ}_{φ}
$Qx_1..x_n.\varphi_1$	c^{ψ}_{φ}	d^{ψ}_{φ}

$$
\begin{aligned}
&= p(\psi_1 \vee .. \vee \psi_n)q(\psi_{n+1}) + p(\psi_{n+1})q(\psi_1 \vee .. \vee \psi_n) \\
&= \frac{p(\psi)q(\psi_{n+1})}{p(\psi_{n+1})} + p(\psi)\sum_{i=1}^{n}\frac{q(\psi_i)}{p(\psi_i)} \\
&= p(\psi)\sum_{i=1}^{n+1}\frac{q(\psi_i)}{p(\psi_i)}
\end{aligned}
$$

On peut alors construire pour la fonction q des notions similaires à celles définies précédemment pour p. Il faut cependant, en ce qui concerne les polynômes P_φ^ψ modifier la définition pour tenir compte du fait que $q(\psi)$ dépend des $q(\varphi), \overline{q}(\varphi)$ ainsi que des $p(\varphi), \overline{p}(\varphi)$ des sous-formules $\varphi \sqsubseteq \psi$.

Définition. $\forall \psi, \forall \varphi \sqsubseteq \psi$, on note Q_φ^ψ (resp. $\overline{Q}_\varphi^\psi$) la fonction de $(I\!N^*)^4$ dans $I\!N^*$ définie par $\forall \varphi', Q_\varphi^\psi(p(\varphi'), \overline{p}(\varphi'), q(\varphi'), \overline{q}(\varphi')) = q(\psi[\varphi \leftarrow \varphi'])$ (resp. $\overline{Q}_\varphi^\psi(p(\varphi'), \overline{p}(\varphi'), q(\varphi'), \overline{q}(\varphi')) = \overline{q}(\psi[\varphi \leftarrow \varphi'])$). $\diamond$

De même qu'au chapitre 2, on peut montrer que $Q_\varphi^\psi(x, y, u, v)$ est un polynôme de la forme $a_\varphi^\psi x + b_\varphi^\psi y + c_\varphi^\psi u + d_\varphi^\psi v + e$. Ces coefficients peuvent être calculés en fonction de φ et ψ à partir de la table 5.2 (avec bien sûr $a_\psi^\psi = b_\psi^\psi = d_\psi^\psi = 0$ et $c_\psi^\psi = 1$). Pour établir ces formules, nous avons besoin des tables 5.1 et 2.1.

Si $\varphi = \varphi_1 \vee .. \vee \varphi_n$, alors :

$$
\begin{aligned}
Q_{\varphi_i}^\psi(x, y, u, v) &= Q_\varphi^\psi(P_\varphi^\psi(x, y), \overline{P}_\varphi^\psi(x, y), Q_\varphi^\psi(x, y, u, v), \overline{Q}_\varphi^\psi(x, y, u, v)) \\
&= a_\varphi^\psi x \prod_{j \neq i} p(\varphi_j) + b_\varphi^\psi (y + \sum_{j \neq i} \overline{p}(\varphi_j)) \\
&\quad + c_\varphi^\psi (x \prod_{j \neq i} p(\varphi_j) \sum_{j \neq i} \frac{q(\varphi_j)}{p(\varphi_j)} + u \prod_{j \neq i} p(\varphi_j)) + d_\varphi^\psi (v + \sum_{j \neq i} \overline{q}(\varphi_j)) + e \\
&= (a_\varphi^\psi + c_\varphi^\psi \sum_{j \neq i} \frac{q(\varphi_j)}{p(\varphi_j)}) \prod_{j \neq i} p(\varphi_j) x + b_\varphi^\psi y + c_\varphi^\psi \prod_{j \neq i} p(\varphi_j) u + d_\varphi^\psi v + e'
\end{aligned}
$$

où e et e' sont indépendants de x, y, u et v.

Les cas $\varphi = \varphi_1 \wedge .. \wedge \varphi_n, \varphi = \varphi_1 \Rightarrow \varphi_2$ sont similaires. Nous justifions également les formules pour $\varphi = \varphi_1 \Leftrightarrow \varphi_2, i = 1$:

$$
\begin{aligned}
Q_{\varphi_1}^\psi(x, y, u, v) &= a_\varphi^\psi(\overline{p}(\varphi_2)x + p(\varphi_2)y) + b_\varphi^\psi(p(\varphi_2)x + \overline{p}(\varphi_2)y) \\
&\quad + c_\varphi^\psi(\overline{p}(\varphi_2)u + q(\varphi_2)y + p(\varphi_2)v + \overline{q}(\varphi_2)x) \\
&\quad + d_\varphi^\psi(p(\varphi_2)u + q(\varphi_2)x + \overline{p}(\varphi_2)v + \overline{q}(\varphi_2)y) + e \\
&= (a_\varphi^\psi \overline{p}(\varphi_2) + b_\varphi^\psi p(\varphi_2) + c_\varphi^\psi \overline{q}(\varphi_2) + d_\varphi^\psi q(\varphi_2))x \\
&\quad + (a_\varphi^\psi p(\varphi_2) + b_\varphi^\psi \overline{p}(\varphi_2) + c_\varphi^\psi q(\varphi_2) + d_\varphi^\psi \overline{q}(\varphi_2))y \\
&\quad + (c_\varphi^\psi \overline{p}(\varphi_2) + d_\varphi^\psi p(\varphi_2))u + (c_\varphi^\psi p(\varphi_2) + d_\varphi^\psi \overline{p}(\varphi_2))v + e
\end{aligned}
$$

Nous n'avons pas ici à établir de nouveaux résultats de complexité : ceux qui ont été établis au chapitre 2 nous suffisent pour montrer que la complexité dans le pire des cas de $q(\psi)$ est $O(|SF(\psi)|2^{|SF(\psi)|})$: il suffit en effet de ne considérer que le cas propositionnel, et de constater qu'alors $\forall\psi, \frac{|fnc(\psi)|}{3} \leq q(\psi) \leq |fnc(\psi)|$. La deuxième inégalité est évidente, et la première provient de $|fnc(\psi)| \leq 3q(\psi)$, puisque dans $fnc(\psi)$ il y a exactement $q(\psi)$ symboles propositionnels, $q(\psi) - p(\psi)$ symboles de disjonction, $p(\psi) - 1$ symboles de conjonction et au plus $q(\psi)$ symboles de négations.

Nous obtenons de la même façon que le nombre de littéraux est linéaire après le renommage structure-preserving :

$$\forall\psi, q(Rnm(struct\_pres(\psi), \psi)) \leq O(|SF(Rnm(struct\_pres(\psi), \psi))|) \leq O(|SF(\psi)|)$$

En fait, toutes les majorations de complexité valides dans le calcul propositionnel et concernant la longueur d'une formule sont également valides pour le nombre de littéraux des formules du premier ordre. Nous n'utiliserons cependant que les majorations que nous venons d'établir.

5.2 Propriétés

Nous allons dans la suite tenter de transposer les résultats obtenus pour le renommage par minimisation de p au renommage par minimisation de q. Cela ne se limite pas toujours à remplacer p par q dans les définitions, théorèmes et preuves énoncés aux chapitres 3 et 4. Nous utiliserons généralement pour q les mêmes notations que pour p, primées si nécessaire. Par exemple, $B'(R, \psi) = q(\psi) - q(Rnm(R, \psi))$ est le *bénéfice en nombre de littéraux* de R dans ψ. Les définitions de renommages *q-décroissants*, R'_{inf} (et la procédure R'_{inf}), *q-complets*, *q-optimaux*, *q-libres* et *q-descendants* se déduisent trivialement des définitions correspondantes pour p.

L'une des principales différences entre les propriétés de p et celles de q concerne l'expression du bénéfice : nous avions une expression relativement simple pour le bénéfice en nombre de clauses, du moins dans la cas linéaire, et on obtenait alors $B(\varphi, \psi) < 0 \Leftrightarrow a^{\psi}_{\varphi} = 1 \vee p(\varphi) = 1$. Ici, on a dans le cas linéaire (et polarité positive), $B'(\varphi, \psi) = (a^{\psi}_{\varphi} - 1)(p(\varphi) - 1) + (c^{\psi}_{\varphi} - 1)(q(\varphi) - 1) - 2$, et nous n'avons plus une caractérisation aussi simple pour $B'(\varphi, \psi) < 0$. C'est l'un des principaux points sur lequel devra porter notre attention au cours de la transposition des preuves.

5.2.1 Le théorème fondamental de monotonie

Nous devons bien entendu nous intéresser en premier lieu au théorème fondamental de monotonie, dont tous les résultats intéressants découlent (optimalité, complexité). Ce théorème n'est cependant pas vérifié ici dans toute sa généralité. Cela vient du fait qu'on peut par renommage faire *croître* les coefficients du polynôme Q_φ^ψ dans un cas heureusement très particulier : si $pol(\varphi,\psi) = 1$, on a $Q_\varphi^{Rnm(\varphi,\psi)}(x,y,u,v) = x+u+e$, bien que $Q_\varphi^\psi(x,y,u,v) = u+e'$ lorsque φ est un conjoint de ψ. Plus généralement, on étend la définition de conjoint :

Définition. On appelle *conjoint* de ψ toute sous-formule $\varphi \sqsubset \psi$ telle que $a_\varphi^\psi = b_\varphi^\psi = 0$ (ce qui signifie que $q(\psi)$ est indépendant de $p(\varphi)$ et $\overline{p}(\varphi)$). $\diamond$

Il est facile alors de voir qu'on a $c_\varphi^\psi = 1, d_\varphi^\psi = 0$ si $pol(\varphi,\psi) = 1$, et $c_\varphi^\psi = 0, d_\varphi^\psi = 1$ si $pol(\varphi,\psi) = -1$, et bien entendu tout conjoint est de polarité non nulle. On peut alors prouver le théorème fondamental en y apportant la restriction suivante :

Théorème 5.1 *Soient ψ une formule, R et R' deux renommages de ψ tels que $R \subset R'$ et $R - R'$ ne contient pas de conjoint de ψ, alors $\forall \varphi \sqsubset \psi$, non conjoint de $\psi, B'(\varphi, Rnm(R,\psi)) \geq B'(\varphi, Rnm(R',\psi))$.*

Preuve. On montre d'abord que $\forall \varphi' \sqsubset \psi$, non conjoint de $\psi, B'(\varphi,\psi) \geq B'(\varphi, Rnm(\varphi',\psi))$. On a $B'(\varphi,\psi) = Q_\varphi^\psi(p(\varphi),\overline{p}(\varphi),q(\varphi),\overline{q}(\varphi)) - Q_\varphi^\psi(1,1,1,1) - q(Def_\psi(\varphi))$. On procède comme pour le théorème 3.3

1. si φ et φ' sont disjointes, $B'(\varphi, Rnm(\varphi',\psi)) = Q_\varphi^{\psi[\varphi']}(p(\varphi),\overline{p}(\varphi),q(\varphi),\overline{q}(\varphi)) - Q_\varphi^{\psi[\varphi']}(1,1,1,1) - q(Def_\psi(\varphi))$, et les coefficients de $Q_\varphi^{\psi[\varphi']}$ sont inférieurs à ceux de Q_φ^ψ.

2. si $\varphi \sqsubseteq \varphi'$: alors

$$B'(\varphi, Rnm(\varphi',\psi)) = Q_\varphi^{Def_\psi(\varphi')}(p(\varphi),\overline{p}(\varphi),q(\varphi),\overline{q}(\varphi)) - Q_\varphi^{Def_\psi(\varphi')}(1,1,1,1) - q(Def_\psi(\varphi))$$

On a par ailleurs

$$\begin{cases} Q_\varphi^\psi(x,y,u,v) = Q_{\varphi'}^\psi(P_\varphi^{\varphi'}(x,y),\overline{P}_\varphi^{\varphi'}(x,y),Q_\varphi^{\varphi'}(x,y,u,v),\overline{Q}_\varphi^{\varphi'}(x,y,u,v)) \\ Q_\varphi^{Def_\psi(\varphi')}(x,y,u,v) = \\ \qquad Q_{\varphi'}^{Def_\psi(\varphi')}(P_\varphi^{\varphi'}(x,y),\overline{P}_\varphi^{\varphi'}(x,y),Q_\varphi^{\varphi'}(x,y,u,v),\overline{Q}_\varphi^{\varphi'}(x,y,u,v)) \end{cases}$$

et puisque φ' n'est pas un conjoint de ψ, les coefficients de $Q_{\varphi'}^{Def_\psi(\varphi')}$ sont inférieurs à ceux de $Q_{\varphi'}^\psi$.

3. si $\varphi' \sqsubseteq \varphi$: on se ramène au cas précédent.

Enfin, si $\{\varphi_1..\varphi_n\} = R' - R$, on a $\forall i \in \{1..n-1\}, B'(\varphi, Rnm(R \cup \{\varphi_1..\varphi_i\}, \psi)) \geq B'(\varphi, Rnm(R \cup \{\varphi_1..\varphi_{i+1}\}, \psi))$, puisque φ_{i+1} n'est pas un conjoint de $Rnm(\{\varphi_1..\varphi_i\}, \psi)$. On obtient par récurrence que $B'(\varphi, Rnm(R, \psi)) \geq B(\varphi, Rnm(R', \psi))$. $\dashv$

5.2.2 Conséquences

Le problème des conjoints est facilement éliminé en constatant que renommer un conjoint fait *toujours* croître le nombre de littéraux. On peut alors très simplement transposer la preuve du théorème 3.4 :

Théorème 5.2 *Soient ψ une formule et R un renommage de ψ non q-décroissant, il existe un renommage $R' \subset R$, q-décroissant tel que $q(Rnm(R', \psi)) < q(Rnm(R, \psi))$.*

Preuve. On procède comme pour le théorème 3.4, sauf que l'on considère le plus grand i tel que $B'(\varphi_i, Rnm(\{\varphi_1..\varphi_{i-1}\}, \psi)) < 0$. Il n'y a donc pas de conjoint dans $\{\varphi_{i+1}..\varphi_n\}$. Si φ_i est un conjoint, on a $B'(\varphi_i, Rnm(R - \{\varphi_i\}, \psi)) < 0$, sinon on peut appliquer le théorème 5.1 et on obtient également $B'(\varphi_i, Rnm(R - \{\varphi_i\}, \psi)) < 0$. Le reste de la preuve est identique. $\dashv$

Nous pouvons donc également nous restreindre aux renommages q-décroissants, c'est à dire à la procédure R'_{inf}. Nous obtenons aussi le résultat d'optimalité de ces renommages sur leurs sur-renommages :

Théorème 5.3 *$\forall \psi$, soient R et R' des renommages de ψ tels que R est q-complet et $R \subset R'$, alors $q(Rnm(R, \psi)) \leq q(Rnm(R', \psi))$.*

Preuve. Soit $R'' = \{\varphi \in R' / \varphi$ n'est pas un conjoint de $\psi\} \cup R$, on a $R \subset R'' \subset R'$ et $q(Rnm(R'', \psi)) \leq q(Rnm(R', \psi))$. Soit $\{\varphi_1..\varphi_n\} = R'' - R$, on a $\forall i, B'(\varphi_i, Rnm(R, \psi)) < 0$ puisque R est q-complet, mais $R'' - R$ ne contient pas de conjoint de ψ, et φ_i n'en est pas un non plus, donc $B'(\varphi_i, Rnm(R, \psi)) \geq B'(\varphi_i, Rnm(R \cup \{\varphi_1..\varphi_{i-1}\}, \psi))$ d'après le théorème 5.1, donc $B'(\varphi_i, Rnm(R \cup \{\varphi_1..\varphi_{i-1}\}, \psi)) < 0$, et donc $q(Rnm(R \cup \{\varphi_1..\varphi_{i-1}\}, \psi)) < q(Rnm(R \cup \{\varphi_1..\varphi_i\}, \psi))$. Par induction, on en déduit $q(Rnm(R, \psi)) \leq q(Rnm(R'', \psi))$. $\dashv$

Pour établir une comparaison avec le renommage particulier *struct_pres*, nous avons le même problème que précédemment, à savoir que $R'_{inf}(\psi)$ peut contenir

des négations. Tous les autres éléments de $R'_{inf}(\psi)$ sont nécessairement dans $struct\_pres(\psi)$, puisque les formules atomiques ont toujours un bénéfice négatif en nombre de littéraux. Ce problème peut se résoudre exactement de la même façon qu'au chapitre 3, et nous avons :

Lemme 5.4 $\forall\psi, \forall\varphi, \varphi' \sqsubseteq \psi$ *telles que* $SN(\varphi) = SN(\varphi')$, *on a :*

1. $q(\psi[\varphi]) = q(\psi[\varphi'])$
2. $q(Def_\psi(\varphi)) = q(Def_\psi(\varphi'))$
3. $q(Rnm(\varphi, \psi)) = q(Rnm(\varphi', \psi))$
4. $\varphi \neq \varphi' \Rightarrow \varphi \notin R'_{inf}(\psi) \vee \varphi' \notin R'_{inf}(\psi)$

Preuve. Voir la preuve du lemme 3.6, en remplaçant p par q. Pour le point 4, on peut appliquer le théorème fondamental car $R'_{inf}(\psi)$ est q-décroissant, et ne contient donc pas de conjoint de ψ. ⊣

On peut donc prouver que :

Théorème 5.5 $\forall\psi, q(Rnm(R'_{inf}(\psi), \psi)) \leq q(Rnm(struct\_pres(\psi), \psi))$

Preuve. Identique à celle du théorème 3.7, modulo les changements syntaxiques évidents. ⊣

Et on peut en déduire de façon tout à fait évidente la complexité en nombre de littéraux :

Corollaire 5.6 $q(Rnm(R'_{inf}(\psi), \psi))$ *est borné en* $O(|SF(\psi)|)$.

Preuve. Il est évident d'après le théorème 3.2 que $q(Rnm(struct\_pres(\psi), \psi))$ est borné en $O(|SF(\psi)|)$, et il ne reste plus qu'à appliquer le théorème 5.5. ⊣

5.2.3 complexité en taille

Ce résultat doit enfin nous être utile pour évaluer la complexité de la longueur de la forme clausale obtenue. D'après les remarques précédentes, il est évident que cette longueur est linéaire dans le cas propositionnel. En logique du premier ordre, si le nombre de littéraux est linéaire, il nous faut également nous intéresser à la longueur de ces littéraux, d'où la définition suivante :

Définition. On note $h'(\psi)$ la longueur de la plus grande sous-formule atomique de ψ. $\diamond$

Contrairement à la définition de h, nous ne considérons pas ici cette quantité pour une forme clausale ; la raison en est que h' est invariant par les transformations *fnc*, *prénexe* et *linéaire* :

Lemme 5.7 $\forall\psi$, *on a :*

1. $h'(linéaire(\psi)) = h'(\psi)$.
2. $h'(prénexe(\psi)) = h'(\psi)$.
3. $h'(fnc(\psi)) = h'(\psi)$
4. $h'(Sklm(\psi)) = (1 + V(\psi))h'(\psi)$.

Preuve. Les points 1, 2 et 3 sont évidents. Le point 4 est similaire au premier point de la preuve du théorème 2.2. $\dashv$

Par ailleurs, le lemme 3.10 est évidemment valide pour le renommage R'_{inf}, comme le montre le commentaire suivant la preuve de ce lemme, et nous pouvons donc facilement transposer la preuve du théorème 3.11 :

Théorème 5.8

$\forall\psi, |clausale(Rnm(R'_{inf}(\psi), \psi))|$ *est borné par* $O((1 + V(\psi))h'(\psi)|SF(\psi)|)$.

Preuve. Soient $\psi' = Rnm(R'_{inf}(\psi), \psi)$ et $\psi'' = prénexe(Sklm(linéaire(\psi')))$, on a :

$$\begin{aligned} |fnc(\psi'')| &\leq q(\psi'')(1 + h'(\psi'')) + 1 + V(\psi'') \\ &\leq O(|SF(\psi)|(1 + V(\psi))h'(\psi) + |\psi''|) \end{aligned}$$

en vertu de ce qui précède. D'après la preuve du théorème 3.11, et comme le seul résultat utilisé pour borner $|\psi''|$ qui dépend du renommage effectué pour obtenir ψ'' est le lemme 3.10, et que celui-ci est également vrai pour R'_{inf}, on a $|\psi''| \leq O((1 + V(\psi))|\psi|)$.

Par ailleurs, on a évidemment $|\psi| \leq h'(\psi)|SF(\psi)|$, ce qui nous donne la borne $O((1 + V(\psi))h'(\psi)|SF(\psi)|)$ pour $|clausale(\psi')|$. $\dashv$

On peut cependant se demander si cette majoration pourrait être améliorée : la complexité dans le calcul propositionnel ($V(\psi) = 0$ et $h'(\psi) = 1$) étant identique à celle de la transformation structure-preserving, pourquoi ne serait-ce pas le cas

dans le premier ordre? L'exemple suivant montre que nous avons en fait obtenu la complexité exacte de cette transformation (dans le pire des cas) :
considérons la formule

$$\psi_{n,k,l} = \forall x_1..x_n.\exists y.[P(x_1..x_n \overbrace{y..y}^{k \text{ fois}}) \vee (Q_1 \wedge .. \wedge Q_l)]$$

dont la longueur est en $O(n+k+l)$. Il est facile de voir qu'on a $R'_{inf}(\psi_{n,k,l}) = \emptyset$, et on obtient donc pour forme clausale :

$$\forall x_1..x_n. \quad [(P(x_1..x_n \overbrace{f(x_1..x_n)..f(x_1..x_n)}^{k \text{ fois}}) \vee Q_1)$$
$$\ldots$$
$$(P(x_1..x_n f(x_1..x_n)..f(x_1..x_n)) \vee Q_l)]$$

dont la longueur est en $O(nkl)$, et on a $V(\psi_{n,k,l}) = O(n)$, $h'(\psi_{n,k,l}) = O(n+k)$ et $|SF(\psi_{n,k,l})| = O(l)$.

Si donc nous obtenons dans le calcul propositionnel une plus grande concision avec le renommage R'_{inf} qu'avec le renommage structure preserving, cela n'est pas toujours le cas en logique du premier ordre ; ce n'est alors que sur le nombre de littéraux que porte l'amélioration, ce qui semble cependant constituer un résultat important.

Il faut noter ici qu'il est relativement facile d'éviter ce problème : il suffit en effet de remplacer le critère que nous avons adopté ici, le nombre d'occurrences de littéraux, par la longueur hors skolemisation de la forme clausale. Plus précisément, il s'agit de la longueur de $fnc(prénexe(linéaire(Rnm(R,\psi))))$, sans toutefois compter les symboles de conjonction, disjonction et négation. On constate que cette quantité se calcule comme la précédente, en suivant la table 5.1, sauf évidemment en ce qui concerne les formules atomiques. En conséquence, les polynômes sont les mêmes, et la table 5.2 ne subit aucune modification. On peut montrer — mais nous ne le ferons pas ici — qu'en tentant de minimiser cette quantité, on obtient alors la même complexité que la transformation structure preserving.

La question se pose alors de savoir si l'option consistant à minimiser la longueur totale, avec skolemisation, de la forme clausale — ce qui, comme nous l'avons déjà dit, impose un traitement plus complexe — permettrait d'améliorer cette complexité. La réponse est évidemment négative, comme le montre l'exemple suivant : $\psi = \forall x_1..x_n.\exists y_1..y_n.P(x_1..x_n y_1..y_n)$. Le renommage ne permettant de supprimer ni quantificateur ni variable (seulement des occurrences de variables, mais pas toutes), toutes les formes clausales que nous pourrions obtenir auraient une longueur en

$O(n^2)$, alors que $|\psi|$ est en $O(n)$. On ne saurait donc faire mieux, en ce qui concerne la complexité en taille dans le pire des cas, que la transformation *struct_pres*.

5.3 Les renommages q-descendants

Nous pouvons maintenant nous intéresser aux renommages q-descendants, et tenter de transposer le résultat d'optimalité. Avec un exemple très proche de celui présenté au début du chapitre 4, nous pouvons déjà montrer que l'optimalité n'est pas vérifiée en présence d'équivalence :

soit $\psi = \varphi \Leftrightarrow (A \wedge B)$ avec $\varphi = \varphi' \vee A'$, nous avons alors :

$$\begin{cases} B'(\varphi,\psi) = p(\varphi') + \overline{p}(\varphi') + \overline{q}(\varphi') - 5 & B'(\varphi', Rnm(\varphi,\psi)) = p(\varphi') - 5 \\ B'(\varphi',\psi) = 2p(\varphi') + \overline{p}(\varphi') + \overline{q}(\varphi') - 8 & B'(\varphi, Rnm(\varphi',\psi)) = -2 \end{cases}$$

si on choisit pour φ' une conjonction de 6 littéraux, on a $p(\varphi') = q(\varphi') = \overline{q}(\varphi') = 6$ et $\overline{p}(\varphi') = 1$. Si on utilise la stratégie descendante en profondeur d'abord et gauche-droite, on renomme d'abord φ, puisque $B'(\varphi,\psi) = 8$, puis on renomme φ' : $B'(\varphi', Rnm(\varphi,\psi)) = 1$, et on ne renomme aucune autre sous-formule. On obtient donc le renommage $\{\varphi, \varphi'\}$, de bénéfice 9. Cependant, le renommage $\{\varphi'\}$, qui n'est pas q-descendant, admet un bénéfice de 11 littéraux, ce qui prouve que $\{\varphi, \varphi'\}$ n'est pas optimal.

Comme précédemment, nous nous limitons donc aux formules linéaires. Il serait alors possible de démontrer l'existence d'un renommage q-descendant optimal, en transposant la preuve du chapitre 4. Cela nous serait cependant inutile, car l'exemple suivant montre qu'il n'y a pas invariance du nombre de littéraux des renommages q-descendants.

Soit $\psi = \varphi \vee A \vee \varphi'$, avec $\varphi = P \wedge Q \wedge R$ et $\varphi' = P' \wedge Q'$. On a :

$$\begin{cases} B'(\varphi,\psi) = 6 & B'(\varphi', Rnm(\varphi,\psi)) = -1 \\ B'(\varphi',\psi) = 5 & B'(\varphi, Rnm(\varphi',\psi)) = 0 \end{cases}$$

donc, si on renomme de gauche à droite, on ne renomme pas φ', et le bénéfice est 6, alors que si on renomme de droite à gauche, on renomme φ', et le bénéfice n'est plus que de 5 littéraux.

On peut donc en conclure qu'il n'y a pas optimalité des renommages q-descendants dans leur ensemble. Sans doute est-il possible de montrer l'optimalité d'une stratégie q-descendante particulière (par exemple : considérer en dernier les sous-formules φ telles que $p(\varphi) = 2$), mais certainement pas par transposition des

résultats du chapitre 4, car il faudrait développer de nouveaux outils pour une classe spécifique de renommages q-descendants, ce que nous ne ferons donc pas ici.

Chapitre 6

Algorithmes de renommage

Comme la vie est lente
Et comme l'Espérance est violente.

Apollinaire

6.1 Un algorithme pour R_{inf}

Nous avons déjà mentionné, à la fin de l'introduction du chapitre 3, l'une des propriétés les plus importantes que nous imposons à nos renommages : que ceux-ci soient calculables efficacement, c'est à dire — au moins — en temps polynômial. C'est l'une des raisons qui nous ont fait préférer des critères de renommages relativement simples ; le nombre de clauses et le nombre de littéraux. On est en effet certain de pouvoir calculer ceux-ci en temps polynômial.

Cependant, il faut également calculer les bénéfices pour chaque sous-formule, en tenant compte des renommages qui ont été effectués. S'il est relativement simple de concevoir un algorithme polynômial répondant aux spécifications, cela devient

plus difficile en s'intéressant à l'efficacité réelle de cet algorithme. En particulier, il convient de minimiser le nombre de parcours de la formule, considérée comme un arbre étiqueté. Il est en fait possible de n'effectuer qu'un seul parcours, pour autant que l'on travaille sur une structure de donnée plus riche que la formule de départ.

Nous nous intéressons ici essentiellement au renommage R_{inf}, pour deux raisons. D'abord parce qu'il est relativement simple — nous indiquerons la marche à suivre — d'en déduire un algorithme pour le renommage R'_{inf}. Ensuite et surtout, nous devrons prouver que l'algorithme calcule effectivement un renommage descendant, pour que la propriété d'optimalité du nombre de clauses soit vérifiée. Il n'est pas évident *a priori* qu'avec un parcours descendant on produise un renommage libre — la condition de saturation, quant à elle, se vérifie facilement d'après le théorème fondamental de monotonie.

En premier lieu, nous donnons les différentes fonctions et procédures qui composent cet algorithme. La fonction principale, qui calcule $Rnm(R_{inf}(\psi), \psi)$ est la suivante :

$PR_{inf}(\psi) =$
debut
  $\langle \psi.p, \psi.\overline{p} \rangle$:=*attribution*$(\psi)$;
  $S := \emptyset$;
  $PR_{inf}\_rec(\psi, 1, 0, +1)$;
  retour$(\psi \wedge \bigwedge_{\varphi \in S} Def_{\psi}(\varphi))$
fin;

Le principe de l'algorithme est donc de collecter dans un ensemble S les sous-formules à renommer, c'est à dire de calculer dans S l'ensemble $R_{inf}(\psi)$. Un premier parcours de ψ permet de construire une "formule attribuée", c'est à dire une structure de donnée — implémentée par exemple par une liste — dans laquelle à chaque sous-formule φ sont associés deux attributs $\varphi.p$ et $\varphi.\overline{p}$ qui doivent contenir, on s'en doute, les entiers $p(\varphi)$ et $\overline{p}(\varphi)$. Cette attribution se fait au moyen des fonctions de la page suivante, dont la correction n'est guère douteuse : l'exécution de *attribution*(φ) renvoie la valeur $\langle p(\varphi), \overline{p}(\varphi) \rangle$ — calculée par la fonction *calcul_attributs*, calquée sur le tableau 2.1 — après avoir effectué l'attribution de toutes les sous-formules strictes de φ. Il est également assez simple d'en évaluer la complexité : il y a $|SF(\psi)|$ appels récursifs, et chaque appel effectue le calcul de $p(\varphi), \overline{p}(\varphi)$, dont la complexité

$attribution(\varphi) =$
si φ atomique **alors retour**$(\langle 1,1\rangle)$
sinon soit $\langle \varphi_1..\varphi_n\rangle := SFd(\varphi)$ **dans**
  debut
    pour $i = 1$ **a** n **faire** $\langle \varphi_i.p, \varphi_i.\bar{p}\rangle := attribution(\varphi_i)$;
    retour$(calcul\_attributs(connectif(\varphi), \langle \varphi_1.p..\varphi_n.p\rangle, \langle \varphi_1.\bar{p}..\varphi_n.\bar{p}\rangle))$
  fin;

$calcul\_attributs(c, \langle p_1..p_n\rangle, \langle \bar{p}_1..\bar{p}_n\rangle) =$
cas c **parmis**
  $\wedge$:$\langle \sum_{i=1}^{n} p_i, \prod_{i=1}^{n} \bar{p}_i\rangle$;
  $\vee$:$\langle \prod_{i=1}^{n} p_i, \sum_{i=1}^{n} \bar{p}_i\rangle$;
  $\Rightarrow$:$\langle \bar{p}_1 * p_2, p_1 + \bar{p}_2\rangle$;
  $\Leftrightarrow$:$\langle \bar{p}_1 * p_2 + p_1 * \bar{p}_2, p_1 * p_2 + \bar{p}_1 * \bar{p}_2\rangle$;
  $\neg$:$\langle \bar{p}_1, p_1\rangle$;
  Quantifier:$\langle p_1, \bar{p}_1\rangle$;
fin;

est en $O(\log p(\varphi) + \log \bar{p}(\varphi)) \leq O(|SF(\varphi)|)$ d'après le théorème 2.4. La complexité de la phase d'attribution de ψ, en temps et en espace, est donc $O(|SF(\psi)|^2)$.

La présence des attributs va permettre de mémoriser les valeurs successives $p(\varphi[S]), \bar{p}(\varphi[S])$ selon les valeurs de S. Ces valeurs sont bien entendu nécessaires au calcul des bénéfices, de même que les coefficients a_φ^ψ et b_φ^ψ, qui sont donnés en paramètre de la procédure PR_{inf}_rec, avec la sous-formule φ et sa polarité s dans ψ. Ce dernier paramètre n'est pas absolument nécessaire, car il peut se calculer à partir de a_φ^ψ et b_φ^ψ, mais cela nuirait à la clarté de l'algorithme.

Bien que nous ne prouverons pas formellement la correction de cet algorithme, nous tenons à en indiquer ici les principaux arguments dans la mesure où ils contribuent à une compréhension claire de son fonctionnement, ce qui devrait suffire à convaincre.

A chaque appel $PR_{inf}\_rec(\varphi, a, b, s)$, on a $s = pol(\varphi, \psi)$, $a = a_\varphi^{Rnm(S,\psi)}$ et $b = b_\varphi^{Rnm(S,\psi)}$. Si $\varphi = \psi$, on a donc $s = 1, a = 1$ et $b = 0$ puisque $P_\psi^\psi(x, y) = x$. S ne contient encore aucune sous-formule de φ, et les attributs à l'intérieur de φ

$PR_{inf}\_rec(\varphi, a, b, s) =$
si $\langle \varphi.p, \varphi.\overline{p} \rangle \neq \langle 1, 1 \rangle$ **alors**
si $a * \varphi.p + b * \varphi.\overline{p} - a - b \geq nb\_def(\varphi.p, \varphi.\overline{p}, s)$ **alors**
debut
$S := S \cup \{\varphi\}$;
$PR_{inf}\_rec(\varphi,$ **si** $s \geq 0$ **alors** 1 **sinon** 0, **si** $s \leq 0$ **alors** 1 **sinon** $0, s)$;
$\langle \varphi.p, \varphi.\overline{p} \rangle := \langle 1, 1 \rangle$
fin
sinon soit $\langle \varphi_1..\varphi_n \rangle := SFd(\varphi)$ **dans**
debut
cas $connectif(\varphi)$ **parmis**
$\wedge$:**debut** $\Pi := \prod_{i=1}^{n} \varphi_i.\overline{p}$;
pour $i = 1$ **a** n **faire**
debut $P := \Pi/\varphi_i.\overline{p}$;
$PR_{inf}\_rec(\varphi_i, a, b * P, s)$;
$\Pi := P * \varphi_i.\overline{p}$
fin
fin;
$\vee$:**debut** $\Pi := \prod_{i=1}^{n} \varphi_i.p$;
pour $i = 1$ **a** n **faire**
debut $P := \Pi/\varphi_i.p$;
$PR_{inf}\_rec(\varphi_i, a * P, b, s)$;
$\Pi := P * \varphi_i.p$
fin
fin;
$\Rightarrow$:**debut** $PR_{inf}\_rec(\varphi_1, b, a * \varphi_2.p, -s)$;
$PR_{inf}\_rec(\varphi_2, a * \varphi_1.\overline{p}, b, s)$
fin;
$\Leftrightarrow$:**debut** $PR_{inf}\_rec(\varphi_1, a * \varphi_2.\overline{p} + b * \varphi_2.p, a * \varphi_2.p + b * \varphi_2.\overline{p}, 0)$;
$PR_{inf}\_rec(\varphi_2, a * \varphi_1.\overline{p} + b * \varphi_1.p, a * \varphi_1.p + b * \varphi_1.\overline{p}, 0)$
fin;
$\neg$:$PR_{inf}\_rec(\varphi_1, b, a, -s)$;
Quantifier:$PR_{inf}\_rec(\varphi_1, a, b, s)$
fin;
$mise\_a\_jour(\varphi)$
fin;

$mise\_a\_jour(\varphi)$ = **soit** $\langle \varphi_1..\varphi_n \rangle := SFd(\varphi)$ **dans**
$\langle \varphi.p, \varphi.\overline{p} \rangle$:=$calcul\_attributs(connectif(\varphi), \langle \varphi_1.p..\varphi_n.p \rangle, \langle \varphi_1.\overline{p}..\varphi_n.\overline{p} \rangle)$;

$nb\_def(x, y, s)$ = (**si** $s \geq 0$ **alors** x **sinon** 0)+(**si** $s \leq 0$ **alors** y **sinon** 0);

sont inchangés. En sortie, l'ensemble S a été augmenté des sous-formules de φ de bénéfice positif, éventuellement de φ elle-même, auquel cas on a $\varphi.p = \varphi.\overline{p} = 1$. Sinon, on a $\varphi.p = p(\varphi[S])$ et $\varphi.\overline{p} = \overline{p}(\varphi[S])$. Ces valeurs sont utiles pour calculer les coefficients $a_{\varphi'}^{Rnm(S,\psi)}$ et $b_{\varphi'}^{Rnm(S,\psi)}$ pour les sous-formules φ' disjointes de φ qui n'ont pas encore été examinées.

Pour comprendre comment ces propriétés sont maintenues pour et par les appels récursifs, il convient d'examiner la procédure PR_{inf}_rec. Le premier test permet d'éliminer le cas $p(\varphi) = \overline{p}(\varphi) = 1$, où il n'y a rien à faire : toutes les sous-formules de φ sont de bénéfice négatif. Le deuxième test équivaut à $B(\varphi, Rnm(S, \psi)) \geq 0$; s'il est vérifié, il faut renommer φ, puis poursuivre le renommage des sous-formules de φ dans $Rnm(S \cup \{\varphi\}, \psi)$, c'est à dire dans $Def_\psi(\varphi)$. On fait donc un deuxième appel récursif sur φ (qui ne risque pas de boucler, car $B(\varphi, Def_\psi(\varphi)) < 0$), avec les valeurs de $a_\varphi^{Def_\psi(\varphi)}$ et $b_\varphi^{Def_\psi(\varphi)}$ qui dépendent de la polarité.

Si le test n'est pas vérifié, on ne renomme pas φ, et on effectue les appels récursifs successivement sur toutes les sous-formules directes de φ (dont la liste est fournie par l'appel $SFd(\varphi)$), selon le connectif de φ. Examinons le cas de l'implication : $\varphi = \varphi_1 \Rightarrow \varphi_2$.

Le premier appel est effectué sur φ, avec les coefficient $a_{\varphi_1}^{Rnm(S,\psi)}, b_{\varphi_1}^{Rnm(S,\psi)}$ calculés en fonction de $a_\varphi^{Rnm(S,\psi)}, b_\varphi^{Rnm(S,\psi)}$ selon les formules du tableau 2.2. Après ce premier appel, la valeur de S a été modifiée, et celle de $\varphi_1.\overline{p}$ l'a été en conséquence, de sorte que les paramètres du deuxième appel récursif correspondent effectivement aux valeurs $a_{\varphi_2}^{Rnm(S,\psi)}, b_{\varphi_1}^{Rnm(S,\psi)}$. En particulier, si φ_1 elle-même a été renommée, on a $\varphi_1.\overline{p} = \overline{p}(SkL_{\varphi_1}^\psi) = 1$.

Les autres appels récursifs suivent le même principe. En sortie, il y a remise à jour des attributs $\varphi.p, \varphi.\overline{p}$ de sorte qu'ils contiennent effectivement $p(\varphi[S]), \overline{p}(\varphi[S])$ en tenant compte de la nouvelle valeur de S.

Pour des raisons d'efficacité, les appels récursifs sur la conjonction et la disjonction sont particuliers. Examinons le cas de la disjonction : il faut d'après le tableau

2.2 disposer à chaque appel récursif du produit $\prod_{j \neq i} p(\varphi_j[S])$. Pour éviter de recalculer ce produit à chaque appel, c'est à dire à chaque fois que S est modifié, il suffit de remplacer les unes après les autres les anciennes valeurs de $p(\varphi_i[S])$ par les nouvelles, ce qui se fait par division et multiplication. Ainsi, avant chaque appel récursif, Π contient la valeur actuelle de $\prod_{i=1}^{n} p(\varphi_i[S])$, et donc P la valeur actuelle de $\prod_{j \neq i} p(\varphi_j[S])$. On vérifie également que la division est entière, de sorte qu'il n'y a pas de problème d'erreur d'arrondi.

Enfin, il est relativement simple d'évaluer la complexité de ce deuxième parcours. Nous avons vu qu'il y a au plus deux appels récursifs sur chaque sous-formule. Le nombre d'instructions exécutées à chaque appel est proportionnel au nombre de sous-formules directes de φ, et on peut donc compter un nombre constant d'instructions pour chaque sous-formule. Parmi ces instructions, seuls les calculs arithmétiques ne sont pas exécutés en temps constant, et on a vu que ceux-ci le sont en temps borné par $O(|SF(\varphi)|)$. La complexité de ce deuxième parcours est donc la même que celle du premier : $O(|SF(\psi)|^2)$.

Cependant, il faut également compter avec le calcul de $Rnm(S, \psi)$, qui nécessite le calcul des variables libres, ce qui implique un parcours à l'intérieur des formules atomiques de ψ. La complexité de l'algorithme est donc $O(|\psi||SF(\psi)|)$ dans le pire des cas. En pratique, la grande efficacité des calculs arithmétiques le rend presque linéaire.

6.2 Optimalité de PR_{inf}

Il faut maintenant montrer que cet algorithme, qui calcule un renommage R_{inf} avec un parcours descendant, produit effectivement, lorsque ψ est linéaire, un renommage descendant : c'est à dire libre dans ψ et vérifiant la condition de saturation (cf. chapitre 4). Pour établir cette première condition, nous devons démontrer deux lemmes postulant l'indépendance du signe des bénéfices quant à l'ordre des renommages. Comme au chapitre 4, nous nous limitons sans perte de généralité aux formes normales négatives, ce qui simplifie les preuves.

Nous commençons par analyser le cas de renommages disjoints, en imposant une hypothèse proche de la condition de saturation :

Lemme 6.1 $\forall \varphi, \varphi' \sqsubset \psi$, *si* φ *et* φ' *sont disjointes, et si* $\forall \xi \sqsupset \varphi, B(\xi, \psi) < 0$, *alors* $B(\varphi, \psi) \geq 0 \wedge B(\varphi', \psi[\varphi]) \geq 0 \Rightarrow B(\varphi, \psi[\varphi']) \geq 0$.

Preuve. Considérons la plus petite sur-formule de φ et φ', que l'on note $\varphi \sqcup \varphi'$ (borne supérieure de φ et φ' dans l'ordre $\sqsubseteq$). Si $\varphi \sqcup \varphi'$ est une conjonction, on a $B(\varphi, \psi[\varphi']) = B(\varphi, \psi)$ (cf. lemme 4.6), et donc l'implication est trivialement vérifiée.

Sinon, $\varphi \sqcup \varphi'$ est une disjonction $\varphi_1 \vee .. \vee \varphi_n$, avec $\varphi \sqsubseteq \varphi_j$ et $\varphi' \sqsubseteq \varphi_{j'}$ $(j \neq j')$. Supposons $B(\varphi, \psi) \geq 0$ et $B(\varphi', \psi[\varphi]) \geq 0$, on a $B(\varphi \sqcup \varphi', \psi) < 0$, car $\varphi \sqsubset \varphi \sqcup \varphi'$, et $p(\varphi \sqcup \varphi') \geq p(\varphi) > 1$, donc $a^{\psi}_{\varphi \sqcup \varphi'} = 1$.

Supposons que $B(\varphi, \psi[\varphi']) < 0$, soient b, b', c tels que $P^{\varphi_j}_{\varphi}(x) = a^{\varphi_j}_{\varphi} x + b$, $P^{\varphi_{j'}}_{\varphi'}(x) = a^{\varphi_{j'}}_{\varphi'} x + b'$, $P^{\psi}_{\varphi \sqcup \varphi'}(x) = x + c$ et $d = \prod_{i \neq j, j'} p(\varphi_i)$, on a :

$$\begin{aligned} P^{\psi[\varphi']}_{\varphi}(x) &= P^{\psi}_{\varphi \sqcup \varphi'}(P^{\varphi \sqcup \varphi'[\varphi']}_{\varphi}(x)) \\ &= P^{\varphi_j}_{\varphi}(x) P^{\varphi_{j'}}_{\varphi'}(1) d + c \\ &= (a^{\varphi_j}_{\varphi} x + b)(a^{\varphi_{j'}}_{\varphi'} + b') d + c \end{aligned}$$

donc $a^{\psi[\varphi']}_{\varphi} = (a^{\varphi_{j'}}_{\varphi'} + b') a^{\varphi_j}_{\varphi} d$, et vaut 1 d'après l'hypothèse (puisque $p(\varphi) > 1$). On a donc $a^{\varphi_j}_{\varphi} = a^{\varphi_{j'}}_{\varphi'} = d = 1$ et $b' = 0$. On en déduit, de même, que $P^{\psi[\varphi]}_{\varphi'}(x) = (1 + b)x + c$, mais comme $B(\varphi', \psi[\varphi]) \geq 0$, on a $b > 0$, et donc $\varphi \sqsubset \varphi_j$. D'après l'hypothèse de saturation, on a donc $B(\varphi_j, \psi) < 0$.

Cependant, $p(\varphi_j) = P^{\varphi_j}_{\varphi}(p(\varphi)) = p(\varphi) + b > 1$ d'après ce qui précède, et $P^{\psi}_{\varphi_j}(x) = P^{\varphi \sqcup \varphi'}_{\varphi_j}(x) + c = p(\varphi_{j'})x + c$, donc $a^{\psi}_{\varphi_j} = p(\varphi_{j'}) = P^{\varphi_{j'}}_{\varphi'}(p(\varphi')) = p(\varphi') > 1$ puisque $B(\varphi', \psi[\varphi]) \geq 0$. On en déduit que $B(\varphi_j, \psi) \geq 0$, ce qui est en contradiction avec ce qui précède, et on ne peut donc qu'avoir $B(\varphi, \psi[\varphi']) \geq 0$. $\dashv$

Nous prouvons maintenant un lemme similaire concernant les renommages non disjoints :

Lemme 6.2 $\forall \varphi' \sqsubseteq \varphi \sqsubseteq \psi, B(\varphi, \psi) \geq 0 \wedge B(\varphi', Def_{\psi}(\varphi)) \geq 0 \Rightarrow B(\varphi[\varphi'], \psi[\varphi']) \geq 0$

Preuve. Supposons $B(\varphi, \psi) \geq 0$ et $B(\varphi', Def_{\psi}(\varphi)) \geq 0$, on a donc $a^{\psi}_{\varphi} > 1$ et $a^{Def_{\psi}(\varphi)}_{\varphi'} = a^{\varphi}_{\varphi'} > 1$. Donc $p(\varphi[\varphi']) = P^{\varphi}_{\varphi'}(1) \geq a^{\varphi}_{\varphi'} > 1$. Mais $a^{\psi[\varphi']}_{\varphi[\varphi']} = a^{\psi}_{\varphi} > 1$, donc $B(\varphi[\varphi'], \psi[\varphi']) \geq 0$. $\dashv$

Nous pouvons maintenant démontrer le résultat annoncé, qui permet d'obtenir les résultats du chapitre 4, et en particulier l'optimalité du nombre de clauses. Pour ce faire, nous devons évidemment considérer l'ordre dans lequel les sous-formules sont renommées dans un parcours descendant, d'où la définition suivante :

Définition. La relation "est à gauche de", notée $\lhd$, est définie par : $\varphi \lhd \varphi'$ si et seulement si on a $\varphi' \sqsubseteq \varphi$, ou si φ et φ' sont disjointes et sachant que $\langle \varphi_1..\varphi_n \rangle = SFd(\varphi \sqcup \varphi'), \varphi \sqsubseteq \varphi_j$ et $\varphi' \sqsubseteq \varphi_{j'}$, on a $j < j'$. $\diamond$

De même que nous n'avons pas démontré formellement la correction de PR_{inf}, nous ne donnerons pas de démonstration du fait que les sous-formules sont renommées dans l'ordre $\lhd$, ce qui est trop évident pour mériter une preuve à la fois fastidieuse et peu originale. Nous nous intéressons ici aux propriétés de la transformation calculée par cet algorithme.

Théorème 6.3 *Le renommage calculé par $PR_{inf}(\psi)$ est descendant dans ψ.*

Preuve. On note $S = PR_{inf}(\psi)$, soient $\varphi_1 \lhd .. \lhd \varphi_n$ telles que $\{\varphi_1..\varphi_n\} = S$. On montre d'abord que la condition de saturation est vérifiée :

$\forall \varphi \sqsubset \psi$, soit i tel que $\varphi_i \lhd \varphi$ et $\varphi_{i+1} \not\lhd \varphi$ (si $\varphi_1 \not\lhd \varphi$, on prend $i = 0$). Soit $\psi' = Sup_S(\varphi)$ si φ admet une sur-formule dans S, et $\psi' = \psi$ sinon. Si $\varphi \in S$, alors $\varphi = \varphi_i = \psi'$, et donc $B(\varphi, \psi'[S - SF(\varphi)]) < 0$. Sinon, d'après les propriétés de PR_{inf}, on a $B(\varphi, \psi'[\varphi_1..\varphi_i]) < 0$, mais $SF(\varphi) \cap \{\varphi_1..\varphi_i\} = \emptyset$ puisque $\varphi_1 \lhd .. \lhd \varphi_i \lhd \varphi$, donc $\{\varphi_1..\varphi_i\} \subset S - SF(\varphi)$, et donc d'après le théorème fondamental de monotonie :

$$\begin{aligned} B(\varphi, \psi'[S - SF(\varphi)]) &= B(\varphi, Rnm(S - SF(\varphi), \psi)) \\ &\leq B(\varphi, Rnm(\varphi_1..\varphi_i, \psi)) \\ &\leq B(\varphi, \psi'[\varphi_1..\varphi_i]) \\ &< 0 \end{aligned}$$

et la condition de saturation est donc établie. Nous devons maintenant prouver que S est libre dans ψ :

$\forall i \in \{1..n\}$, on a $B(\varphi_i, Rnm(\varphi_1..\varphi_i, \psi)) \geq 0$ puisque S est p-décroissant. Soit k tel que $\varphi_k \sqsubseteq \varphi_i$ et $\varphi_{k+1} \not\sqsubseteq \varphi_i$, pour la même raison on a $\forall j \in \{i+1..k\}, B(\varphi_j, Def_\psi(\varphi_i[\varphi_{i+1}..\varphi_{j-1}])) \geq 0$. D'après le lemme 6.2, on obtient pour $j = i+1$: $B(\varphi_i[\varphi_{i+1}], Rnm(\varphi_1..\varphi_{i-1}, \psi)[\varphi_{i+1}]) \geq 0$, donc $B(\varphi_i[\varphi_{i+1}], Rnm(\varphi_1..\varphi_{i-1}\varphi_{i+1}, \psi)) \geq 0$. Par récurrence, on obtient finalement $B(\varphi_i[\varphi_{i+1}..\varphi_k], Rnm(\varphi_1..\varphi_{i-1}\varphi_{i+1}..\varphi_k, \psi)) \geq 0$. Dans la suite, on note $B(\varphi_i, Rnm(\varphi_1..\varphi_{i-1}\varphi_{i+1}..\varphi_k, \psi)) \geq 0$, en vertu de l'abus de notation autorisé.

On a également $\forall j \in \{k+1..n\}, B(\varphi_j, Rnm(\varphi_1..\varphi_{j-1}, \psi)) \geq 0$, donc $B(\varphi_j, Rnm(\varphi_1..\varphi_{i-1}\varphi_{i+1}..\varphi_{j-1}, \psi)[\varphi_i]) \geq 0$ (on enlève la définition de φ_i, ce qui est possible puisqu'elle est disjointe de φ_j). Pour $j = k+1$, on peut donc appliquer le lemme 6.1, et on obtient $B(\varphi_i, Rnm(\varphi_1..\varphi_{i-1}\varphi_{i+1}..\varphi_k, \psi)[\varphi_{k+1}]) \geq 0$,

donc $B(\varphi_i, Rnm(\varphi_1..\varphi_{i-1}\varphi_{i+1}..\varphi_{k+1}, \psi)) \geq 0$ (on rajoute la définition de φ_{k+1}). Par récurrence, on obtient donc $B(\varphi_i, Rnm(\varphi_1..\varphi_{i-1}\varphi_{i+1}..\varphi_n, \psi)) \geq 0$, c'est à dire $B(\varphi_i, Rnm(S - \{\varphi_i\}, \psi)) \geq 0$.

D'après le théorème 4.2, on en déduit que S est libre dans ψ. ⊣

Nous en déduisons donc, toujours dans le cas où ψ est linéaire :

Corollaire 6.4 *Le renommage calculé par $PR_{inf}(\psi)$ est optimal dans ψ.*

Preuve. Conséquence directe du théorème 6.3 et du corollaire 4.17. ⊣

6.3 Modifications pour le calcul de R'_{inf}

A priori, les modifications à apporter à l'algorithme présenté ci-dessus pour obtenir un algorithme calculant efficacement $R'_{inf}(\psi)$ sont relativement simples. Il nous faudra cependant détailler un point difficile, qui sera donc l'objet principal de cette section.

Si les modifications sont simples, il n'en reste pas moins vrai que toutes les procédures sont à modifier. Pour le premier parcours, c'est à dire les procédures *attribution* et *calcul_attributs*, il suffit de se référer au tableau 5.1. Il est clair qu'on a alors 4 attributs par sous-formule au lieu de 2 : $\varphi.p, \varphi.\overline{p}$, mais aussi $\varphi.q$ et $\varphi.\overline{q}$. La modification de la procédure *mise_a_jour* suit celle de *calcul_attributs*, et la procédure *nb_def* devient :

$nb\_def'(x, y, u, v, s) =$
(**si** $s \geq 0$ **alors** $x + u$ **sinon** 0)+ (**si** $s \leq 0$ **alors** $y + v$ **sinon** 0)

Dans PR_{inf}, il n'y a que les paramètres et les affectations qui changent, pour tenir compte des nouveaux attributs et des nouveaux coefficients qui sont en argument de la procédure récursive : $PR'_{inf}\_rec(\varphi, a, b, c, d, s)$. Les coefficients initiaux sont évidemment $a = b = d = 0$ et $c = 1$.

Dans la procédure PR'_{inf}_rec, la plupart des changements sont relativement simples : le premier test peut être remplacé par $\varphi.q \neq 1$, qui suffit à éliminer les formules atomiques. Le deuxième est évidemment remplacé par $a * \varphi.p + b * \varphi.\overline{p} + c * \varphi.q + d * \varphi.\overline{q} - a - b - c - d \geq nb\_def'(\varphi.p, \varphi.\overline{p}, \varphi.q, \varphi.\overline{q}, s)$. S'il y a renommage de φ, les nouveaux coefficients sont :

$a = c =$ **si** $s \geq 0$ **alors** 1 **sinon** 0
$b = d =$ **si** $s \leq 0$ **alors** 1 **sinon** 0

S'il n'y a pas renommage, on procède de même par appels récursifs, en calculant les nouveaux coefficients d'après la table 5.2. La difficulté est alors de faire ce calcul efficacement dans les cas de la disjonction et de la conjonction. Nous indiquons la façon de procéder pour la disjonction, et nous en prouverons ensuite la correction. Ceci fournira également une justification plus formelle de la correction du calcul des coefficients dans PR_{inf}_rec pour la conjonction et la disjonction (et montrera également que l'on peut y remplacer chaque occurrence de la variable P par Π)

$\Pi := \prod_{i=1}^{n} \varphi_i.p$;
$\Sigma := \sum_{i=1}^{n} (\Pi * \varphi_i.q)/\varphi_i.p$;
pour $i = 1$ **a** n **faire**
  debut
    $\Sigma := \Sigma - \Pi$; $\Pi := \Pi/\varphi_i.p$; $\Sigma := (\Sigma + \Pi * (\varphi_i.p - \varphi_i.q))/\varphi_i.p$;
    $PR'_{inf}\_rec(\varphi_i, a * \Pi + c * \Sigma, b, c * \Pi, d, s)$;
    $\Sigma := \Sigma * \varphi_i.p + \Pi * \varphi_i.q$; $\Pi := \Pi * \varphi_i.p$
  fin;

D'après le tableau 5.2, on a $a^{\psi}_{\varphi_i} = a^{\psi}_{\varphi} P_i + c^{\psi}_{\varphi} S_i$ et $c^{\psi}_{\varphi_i} = c^{\psi}_{\varphi} P_i$, avec :

$$P_i = \prod_{j \neq i} p(\varphi_j) \text{ et } S_i = \sum_{j \neq i} \frac{q(\varphi_j) P_i}{p(\varphi_j)}$$

Mais comme précédemment, les valeurs de $\varphi_i.p, \varphi_i.q$ sont modifiées par l'appel récursif sur φ_i. Si on note p_i, q_i leur valeur avant cet appel, et p'_i, q'_i leur valeur après, comme ces appels se font dans l'ordre $\varphi_1..\varphi_n$, on a :

$$P_i = \prod_{j=1}^{i-1} p'_j \prod_{j=i+1}^{n} p_j \text{ et } S_i = (\sum_{j=1}^{i-1} \frac{q'_j}{p'_j} + \sum_{j=i+1}^{n} \frac{q_j}{p_j}) P_i$$

Il faut donc montrer qu'à chaque itération i, les valeurs de Π et Σ sont effectivement P_i et S_i pendant l'appel récursif. Pour cela, il faut être capable d'exprimer P_{i+1} (resp. S_{i+1}) en fonction de $P_i, S_i, p'_i, q'_i, p_{i+1}, q_{i+1}$ (resp. et P_{i+1} si cette valeur est calculée avant S_{i+1}). On a :

$$P_{i+1} = \prod_{j=1}^{i} p'_j \prod_{j=i+2}^{n} p_j = p'_i \prod_{j=1}^{i-1} p'_j \frac{1}{p_{i+1}} \prod_{j=i+1}^{n} p_j = \frac{P_i p'_i}{p_{i+1}}$$

et :

$$\begin{aligned} S_{i+1} &= (\sum_{j=1}^{i} \frac{q'_j}{p'_j} + \sum_{j=i+2}^{n} \frac{q_j}{p_j}) P_{i+1} \\ &= (\sum_{j=1}^{i-1} \frac{q'_j}{p'_j} + \frac{q'_i}{p'_i} + \sum_{j=i+1}^{n} \frac{p_j}{q_j} - \frac{q_{i+1}}{p_{i+1}}) \frac{P_i p'_i}{p_{i+1}} \\ &= \frac{S_i p'_i}{p_{i+1}} + \frac{P_i q'_i}{p_{i+1}} - \frac{q_{i+1}}{p_{i+1}} \frac{P_i p'_i}{p_{i+1}} \\ &= \frac{S_i p'_i + P_i q'_i - P_{i+1} q_{i+1}}{p_{i+1}} \end{aligned}$$

on obtient donc les deux relations :

$$P_{i+1} p_{i+1} = P_i p'_i \qquad \text{et} \qquad S_{i+1} p_{i+1} + P_{i+1} q_{i+1} = S_i p'_i + P_i q'_i$$

Après l'initialisation des variables Π et Σ, on a :

$$\begin{aligned} \Pi &= \prod_{i=1}^{n} p'_i = P_1 p_1 \\ \Sigma &= \sum_{i=1}^{n} \frac{q_i}{p_i} P_1 p_1 = S_1 p_1 + P_1 q_1 \end{aligned}$$

Nous postulons donc qu'en entrée de l'itération on a $\Pi = P_i p_i$ et $\Sigma = S_i p_i + P_i q_i$. Après la première affectation, on a donc $\Sigma = S_i p_i + P_i(q_i - p_i)$. Après la deuxième, $\Pi = P_i$, et après la troisième, $\Sigma = \frac{S_i p_i + P_i(q_i - p_i) + P_i(p_i - q_i)}{p_i} = S_i$, car $\varphi_i.p = p_i$ et $\varphi_i.q = q_i$. Π et Σ ont donc les valeurs correctes pour l'appel récursif. Au retour de cet appel, on a $\varphi_i.p = p'_i$ et $\varphi_i.q = q'_i$. L'avant-dernière affectation donne donc $\Sigma = S_i p'_i + P_i q'_i = S_{i+1} p_{i+1} + P_{i+1} q_{i+1}$, et la dernière $\Pi = P_i p'_i = P_{i+1} p_{i+1}$. Au début de l'itération suivante, i est incrémenté de 1, et on retrouve donc les valeurs postulées pour Π et Σ.

6.4 Une optimisation

On peut se demander s'il n'est pas préférable d'éviter les renommages dans le cas $B(\varphi, \psi) = 0$, et donc de remplacer le test de renommage $B(\varphi, \psi) \geq 0$ par $B(\varphi, \psi) > 0$. Ce qui signifie qu'on se limite alors à des renommages *strictement* p-décroissants. Il n'est cependant pas évident qu'on obtienne alors moins de clauses : si une sous-formule de bénéfice nul n'est pas renommée, les bénéfices subséquents s'en trouvent augmentés d'après le théorème fondamental de monotonie. Il y aura donc plus de

renommages dans un premier temps, et cela va faire décroître les bénéfices ultérieurs. On peut en définitive obtenir des renommages très différents, et il est difficile de comparer leur nombre de clauses respectifs. Il en est bien entendu de même pour le nombre de littéraux.

Intuitivement, cette façon de faire semble cependant une amélioration de la précédente, et on devrait obtenir les mêmes propriétés de complexité en taille. C'est relativement évident en ce qui concerne le nombre de clauses (resp. le nombre de littéraux). Si en effet on considère un renommage R strictement p-décroissant, et strictement complet, c'est à dire tel que $\forall \varphi \sqsubset \psi, B(\varphi, Rnm(R,\psi)) \leq 0$. On peut alors compléter R en y rajoutant les sous-formules de bénéfice nul, pour obtenir $R' \supset R$ (R' est le plus petit renommage complet de ψ contenant R). On a alors $p(Rnm(R,\psi)) = p(Rnm(R',\psi))$, et R' est un renommage R_{inf}. D'après le théorème 3.7, on obtient donc moins de clauses avec R' qu'avec le renommage structure-preserving.

Ce résultat est suffisant pour obtenir la même borne pour la longueur de la forme clausale obtenue avec R que celle obtenue par le théorème 3.11, car c'est le seul résultat utilisé qui dépend du renommage (avec le lemme 3.10, qui est également démontré à partir de ce résultat uniquement). Toutes ces remarques sont évidemment applicables aux renommages strictement q-décroissants.

Par contre, la construction de R' à partir de R ne respecte pas nécessairement la stratégie. En particulier, si R' est descendant, R ne l'est pas forcément, et il nous faut donc trouver un autre argument pour démontrer l'optimalité de l'équivalent strict des renommages descendants dans le cas linéaire : nous noterons PR^{+}_{inf} la procédure obtenue à partir de PR_{inf} en remplaçant $\geq$ par $>$. Nous montrons que dans le cas linéaire, on obtient avec cette nouvelle procédure un sous-renommage du précédent. De même que précédemment, nous nous limitons sans perte de généralité aux formes normales négatives.

Théorème 6.5 $\forall \psi$ *linéaire,* $PR^{+}_{inf}(\psi) \subset PR_{inf}(\psi)$.

Preuve. $\forall \varphi \sqsubset \psi$, on note $R_\varphi = \{\xi \in PR^{+}_{inf}(\psi) / \xi \lhd \varphi\}$ et $R'_\varphi = \{\xi \in PR_{inf}(\psi) / \xi \lhd \varphi\}$. On a évidemment $(\forall \varphi \sqsubset \psi, R_\varphi \subset R'_\varphi) \Rightarrow PR^{+}_{inf}(\psi) \subset PR_{inf}(\psi)$, et nous allons donc montrer $\forall \varphi \sqsubset \psi, R_\varphi \subset R'_\varphi$, par induction faible sur φ dans l'ordre $\lhd$: on suppose donc, pour un φ donné, qu'on a $\forall \xi \sqsubset \psi, \xi \lhd \varphi \wedge \xi \neq \varphi \Rightarrow R_\xi \subset R'_\xi$.

On procède par l'absurde, en supposant que $R_\varphi \not\subset R'_\varphi$. On a donc $\varphi \in R_\varphi$ et $\varphi \notin R'_\varphi$, puisque $R_\varphi - \{\varphi\} \subset R'_\varphi - \{\varphi\}$ d'après l'hypothèse d'induction. Par définition de

PR^{+}_{inf} et PR_{inf}, on a donc $B(\varphi, Rnm(R_\varphi - \{\varphi\}, \psi)) > 0$ et $B(\varphi, Rnm(R'_\varphi, \psi)) < 0$. Comme les éléments de R_φ et R'_φ précèdent φ dans l'ordre $\lhd$, on a $R_\varphi - \{\varphi\} \cap SF(\varphi) = \emptyset$ et $R'_\varphi \cap SF(\varphi) = \emptyset$, donc $B(\varphi, Rnm(R_\varphi - \{\varphi\}, \psi)) = (a_\varphi^{Rnm(R_\varphi - \{\varphi\}, \psi)} - 1)(p(\varphi) - 1) - 1$ et $B(\varphi, Rnm(R'_\varphi, \psi)) = (a_\varphi^{Rnm(R'_\varphi, \psi)} - 1)(p(\varphi) - 1) - 1$. On a donc $a_\varphi^{Rnm(R_\varphi - \{\varphi\}, \psi)} > 1$ et $p(\varphi) > 1$, donc $a_\varphi^{Rnm(R'_\varphi, \psi)} = 1$.

Par un raisonnement similaire, on montre que $\forall \xi \lhd \varphi$, si $\xi \in R'_\varphi - R_\varphi$ alors $a_\xi^{Rnm(R_\xi, \psi)} = a_\xi^{Rnm(R'_\xi, \psi)} = p(\xi) = 2$. On a en effet les deux inégalités $B(\xi, Rnm(R'_\xi - \{\xi\}, \psi)) \geq 0$ et $B(\xi, Rnm(R_\xi, \psi)) \leq 0$. Comme $R_\xi \subset R'_\xi - \{\xi\}$ par hypothèse d'induction, on a d'après la preuve du théorème fondamental de monotonie (cas 1) que $a_\xi^{Rnm(R'_\xi - \{\xi\}, \psi)} \leq a_\xi^{Rnm(R_\xi, \psi)}$. D'après la première inégalité, on a $p(\xi) > 1$ et $a_\xi^{Rnm(R'_\xi - \{\xi\}, \psi)} > 1$. La seule possibilité pour que la deuxième soit vérifiée est donc $p(\xi) = 2$ et $a_\xi^{Rnm(R_\xi, \psi)} = 2$, ce qui donne également $a_\xi^{Rnm(R'_\xi - \{\xi\}, \psi)} = 2$. Ceci nous sera utile par la suite.

On note $\psi' = Sup_{R_\varphi - \{\varphi\}}(\varphi)$ s'il existe, et $\psi' = \psi$ sinon, de sorte que $a_\varphi^{Rnm(R_\varphi - \{\varphi\}, \psi)} = a_\varphi^{\psi'[R_\varphi]}$.

On montre que $Sup_{R'_\varphi}(\varphi) = \psi'$ par l'absurde. On note $\xi = Sup_{R'_\varphi}(\varphi)$, supposons que $\xi \sqsubset \psi'$, on a $\xi \in R'_\varphi - R_\varphi$ avec $\xi \lhd \varphi$ (car $\varphi \sqsubset \xi$), et donc $a_\xi^{\psi'[R_\xi]} = p(\xi) = 2$. Mais on a alors $a_\varphi^{\xi[R'_\varphi]} = a_\varphi^{Rnm(R'_\varphi, \psi)} = 1$, donc $p(\xi[R'_\varphi]) = p(\varphi) > 1$, or $p(\xi[R'_\varphi]) \leq p(\xi) = 2$, donc $p(\varphi) = 2$. Comme $p(\xi) = a_\varphi^\xi p(\varphi)$, on a alors $a_\varphi^\xi = 1$. Comme $B(\varphi, Rnm(R_\varphi - \{\varphi\}, \psi)) > 0$, on a également $a_\varphi^{\psi'[R_\varphi]} > 2$. Mais $a_\varphi^{\psi'[R_\varphi]} = a_\xi^{\psi'[R_\xi]} a_\varphi^{\xi[R_\varphi]}$ et $a_\varphi^{\xi[R_\varphi]} \leq a_\varphi^\xi$, ce qui donne d'après les valeurs obtenues $a_\varphi^{\psi'[R_\varphi]} = 2$. On a donc une contradiction, ce qui permet de déduire que $Sup_{R'_\varphi}(\varphi) = \psi'$.

De sorte qu'on a également $a_\varphi^{\psi'[R'_\varphi]} = a_\varphi^{Rnm(R'_\varphi, \psi)} = 1$. Comme $a_\varphi^{\psi'[R_\varphi]} > 1$, donc différent du précédent, on a $Inf_{R_\varphi - \{\varphi\}}(\psi') \neq Inf_{R'_\varphi}(\psi')$. On a donc $Inf_{R'_\varphi}(\psi') \not\subset R_\varphi$ (sinon il y aurait égalité, puisque $R_\varphi - \{\varphi\} \subset R'_\varphi$: en descendant dans ψ', chaque sous-formule rencontrée appartenant à l'un des renommages appartiendrait également à l'autre), d'où $Inf_{R'_\varphi}(\psi') - R_\varphi \neq \emptyset$.

Si $\forall \varphi' \in Inf_{R'_\varphi}(\psi') - R_\varphi$, $\varphi' \sqcup \varphi$ est une conjonction, on montre facilement d'après le lemme 4.6 que $B(\varphi, \psi'[R'_\varphi]) = B(\varphi, \psi'[R_\varphi - \{\varphi\}])$, ce qui est impossible puisque le premier est strictement négatif et le deuxième strictement positif. Soit donc $\varphi' \in Inf_{R'_\varphi}(\psi') - R_\varphi$ telle que $\varphi' \sqcup \varphi$ est une disjonction $\varphi_1 \vee .. \vee \varphi_n$. Soient i et j tels que $\varphi' \sqsubseteq \varphi_i$ et $\varphi \sqsubseteq \varphi_j$, comme $\varphi' \lhd \varphi$, on a $i < j$.

On a également $\varphi' \in R'_\varphi - R_\varphi$, donc $a_{\varphi'}^{\psi'[R_{\varphi'}]} = p(\varphi') = 2$. Or :

$$a_{\varphi'}^{\psi'[R_{\varphi'}]} = a_{\varphi' \sqcup \varphi}^{\psi'[R_{\varphi'}]} \prod_{k=1}^{i-1} p(\varphi_k[R_{\varphi'}]) a_{\varphi'}^{\varphi_i[R_{\varphi'}]} \prod_{k=i+1}^{j-1} p(\varphi_k) a_\varphi^{\varphi_j} p(\varphi) \prod_{k=j+1}^{n} p(\varphi_k)$$

et $p(\varphi) > 1$, donc $p(\varphi) = 2$ et tous les autres facteurs sont égaux à 1. On a de même :

$$a_\varphi^{\psi'[R_\varphi]} = a_{\varphi' \sqcup \varphi}^{\psi'[R_\varphi]} \prod_{k=1}^{i-1} p(\varphi_k[R_\varphi]) a_{\varphi'}^{\varphi_i[R_\varphi]} p(\varphi'[R_\varphi]) \prod_{k=i+1}^{j-1} p(\varphi_k[R_\varphi]) a_\varphi^{\varphi_j[R_\varphi]} \prod_{k=j+1}^{n} p(\varphi_k)$$

avec $p(\varphi'[R_\varphi]) \leq p(\varphi') = 2$, et comme $R_{\varphi'} \subset R_\varphi$, tous les autres facteurs sont inférieurs à ce qu'ils étaient auparavant, donc égaux à 1. On a donc $a_\varphi^{\psi'[R_\varphi]} \leq 2$, ce qui montre que $B(\varphi, \psi'[R_\varphi - \{\varphi\}]) \leq 0$, en contradiction avec l'hypothèse de départ.

On a donc bien $R_\varphi \subset R'_\varphi$, et l'induction est terminée. ⊣

On en déduit facilement le

Corollaire 6.6 *$\forall \psi$ linéaire, $PR^+_{inf}(\psi)$ est optimal.*

Preuve. $PR^+_{inf}(\psi) \subset PR_{inf}(\psi)$ d'après le théorème 6.5, donc $p(Rnm(PR^+_{inf}(\psi), \psi)) \leq p(Rnm(PR_{inf}(\psi), \psi))$ d'après le théorème 3.5, et donc $PR^+_{inf}(\psi)$ est optimal d'après le corollaire 6.4. ⊣

Il semble bien qu'on obtienne toujours moins de clauses avec $PR^+_{inf}(\psi)$ qu'avec $PR_{inf}(\psi)$, quelque soit ψ. Cependant, lorsque ψ contient des équivalences, l'inclusion démontrée au théorème 6.5 n'est pas toujours vérifiée, et nous ne pouvons procéder de la même façon que dans le cas linéaire. Nous n'essayons donc pas de le démontrer, et nous nous contenterons du corollaire 6.6 pour adopter cette optimisation, car elle permet au moins sur les formules linéaires de diminuer le nombre de sous-formules renommées, et nous verrons au chapitre suivant que ce n'est pas sans importance. Dans la suite, nous noterons R_{opt} le renommage calculé par PR^+_{inf}.

Chapitre 7

Comparaisons et expérimentations

Et près de son oreille,
Merveille, un réveil vermeil
Lui prodigue des conseils
Pendant son sommeil.

Bobby Lapointe

7.1 Introduction

Nous nous sommes jusqu'à présent intéressés à des problèmes exclusivement syntaxiques concernant le problème de la transformation sous forme clausale. Nous devons maintenant nous pencher sur l'influence que peuvent avoir ces transformations sur le mécanisme de démonstration — i.e. la méthode de résolution — qui en constitue la raison d'être. Par delà la minimisation du nombre de clauses ou du

nombre de littéraux, notre but initial est de minimiser le temps de réfutation de la formule considérée, selon la forme clausale obtenue. Il ne s'agit pas uniquement de minimiser la longueur des réfutations, comme dans [Tse68], si cela doit augmenter l'espace de recherche.

Mais alors que nous avons pu caractériser précisément les propriétés syntaxiques des différentes transformations présentées, nous ne pouvons nous attendre à en faire autant des réfutations. Il faudrait pour cela commencer par caractériser les formes clausales issues de ces transformations, ce qui est d'autant plus difficile qu'il semble nécessaire de distinguer les littéraux de Skolem des autres (cf. l'introduction du chapitre 3).

Il faudrait ensuite, au minimum, pouvoir comparer les classes de formes clausales ainsi obtenues quant aux possibilités de réfutation par une stratégie de résolution tenant compte des littéraux de Skolem. Au vu de la généralité de ces classes — à *toute* formule on peut faire correspondre une de ces formes clausales —, et dans l'état actuel des connaissances, cela semble illusoire.

Nous devons donc nous contenter de comparaisons partielles ou imprécises, que nous espérons significatives. Bien entendu, nous avons déjà établi de telles comparaisons dans les précédents chapitres. Nous voulons ici compléter cette étude par un ensemble de considérations qui nous semblent importantes quant aux mérites respectifs de ces formes clausales, que nous abordons dans la suite l'une après l'autre. Nous examinerons en dernier lieu certains résultats expérimentaux, qui constituent d'une certaine façon la meilleure comparaison possible en l'absence de résultats plus généraux.

7.2 Idempotence

Nous avons déjà mentionné l'intérêt d'un renommage idempotent en introduction du chapitre 3. De toutes les transformations que nous avons étudiées, seule la transformation structure preserving n'est pas idempotente (ni le renommage, ni la mise sous forme clausale correspondante). Il est alors clair qu'une telle transformation ne dépend pas que de la nature du problème, mais aussi de la façon dont il est formulé. C'est bien entendu également le cas des autres transformations, mais dans une moindre mesure ; *struct_pres* est plus à même qu'un autre renommage de produire une forme clausale plus complexe et difficile à prouver qu'en absence de renommage, ce que nous illustrons par l'exemple ci-dessous.

Considérons la formule $\neg(R_1 \Leftrightarrow (R_2 \Leftrightarrow (R_1 \Leftrightarrow R_2)))$, qui est insatisfaisable comme on peut facilement s'en rendre compte en la mettant sous forme clausale standard et en simplifiant l'ensemble de clauses obtenu ; il ne reste alors que 4 clauses, à partir desquelles il est très facile de produire la clause vide. Après le renommage *struct_pres*, on obtient $P_1 \wedge (P_1 \Rightarrow (R_1 \Leftrightarrow P_2)) \wedge (P_2 \Leftrightarrow (R_2 \Leftrightarrow P_3)) \wedge (P_3 \Leftrightarrow (R_1 \Leftrightarrow R_2))$. On se rend facilement compte que cette dernière formule est plus difficile à réfuter que la précédente. Les autres renommages laissent cette formule intacte.

7.3 Elimination d'inférences triviales

Bien qu'il soit en général impossible de prévoir l'influence du renommage d'une sous-formule sur la longueur de la réfutation, on peut cependant détecter certains cas où le renommage conduit nécessairement à une réfutation plus longue, contenant des "inférences triviales" qu'il est souhaitable d'éliminer, ce qui correspond à la suppression de ce renommage. Il faut pour cela tenir compte de la stratégie utilisée qui, dans tous les cas (ordonnancement des prédicats, lock resolution...), vérifie les conditions suivantes :

1. Ne pas résoudre sur un littéral de Skolem une clause qui contient un littéral de la formule initiale.
2. Ne pas résoudre sur un littéral SkP_{φ}^{ψ} une clause qui contient $SkP_{\varphi'}^{\psi}$ avec $\varphi' \sqsubset \varphi \sqsubset \psi$.

Déjà informellement mentionnées au chapitre 3, ces règles ont pour but d'empêcher, lors de la réfutation des clauses de $clausale(Rnm(R, \psi))$, l'apparition de clauses d'un $clausale(Rnm(R', \psi))$ avec $R' \subset R$. Leur absence entraîne la réfutation simultanée de toutes ces formules. Elles permettent donc d'assurer une discrimination entre les différents renommages — condition d'une comparaison saine —, et de rapprocher la complexité du démonstrateur de celle des plus courtes preuves.

Les premiers résultats que nous abordons ici concernent des éliminations de renommage qui sont déjà réalisées dans *struct_pres* : celles des négations et des formules atomiques. Nous en montrons ici la pertinence. Dans le cas de la négation, le résultat n'est valide que si la formule niée est elle-même renommée :

Théorème 7.1 *Soit D une réfutation de $clausale(Rnm(R \cup \{\varphi, \neg\varphi\}, \psi))$, il existe une réfutation D' de $clausale(Rnm(R \cup \{\varphi\}, \psi))$ ayant moins de pas d'inférence que D.*

Preuve. Nous nous limitons au cas où $pol(\varphi,\psi) = 0$, sachant que celui-ci englobe les deux autres. Soient $x_1..x_n$ les variables libres de φ, nous obtenons à partir de $Def_\psi(\neg\varphi[\varphi])$ les deux clauses suivantes :

$$\begin{aligned} C^+ &= SkP^\psi_{\neg\varphi}(x_1..x_n) \vee SkP^\psi_\varphi(x_1..x_n) \\ C^- &= \neg SkP^\psi_{\neg\varphi}(x_1..x_n) \vee \neg SkP^\psi_\varphi(x_1..x_n) \end{aligned}$$

D'après la règle 2, toute résolution avec C^+ et C^- se fait sur SkP^ψ_φ, et consiste donc essentiellement à remplacer dans une clause $\neg SkP^\psi_\varphi$ (resp. SkP^ψ_φ) par $SkP^\psi_{\neg\varphi}$ (resp. $\neg SkP^\psi_{\neg\varphi}$). On obtient D' en supprimant dans D toute occurrence de C^+ et C^-, ainsi que les résolutions sur ces clauses, et en y remplaçant partout $\neg SkP^\psi_{\neg\varphi}$ (resp. $SkP^\psi_{\neg\varphi}$) par SkP^ψ_φ (resp. $\neg SkP^\psi_\varphi$). Cette dernière opération identifie toute clause résolue sur C^+ et C^- avec sa résolvente, ce qui justifie leur disparition. On vérifie aisément que D' est une réfutation : tous les pas de résolution, même sur $SkP^\psi_{\neg\varphi}$, restent corrects, ainsi que les pas de factorisation. Enfin, les clauses d'entrées de D' sont celles de D, sauf C^+ et C^-, et moyennant la disparition des $SkP^\psi_{\neg\varphi}$, au profit de SkP^ψ_φ : on obtient exactement les clauses de $clausale(Rnm(R \cup \{\varphi\},\psi))$.

La différence en nombre d'inférences entre D et D' est ici exactement le nombre d'occurrences de C^+ et C^- dans D. $\dashv$

On remarque d'après cette preuve qu'on aurait très bien pu supprimer plutôt le renommage de φ : les réfutations de $clausale(Rnm(R \cup \{\varphi\},\psi))$ sont isomorphes à celles de $clausale(Rnm(R \cup \{\neg\varphi\},\psi))$. Ceci justifie que R_{inf} renomme indifféremment une négation ou la formule niée, mais jamais les deux. On peut se rendre compte aisément qu'il en est de même pour R'_{inf}. Dans le cas de R_{opt}, si l'une des deux est renommée, c'est la négation.

Nous nous intéressons maintenant au cas des formules atomiques, plus complexe.

Théorème 7.2 *Soit D une réfutation de $clausale(Rnm(R \cup \{\varphi\},\psi))$, où φ est atomique, il existe alors une réfutation de $clausale(Rnm(R,\psi))$ ayant moins de pas d'inférence que D.*

Preuve. Ici aussi, nous nous limitons au cas $pol(\varphi,\psi) = 0$. Si $x_1..x_n$ sont les variables libres de φ, et $\varphi = P(t_1..t_m)$, on obtient de $Def_\psi(\varphi)$ les clauses :

$$\begin{aligned} C^+ &= \neg SkP^\psi_\varphi(x_1..x_n) \vee P(t_1..t_m) \\ C^- &= SkP^\psi_\varphi(x_1..x_n) \vee \neg P(t_1..t_m) \end{aligned}$$

A partir de D, on obtient D^* en y remplaçant chaque clause C par C^*, en y remplaçant chaque littéral L par L^*, défini par $SkP^{\psi}_{\varphi}(t'_1..t'_n)^* = P(t_1..t_m)[x_i \leftarrow t'_i]^n_{i=1}$, $(\neg L)^* = \neg L^*$ et $L^* = L$ dans tous les autres cas. Pour montrer que D^* est une réfutation correcte, nous montrons d'abord que les pas de résolution de D sont transformés en pas de résolution dans D^* :

$$\frac{L \vee C \qquad L' \vee C'}{\sigma(C \vee C')} \longrightarrow \frac{L^* \vee C^* \qquad L'^* \vee C'^*}{[\sigma(C \vee C')]^*}$$

où $\sigma = pgu(L^c, L')$. On vérifie d'abord que $[\sigma(C \vee C')]^* = \sigma(C \vee C')^*$, en montrant que pour tout littéral l, on a $(\sigma l)^* = \sigma l^*$. Si $l = SkP^{\psi}_{\varphi}(t'_1..t'_n)$, alors $\sigma l^* = \sigma P(t_1..t_m)[x_i \leftarrow t'_i]^n_{i=1} = P(t_1..t_m)[x_i \leftarrow \sigma t'_i]^n_{i=1} = (\sigma l)^*$; sinon, $l^* = l$, $(\sigma l)^* = \sigma l$, et l'égalité est évidente.

On vérifie ensuite que $pgu(L^{c*}, L'^*) = \sigma$. Si $L = SkP^{\psi}_{\varphi}(t'_1..t'_n)$, alors $L' = SkP^{\psi}_{\varphi}(t''_1..t''_n)$, avec $\sigma = pgu(\langle t'_1..t'_n\rangle, \langle t''_1..t''_n\rangle)$, et $pgu(L^{c*}, l'^*) = pgu(\langle t_1..t_m\rangle[x_i \leftarrow t'_i]^n_{i=1}, \langle t_1..t_m\rangle[x_i \leftarrow t''_i]^n_{i=1}) = \sigma$; sinon, $L^{c*} = L^c$, $L'^* = L'$, et l'égalité est évidente. On a donc prouvé que la transformée de la résolvente est bien la résolvente des transformées.

Nous considérons maintenant les pas de factorisation :

$$\frac{L \vee L' \vee C}{\sigma L \vee \sigma C} \longrightarrow \frac{L^* \vee L'^* \vee C^*}{(\sigma L \vee \sigma C)^*}$$

où $\sigma = pgu(L, L')$. Comme précédemment, on a $(\sigma L \vee \sigma C)^* = \sigma L^* \vee \sigma C^*$, et $\sigma = pgu(L^*, L'^*)$, ce qui prouve que les transformées des facteurs sont bien les facteurs des transformées.

D^* est donc une réfutation, ayant même nombre de pas d'inférence que D. Ses clauses d'entrées sont celles de $clausale(Rnm(R, \psi))$, augmentées des deux tautologies $C^{+*} = \neg P(t_1..t_m) \vee P(t_1..t_m)$ et C^{-*}, et il existe donc une réfutation de $clausale(Rnm(R, \psi))$ plus courte que D^*, donc que D. $\dashv$

De même que précédemment, les renommages R_{inf} n'incluent aucune formule atomique : $p(Rnm(R, \psi)) < p(Rnm(R \cup \{\varphi\}, \psi))$. Il en est évidemment de même des renommages R'_{inf}.

Nous montrons maintenant un résultat plus général, dans ce sens qu'il permet de supprimer les renommages d'un ensemble plus large de sous-formules, autres que simplement les négations et les formules atomiques. Ce résultat n'est cependant acquis que pour des sous-formules de polarité non nulle. Comme pour les négations, il est relatif au renommage d'une autre sous-formule. Avant de caractériser les sous-formules dont le renommage est ainsi rendu caduque, nous exprimons

ce théorème[1] directement sur la forme clausale. Nous utiliserons dans la preuve des résultats élémentaires concernant l'unification syntaxique, que l'on pourra trouver dans [JK90].

Théorème 7.3 *Si clausale*$(Rnm(R \cup \{\varphi, \varphi'\}, \psi))$ *contient une clause*

$$C^+ = \neg SkL^{\psi}_{\varphi'} \vee SkP^{\psi}_{\varphi}(t_1..t_m)$$

qui contient la seule occurrence positive de SkP^{ψ}_{φ}*, avec* $\varphi \sqsubset \varphi'$*, et si* D *est une réfutation de cet ensemble de clauses, alors il existe une réfutation de* *clausale*$(Rnm(R \cup \{\varphi'\}, \psi))$ *ayant moins de pas d'inférence que* D*.*

Preuve. Comme $\varphi \sqsubset \varphi'$, d'après la règle 2, toute résolution avec C^+ se fait sur SkP^{ψ}_{φ}. On peut donc supprimer ces résolutions, et la clause C^+ elle-même, de D en y remplaçant toute clause contenant $\neg SkP^{\psi}_{\varphi}$ par sa résolvente avec C^+. Avant de définir précisément cette opération, nous commençons par analyser la structure de $t_1..t_m$.

En effet, nous n'avons avant la mise sous forme clausale dans $Rnm(R \cup \{\varphi, \varphi'\}, \psi)$ que des occurrences de SkL^{ψ}_{φ} ; les termes $t_1..t_m$ qui ne seraient pas des variables proviennent de skolemisations, et ont donc une forme particulière. Soient $x_1..x_n$ les variables libres de φ', celles-ci ne peuvent évidemment être skolemisées, puisqu'elles sont universellement quantifiées dans $Def_{\psi}(\varphi')$, d'où vient C^+. A chacune de ces variables correspond donc un t_i qui lui est égal. Il en est de même des autres variables libres de $t_1..t_m$, qu'on note $y_1..y_p$: elles correspondent aux quantificateurs universels dans φ' qui englobent φ. Enfin, les autres termes correspondent aux quantificateurs existentiels entre φ et φ', et sont donc de la forme[2] $f_i(x_1..x_n y_1..y_{p_i})$, et deux à deux distincts puisque SkL^{ψ}_{φ} est linéaire.

Nous noterons X pour la suite $x_1..x_n$, Y pour la suite $y_1..y_p$, et par abus de notation $f(XY)$ pour la suite des termes de Skolem. Nous supposons que les t_i sont rangés dans un ordre ad-hoc, de sorte que l'on peut écrire $C^+ = \neg SkP^{\psi}_{\varphi'}(X) \vee SkP^{\psi}_{\varphi}(XYf(XY))$. Enfin, les littéraux contenant $\neg SkP^{\psi}_{\varphi}$ seront notés $\neg SkP^{\psi}_{\varphi}(T^x T^y T^f)$, ce qui nous permet d'écrire $pgu(\langle XYf(XY)\rangle, \langle T^x T^y T^f\rangle) = pgu(T^f, f(T^x T^y))$.

Avant de définir D^*, il faut constater que toute occurrence $\neg SkP^{\psi}_{\varphi}(T^x T^y T^f)$ dans D est unifiable avec $SkP^{\psi}_{\varphi}(XYf(XY))$. Dans le cas contraire, on ne pourrait

[1] l'idée de ce théorème, qui apparaît dans [BdlTC90], a été suggéré par Gilles Chaminade.

[2] en supposant que, si $i < j$, la formule quantifiant y_i est une sous formule de celle quantifiant y_j.

éliminer un tel littéral, puisque C^+ contient l'unique occurrence positive de SkP^ψ_φ, et D ne saurait alors être une réfutation.

Dans une première étape, on transforme les clauses dans D de la façon suivante :

$$(\neg SkP^\psi_\varphi(T^x_1 T^y_1 T^f_1) \vee .. \vee \neg SkP^\psi_\varphi(T^x_n T^y_n T^f_n) \vee C)^* = \sigma(\neg SkP^\psi_{\varphi'}(T^x_1) \vee .. \vee \neg SkP^\psi_{\varphi'}(T^x_n) \vee C)$$

où $\sigma = pgu(\langle T^f_1 .. T^f_n \rangle, \langle f(T^x_1 T^y_1) .. f(T^x_n T^y_n) \rangle)$, et SkP^ψ_φ n'apparaît pas dans C. Si $n = 0$, on a bien entendu $\sigma = Id$; ne sont modifiées que les clauses contenant SkP^ψ_φ.

La deuxième étape consiste à éliminer les résolutions sur C^+. On constate en effet que la transformation précédente identifie les clauses résolues avec C^+ et leur résolventes :

$$\frac{C^+ \qquad \neg SkP^\psi_\varphi(T^x_1 T^y_1 T^f_1) \vee .. \vee \neg SkP^\psi_\varphi(T^x_n T^y_n T^f_n) \vee C}{\sigma_1(\neg SkP^\psi_{\varphi'}(T^x_1) \vee \neg SkP^\psi_\varphi(T^x_2 T^y_2 T^f_2) \vee .. \vee \neg SkP^\psi_\varphi(T^x_n T^y_n T^f_n) \vee C)}$$

où $\sigma_1 = pgu(T^f_1, f(T^x_1 T^y_1))$, est transformé en :

$$\frac{C^+ \qquad \sigma(\neg SkP^\psi_{\varphi'}(T^x_1) \vee .. \vee \neg SkP^\psi_{\varphi'}(T^x_n) \vee C)}{\sigma'\sigma_1(\neg SkP^\psi_{\varphi'}(T^x_1) \vee .. \vee \neg SkP^\psi_{\varphi'}(T^x_n) \vee C)}$$

avec

$$\begin{aligned} \sigma &= pgu(\langle T^f_1 .. T^f_n \rangle, \langle f(T^x_1 T^y_1) .. f(T^x_n T^y_n) \rangle) \\ \sigma' &= pgu(\sigma_1 \langle T^f_2 .. T^f_n \rangle, \sigma_1 \langle f(T^x_2 T^y_2) .. f(T^x_n T^y_n) \rangle) \end{aligned}$$

et on a donc $\sigma'\sigma_1 = \sigma$, ce qui prouve l'identité énoncée.

Il reste maintenant à montrer que les pas de résolution et de factorisation dans D^* sont corrects. Nous montrons en fait que la transformée des résolventes (resp. des facteurs) est *subsumée* par la résolvente (resp. le facteur) des transformées. On commence ici par les pas de factorisations, et nous considérons d'abord le cas où le littéral factorisé n'est pas SkP^ψ_φ :

$$\frac{\neg SkP^\psi_\varphi(T^x_1 T^y_1 T^f_1) \vee .. \vee \neg SkP^\psi_\varphi(T^x_n T^y_n T^f_n) \vee P(T) \vee P(T') \vee C}{\sigma'(\neg SkP^\psi_\varphi(T^x_1 T^y_1 T^f_1) \vee .. \vee \neg SkP^\psi_\varphi(T^x_n T^y_n T^f_n) \vee P(T) \vee C)}$$

où T et T' sont des listes de termes, et $\sigma' = pgu(T, T')$, est transformé en :

$$\frac{\sigma(\neg SkP^\psi_{\varphi'}(T^x_1) \vee .. \vee \neg SkP^\psi_{\varphi'}(T^x_n) \vee P(T) \vee P(T') \vee C)}{\sigma''\sigma'(\neg SkP^\psi_{\varphi'}(T^x_1) \vee .. \vee \neg SkP^\psi_{\varphi'}(T^x_n) \vee P(T) \vee C)}$$

avec

$$\begin{aligned}\sigma &= pgu(\langle T_1^f..T_n^f\rangle, \langle f(T_1^x T_1^y)..f(T_n^x T_n^y)\rangle)\\ \sigma'' &= pgu(\sigma'\langle T_1^f..T_n^f\rangle, \sigma'\langle f(T_1^x T_1^y)..f(T_n^x T_n^y)\rangle)\end{aligned}$$

et donc $\sigma''\sigma'$ est un unificateur de $\langle T_1^f..T_n^f\rangle$ et $\langle f(T_1^x T_1^y)..f(T_n^x T_n^y)\rangle$, et est donc moins général que σ ; la deuxième clause est donc bien une instance de la première, excepté pour la disparition de $P(T')$. Par ailleurs, on a $\sigma''\sigma' T = \sigma''\sigma' T' < \sigma T'$, donc $\sigma''\sigma' T$ est bien une instance commune de σT et $\sigma T'$. Le pas de factorisation est donc correct. Comme dans la suite, nous ne prouvons pas ici que $\sigma''\sigma'$ est l'unificateur le plus général de σT et $\sigma T'$. Nous avons donc prouvé que la deuxième clause est une instance du facteur — par les littéraux considérés — de la première.

Nous considérons maintenant le cas où la factorisation porte sur SkP_φ^ψ :

$$\frac{\neg SkP_\varphi^\psi(T_1^x T_1^y T_1^f)\vee..\vee\neg SkP_\varphi^\psi(T_n^x T_n^y T_n^f)\vee C}{\sigma'(\neg SkP_\varphi^\psi(T_2^x T_2^y T_2^f)\vee..\vee\neg SkP_\varphi^\psi(T_n^x T_n^y T_n^f)\vee C)}$$

où $\sigma' = pgu(\langle T_1^x T_1^y T_1^f\rangle, \langle T_2^x T_2^y T_2^f\rangle)$, est transformé en :

$$\frac{\sigma(\neg SkP_{\varphi'}^\psi(T_1^x)\vee..\vee\neg SkP_{\varphi'}^\psi(T_n^x)\vee C)}{\sigma''\sigma'(\neg SkP_{\varphi'}^\psi(T_2^x)\vee..\vee\neg SkP_{\varphi'}^\psi(T_n^x)\vee C)}$$

avec

$$\begin{aligned}\sigma &= pgu(\langle T_1^f..T_n^f\rangle, \langle f(T_1^x T_1^y)..f(T_n^x T_n^y)\rangle)\\ \sigma'' &= pgu(\sigma'\langle T_2^f..T_n^f\rangle, \sigma'\langle f(T_2^x T_2^y)..f(T_n^x T_n^y)\rangle)\end{aligned}$$

On a donc :

$$\begin{aligned}\sigma''\sigma'\langle T_1^f..T_n^f\rangle &= \sigma''\sigma'\langle T_2^f T_2^f..T_n^f\rangle\\ &= \sigma''\sigma'\langle f(T_2^x T_2^y) f(T_2^x T_2^y)..f(T_n^x T_n^y)\rangle\\ &= \sigma''\sigma'\langle f(T_1^x T_1^y)..f(T_n^x T_n^y)\rangle\end{aligned}$$

$\sigma''\sigma'$ unifie donc les mêmes termes que σ, donc $\sigma''\sigma' < \sigma$, ce qui prouve que la deuxième clause est une instance de la première. Enfin, on a $\sigma''\sigma' T_1^x = \sigma''\sigma' T_2^x < \sigma T_2^x$, donc $\sigma''\sigma' T_2^x$ est une instance commune de σT_1^x et σT_2^x.

Nous nous intéressons maintenant aux pas de résolution : la clause

$$\neg SkP_\varphi^\psi(T_1^x T_1^y T_1^f)\vee..\vee\neg SkP_\varphi^\psi(T_n^x T_n^y T_n^f)\vee C\vee P(T)$$

est résolue avec la clause

$$\neg P(T') \vee C' \vee \neg SkP^{\psi}_{\varphi}(T'^{x}_{1} T'^{y}_{1} T'^{f}_{1}) \vee .. \vee \neg SkP^{\psi}_{\varphi}(T'^{x}_{n'} T'^{y}_{n'} T'^{f}_{n'})$$

pour donner la résolvente

$$\sigma_1(\neg SkP^{\psi}_{\varphi}(T^{x}_{1} T^{y}_{1} T^{f}_{1}) \vee .. \vee \neg SkP^{\psi}_{\varphi}(T^{x}_{n} T^{y}_{n} T^{f}_{n}) \vee C \vee C' \vee \neg SkP^{\psi}_{\varphi}(T'^{x}_{1} T'^{y}_{1} T'^{f}_{1}) \vee .. \vee \neg SkP^{\psi}_{\varphi}(T'^{x}_{n'} T'^{y}_{n'} T'^{f}_{n'}))$$

où T et T' sont des listes de termes, et $\sigma_1 = pgu(T, T')$. Dans D^*, on trouve à la place de ce pas de résolution l'inférence suivante :

$$\frac{\sigma(\neg SkP^{\psi}_{\varphi'}(T^{x}_{1}) \vee .. \vee \neg SkP^{\psi}_{\varphi'}(T^{x}_{n}) \vee C) \qquad \sigma'(\neg SkP^{\psi}_{\varphi'}(T'^{x}_{1}) \vee .. \vee \neg SkP^{\psi}_{\varphi'}(T'^{x}_{n'}) \vee C')}{\sigma''\sigma_1(\neg SkP^{\psi}_{\varphi'}(T^{x}_{1}) \vee .. \vee \neg SkP^{\psi}_{\varphi'}(T^{x}_{n}) \vee C \vee C' \vee \neg SkP^{\psi}_{\varphi'}(T'^{x}_{1}) \vee .. \vee \neg SkP^{\psi}_{\varphi'}(T'^{x}_{n'}))}$$

avec

$$\begin{aligned}
\sigma &= pgu(\langle T^{f}_{1} .. T^{f}_{n} \rangle, \langle f(T^{x}_{1} T^{y}_{1}) .. f(T^{x}_{n} T^{y}_{n}) \rangle) \\
\sigma' &= pgu(\langle T'^{f}_{1} .. T'^{f}_{n} \rangle, \langle f(T'^{x}_{1} T'^{y}_{1}) .. f(T'^{x}_{n} T'^{y}_{n}) \rangle) \\
\sigma'' &= pgu(\sigma_1\langle T^{f}_{1} .. T^{f}_{n} T'^{f}_{1} .. T'^{f}_{n} \rangle, \sigma_1\langle f(T^{x}_{1} T^{y}_{1}) .. f(T^{x}_{n} T^{y}_{n}) f(T'^{x}_{1} T'^{y}_{1}) .. f(T'^{x}_{n} T'^{y}_{n}) \rangle)
\end{aligned}$$

donc $\sigma''\sigma_1$ est un unificateur des mêmes termes que σ, et il est donc moins général que σ : on a donc $\sigma''\sigma_1 = \sigma''\sigma_1\sigma$. De même pour σ', on obtient $\sigma''\sigma_1 = \sigma''\sigma_1\sigma'$. Par ailleurs, on a $\sigma''\sigma_1\sigma T = \sigma''\sigma_1 T = \sigma''\sigma_1 T' = \sigma''\sigma_1\sigma' T'$. Ceci prouve que l'on a bien un pas de résolution correct, avec l'unificateur $\sigma''\sigma_1$ de σT et $\sigma' T'$.

D^* est donc une réfutation correcte, et plus courte que D. De plus, les clauses non initiales dans D^* sont subsumées par la résolvente (resp. le facteur) des clauses parentes ; il existe donc une réfutation plus courte que D^* à partir des mêmes clauses initiales, qui sont celles de $clausale(Rnm(R \cup \varphi', \psi))$. $\dashv$

Nous n'avons énoncé ce théorème que dans le cas où $pol(\varphi, \psi) = 1$. Il est cependant évident qu'il est également vrai si $pol(\varphi, \psi) = -1$. Nous pouvons en conclure que ce théorème s'applique lorsqu'il n'y a sur le chemin de φ à φ' que des quantificateurs, existentiels ou universels, des négations et des conjonctions ; tous connectifs qui ne multiplient pas le nombre d'occurrences de SkP^{ψ}_{φ} dans la forme clausale. Enfin, il faut constater que les renommages R_{inf} ne contiennent jamais de telles sous-formules, puisque $p(Rnm(R \cup \{\varphi'\}, \psi)) < p(Rnm(R \cup \{\varphi, \varphi'\}, \psi))$. Il en est de même des renommages R'_{inf}, car $q(Rnm(R \cup \{\varphi'\}, \psi)) < q(Rnm(R \cup \{\varphi, \varphi'\}, \psi))$

7.4 A propos des équivalences

Les résultats les plus intéressants que nous avons obtenus jusqu'ici ont la particularité de ne pas être vérifiés en présence d'équivalences. Le problème du renommage des sous-formules de polarité nulle reste donc entier. Si l'exemple exposé en section 7.2 montre qu'il peut-être néfaste de renommer une telle sous-formule, cela semble être l'exception. Un critère apparemment important est le nombre de duplications d'une sous-formule lors de la transformation *clausale* qui, on le sait, double à chaque équivalence traversée. Après renommage d'une sous-formule de polarité nulle, celle-ci n'est plus surmontée que d'une équivalence, ce qui suggère de renommer systématiquement les sous-formules surmontées d'au moins deux équivalences. Une telle règle doit évidemment être appliquée selon une stratégie descendante : si φ est une sous-formule de ψ vérifiant cette condition, il en est de même de toute sous-formule $\varphi' \sqsubset \varphi$. Cependant, si $pol(\varphi', \varphi) \neq 0$, φ' n'est plus surmontée que d'une équivalence dans $Rnm(\varphi, \psi)$, et il ne faut donc pas la renommer.

Bien entendu, le nombre de clauses (resp. de littéraux) double également à chaque équivalence traversée, ce qui montre que les renommages R_{inf} (resp. R'_{inf}) respectent dans une large mesure la règle énoncée. De plus la stratégie descendante se trouve ici pleinement justifiée, contrairement à ce que semblait suggérer l'exemple donné au début du chapitre 4.

Il convient ici de faire une petite remarque concernant la réfutation des formules dont un conjoint est une équivalence : $C \wedge (\varphi_1 \Leftrightarrow \varphi_2)$. On sait en effet qu'une disjonction close est insatisfaisable si et seulement si chaque disjoint est insatisfaisable. Au lieu donc de linéariser en $C \wedge (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$, il peut être intéressant de linéariser en $C \wedge ((\varphi_1 \wedge \varphi_2) \vee (\neg\varphi_1 \wedge \neg\varphi_2))$, pour ramener la réfutation de cette formule aux réfutations de $C \wedge \varphi_1 \wedge \varphi_2$ et $C \wedge \neg\varphi_1 \wedge \neg\varphi_2$, ce qui évite de calculer — et de réfuter — les formes clausales de $\varphi_1 \Rightarrow \varphi_2$ et $\varphi_2 \Rightarrow \varphi_1$. Bien entendu, cette technique est particulièrement intéressante lorsqu'on a $C = \emptyset$, et des formules φ_1 et φ_2 relativement complexes.

7.5 Factorisation contre résolution

Nous avons montré que les renommages R_{inf} contiennent toujours moins de formules que *struct_pres*, ce qui a pour conséquence d'éventuelles duplications de sous-formules pendant le calcul de $clausale(Rnm(R_{inf}(\psi), \psi))$, alors qu'il n'y en a jamais dans le calcul de la forme clausale structure preserving (sauf pour les

sous-formules de polarité nulle). Les sous-formules dupliquées peuvent alors avoir à être factorisées pendant la réfutation. L'expérience montre que la réfutation de $clausale(Rnm(R_{inf}(\psi),\psi))$ a souvent moins de pas de résolution et plus de pas de factorisation que celle de $clausale(Rnm(struct\_pres(\psi),\psi))$.

Dans le cas propositionnel, cet argument va en faveur des renommages R_{inf}, car les pas de factorisation peuvent être effectués plus efficacement que les pas de résolution. On ne saurait être aussi affirmatif en ce qui concerne le premier ordre. L'exemple suivant[3] montre que sans éliminer des duplication de sous-formules (puisqu'il n'y en a pas ici), des renommages qui font croître le nombre de clauses, ainsi que le nombre de littéraux, peuvent supprimer de nombreux pas de factorisation.

On note $C_n = \forall x_1.(P_1(x_1) \vee \forall x_2.(P_2(x_1x_2) \vee .. \vee \forall x_n.P_n(x_1..x_n)..))$, et $B_i = \forall x_1..x_{i-1}.\neg P_i(x_1..x_{i-1}a) \vee \neg P_i(x_1..x_{i-1}b)$. On considère enfin la formule $F_n = C_n \wedge \bigwedge_{i=1}^{n} B_i$. On remarque que $clausale(F_n) = prénexe(F_n)$, ce qui montre que $R_{inf}(F_n) = R'_{inf}(F_n) = \emptyset$. On peut obtenir une réfutation exponentielle de F_n si l'on décide de résoudre P_i avant P_{i+1} mais nous nous intéressons ici à la réfutation la plus courte, obtenue avec l'ordre inverse. Celle-ci procède comme suit :

$$\dfrac{\dfrac{\dfrac{C_i \qquad B_i}{C_{i-1} \vee \neg P_i(x_1..x_{i-1}b)} \qquad C_i}{\dfrac{C_{i-1} \vee C_{i-1}}{C_{i-1}}}}{} \; i-1 \text{ factorisations}$$

pour $i = n$ à $i = 2$, et :

$$\dfrac{\dfrac{C_1 \qquad B_1}{\neg P_1(b)} \qquad C_1}{\square}$$

ce qui fait $2n$ pas de résolution et $\sum_{i=2}^{n} i-1 = \frac{1}{2}n(n-1)$, donc $O(n^2)$ inférences.

Si maintenant on renomme les quantificateurs universels dans C_n, en évitant évidemment le premier, on obtient une nouvelle forme clausale. Dans la suite, on note SkP_i^n le prédicat de Skolem correspondant au quantificateur de x_{i+1}, pour $i = 1..n-1$. On note R_n le renommage obtenu. Enfin, pour $i = 1..n-2$, D_i dénote la clause résultant de la définition de SkP_i^n, c'est à dire $\forall x_1..x_{i+1}.\neg SkP_i^n(x_1..x_i) \vee P_{i+1}(x_1..x_{i+1}) \vee SkP_{i+1}^n(x_1..x_{i+1})$, et pour $i = 1..n-1$, A_i dénote la clause $\forall x_1..x_{i+1}.\neg SkP_i^n(x_1..x_i) \vee P_{i+1}(x_1..x_{i+1})$, et donc A_{n-1} est la

[3] proposé par Gilles Chaminade, et qui figure dans [BdlTC90]

clause correspondant à la définition de SkP^n_{n-1}. On a alors :

$$clausale(Rnm(R_n, F_n)) = (\forall x_1.P_1 \vee SkP^n_1(x_1)) \wedge A_{n-1} \wedge \bigwedge_{i=1}^{n-2} D_i \wedge \bigwedge_{i=1}^{n} B_i$$

Pour simplifier, on note $D_0 = \forall x_1.P_1(x_1) \vee SkP^n_1(x_1)$, et $A_0 = \forall x_1.P_1(x_1)$. La réfutation procède alors de la façon suivante :

$$\dfrac{\dfrac{\dfrac{A_i \qquad B_{i+1}}{\neg SkP^n_i(x_1..x_i) \vee \neg P_{i+1}(x_1..x_i b)} \qquad A_i}{\neg SkP^n_i(x_1..x_i) \vee \neg SkP^n_i(x_1..x_i)}}{\dfrac{\neg SkP^n_i(x_1..x_i) \qquad D_{i-1}}{A_{i-1}}}$$

pour $i = n-1$ à $i = 1$. On termine avec :

$$\dfrac{\dfrac{A_0 \qquad B_1}{\neg P_1(b)} \qquad A_0}{\square}$$

ce qui fait $3n-1$ pas de résolution et $n-1$ pas de factorisation, et donc une réfutation en $O(n)$ inférences. Il faut noter que R_n peut-être obtenu à partir de $struct\_pres(F_n)$ en appliquant les restrictions autorisées par le théorème 7.3.

7.6 A propos des simplifications

Lorsque dans R_{inf} ou R'_{inf} nous calculons le nombre de clauses ou de littéraux, il est évident que nous ne tenons pas compte des possibilités de simplification. Ceci est une des conditions de l'efficacité du calcul de ces transformations. Bien entendu, il serait erroné d'en conclure qu'il ne faut pas effectuer de simplifications lorsqu'on renomme : cette technique est toujours intéressante, sinon indispensable, mais le problème est ici que, loin d'introduire des possibilités de simplification, le renommage peut en supprimer.

Considérons par exemple la formule $\psi = (\neg A \wedge B) \vee A$, dont la forme clausale, $(\neg A \vee A) \wedge (B \vee A)$, contient une clause tautologique, donc simplifiable. Si on y renomme la conjonction, on obtient la forme clausale $(SkP^{\psi}_{\neg A \wedge B} \vee A) \wedge (\neg SkP^{\psi}_{\neg A \wedge B} \vee \neg A) \wedge (\neg SkP^{\psi}_{\neg A \wedge B} \vee B)$, qui n'est plus simplifiable. De plus, la résolvente des deux premières clauses sur A est une tautologie, qu'il faut donc effacer, ce qui rend le calcul de cette résolvente inutile, et contribue à un gaspillage du temps de calcul.

Une solution relativement simple à ce problème est d'effectuer *avant* le renommage des simplifications non clausales (voir [VG84] pour le calcul propositionnel :

ces simplifications ont été étendues au premier ordre par Gilles Chaminade). Cette méthode ne doit d'ailleurs pas être négligée lorsqu'on effectue des simplifications sur la forme clausale : aucune de ces deux techniques ne subsume l'autre. Une formule simplifiée peut produire des clauses simplifiables, et ces simplifications non clausales peuvent faire disparaître (ou modifier) des clauses qui n'auraient pas été individuellement simplifiables dans le cas contraire.

Bien que difficile à programmer[4], cette méthode s'est révélée très efficace. Ainsi, tous les théorèmes expérimentés dans [PG86] sont simplifiés de telle façon qu'ils en deviennent extrêmement simples, au point de rendre tout renommage inutile, et ce en un temps très largement inférieur à ceux obtenus dans [PG86], que ce soit avec ou sans renommage.

Comme nous l'avons dit, cela ne rend pas caduques les techniques de simplifications clausales, d'autant que les simplifications de [VG84] ne s'appliquent pas en dessous des équivalences. Nous pouvons en conclure qu'il convient d'éviter les renommages qui ne seraient pas absolument nécessaires, ce qui est donc à l'avantage des renommages R_{inf} et R'_{inf} par rapport à *struct_pres*. Cette remarque justifie également le choix de PR^{+}_{inf} plutôt que PR_{inf}.

7.7 Quelques résultats expérimentaux

En tenant compte des remarques qui précèdent, nous avons appliqué bon nombre de transformations différentes sur un même théorème, considéré comme difficile (voir [HLO+80]), et qui a été proposé par Peter Andrews pour illustrer l'inadéquation de la méthode de résolution dans certains cas. Il est donc devenu un test classique pour les démonstrateurs par résolution, ainsi que pour les transformateurs de formules sous forme clausale. Le problème est le suivant :

$$\begin{aligned} I \;=\; & \neg[\exists x.\forall y.(P(x) \Leftrightarrow P(y)) \Leftrightarrow (\exists x.Q(x) \Leftrightarrow \forall y.P(y)) \\ & \Leftrightarrow \exists x.\forall.y(Q(x) \Leftrightarrow Q(y)) \Leftrightarrow (\exists x.P(x) \Leftrightarrow \forall y.Q(y))] \end{aligned}$$

Nous avons effectué trois ensembles d'expérimentations (voir table 7.1). Dans le premier ensemble (I), nous avons appliqué la technique de linéarisation de

[4] le simplificateur non clausal de ATINF, de même que le démonstrateur par résolution, ont été réalisés par Gilles Chaminade. Les autres transformations, y compris les renommages, l'ont été par moi-même. Ce travail a bien entendu été indispensable à la rédaction de ce chapitre, sinon même à la validation des résultats obtenus dans tous les précédents, et ne doit donc pas être occulté par ceux-ci.

l'équivalence dominante exposée à la fin de la section 7.4. Les deux sous-problèmes sont alors transformés avec les renommages indiqués.

Dans le deuxième ensemble (II), nous avons appliqué directement ces renommages sur la formule globale. Dans le troisième ensemble (III), nous avons tout d'abord renommé systématiquement les sous-formules dont la profondeur en équivalence dépassait 2 (sauf bien sûr les formules atomiques et les négations), suivant en cela les considérations exposées en 7.4. Le résultat est alors linéarisé, puis les renommages indiqués sont appliqués avant la mise sous forme clausale.

Les expérimentations I et II sont divisées en deux parties : dans la première (1), les renommages sont appliqués directement, tandis que dans la seconde (2), la formule est d'abord linéarisée avant d'être renommée et mise sous forme clausale, ce qui constitue, dans le pire des cas, une transformation exponentielle. Cette tentative a pour but de tester, sur un cas pratique, l'importance du critère que constitue la complexité en taille.

Dans chaque cas, trois renommages sont effectués : le renommage $\emptyset$, de façon à obtenir la forme clausale standard, le renommage R_{opt} et le renommage sp^+, qui est une amélioration de *struct_pres* tenant compte du théorème 7.3 pour les sous-formules de polarité non nulle. Par contre, elle en diffère fortement en dessous des équivalences, car on ne renomme jamais qu'un côté des équivalence : cette transformation est donc exponentielle, sauf bien sûr dans l'expérimentation III.

Les expérimentations sont effectuées avec notre démonstrateur (écrit en Lucid Common Lisp, voir [BdlTCC88]) sur un SUN3/60-8MB. La stratégie est toujours la même ; un ordonnancement des prédicats respectant les deux règles données section 7.3, où de plus P est résolu avant Q, ce qui explique l'asymétrie des résultats obtenus pour les sous-problèmes de l'expérimentation I. Nous n'avons pas essayé de trouver une stratégie efficace pour ce problème particulier : il s'agit simplement de la stratégie la plus simple, que nous avons adoptée pour bon nombre d'exemple, la plupart étant plus simple, d'autres trop difficiles, donc moins discriminant que celui-ci.

Nous considérons tout d'abord les résultats de l'expérimentation I.2. Les deux sous-problèmes à réfuter sont relativement simples, et ne contiennent que peu d'équivalences. Leur forme clausale standard est donc limitée (on remarque que R_{opt} n'effectue aucun renommage) et peut donc être réfutée facilement. Cependant, lorsque les sous-problèmes, après linéarisation, sont renommés par sp^+, le nombre de clauses double, et 20 nouveaux symboles de prédicats sont introduits : il en résulte

Table 7.1: Résultats

		p	$\|R\|$	t	r	$\bar{n}$	f
I.1	$\emptyset$	16 + 16	0 + 0	8.3 + 6.4	12 + 10	11.5 + 10.5	25% + 30%
	R_{opt}	16 + 16	0 + 0	8.3 + 6.4	12 + 10	11.5 + 10.5	25% + 30%
	sp^+	16 + 16	2 + 2	241 + 347	19 + 14	60 + 93	2.4% + 2.9%
I.2	$\emptyset$	16 + 16	0 + 0	8.3 + 6.4	12 + 10	11.5 + 10.5	25% + 30%
	R_{opt}	16 + 16	0 + 0	8.3 + 6.4	12 + 10	11.5 + 10.5	25% + 30%
	sp^+	32 + 32	20 + 20	46 + 45	28 + 28	26.8 + 26.4	10.5% + 10.5%
II.1	$\emptyset$	128	0	—	—	—	—
	R_{opt}	24	3	28.3	19	27.5	10.2%
	sp^+	24	3	587	26	60	3.2%
II.2	$\emptyset$	128	0	—	—	—	—
	R_{opt}	32	2	816	36	62	2.7%
	sp^+	66	44	—	—	—	—
III	$\emptyset$	24	4	27.5	22	23.2	10.5%
	R_{opt}	24	4	27.5	22	23.2	10.5%
	sp^+	46	26	200	50	26.3	10.1%

p est le nombre de clauses, $|R|$ le nombre de sous-formules renommées, t la durée de réfutation en seconde, r le nombre de "runs", c'est à dire de calcul de toutes les résolventes à un niveau, $\bar{n}$ le nombre moyen de clauses engendrées à chaque niveau, et f est le *focus*, c'est à dire le rapport du nombre de clauses utilisées dans la preuve au nombre total de clauses produites.

une forte diminution du focus, d'où la dégradation de performance enregistrée.

Le même accroissement du nombre de clauses, et du nombre de littéraux de Skolem avec le renommage sp^+ est constaté dans l'expérimentation III, ce qui mène également à de pauvres performances, bien que le focus ne soit pas ici diminué. Il en est de même dans l'expérience II.2, ce qui suggère que sp^+ s'accomode assez mal d'une linéarisation préliminaire, même si la profondeur en équivalence est limitée à 2, que se soit naturellement (I.2) ou après renommage (III).

Considérons maintenant l'expérimentation II.1, qui est la plus simple : les renommages sont appliqués directement sur I, la mise sous forme clausale est effectuée ensuite. Sur ce problème conséquent, le renommage sp^+ se trouve justifié par rapport à la forme clausale standard, bien que le focus reste faible : la différence provient principalement du nombre de clauses. Par contre, bien que l'on obtienne avec R_{opt} le même nombre de clauses, avec le même nombre de sous-formules renommées, qu'avec sp^+, les performances sont très nettement améliorées. Cela provient de ce que R_{opt} crée des définitions ayant moins de variables libres (ceci provenant de la stratégie descendante). Les bonnes instances sont alors rapidement trouvées, alors qu'avec sp^+, les clauses provenant des définitions contiennent des littéraux non clos, qui peuvent se résoudre sur un grand nombre de littéraux complémentaires, ce qui explique un fort manque de focus. C'est principalement ce phénomène qui explique les mauvaises performances de sp^+ dans l'expérimentation I.1, où le nombre de clauses n'est pas modifié par les deux renommages effectués, qui résultent cependant en un net déficit en focus.

Pour ce qui est de la comparaison des différents renommages, il semble donc que R_{opt} sache effectuer des renommages judicieux, tout en restant suffisamment économe sur ce point pour éviter une dégradation des performances qui pourrait résulter d'un trop grand nombre de définitions.

Nous pouvons également comparer les différentes expérimentations entre elles, pour constater tout d'abord l'intérêt de la séparation en deux sous-problèmes indépendants (I), chacun devenant beaucoup plus simple que le problème complet. On ne peut alors établir, sur des problèmes aussi simples, de distinction entre les sous-ensembles I.1 et I.2.

Par contre, la différence entre II.1 et II.2 est très nette : dès que la profondeur en équivalence dépasse 2, une linéarisation préliminaire — donc exponentielle dans le pire des cas — mène à des résultats catastrophiques. L'expérimentation III montre qu'il est en fait très intéressant d'effectuer un renommage qui se contente de

limiter la profondeur en équivalence à 2, et que l'on peut alors linéariser avant de renommer, ce qui permet d'assurer l'optimalité du nombre de clauses avec R_{opt} — qui ici n'effectue aucun renommage supplémentaire.

Enfin, nous pouvons ajouter que des résultats forts similaires à ceux de R_{opt} ont été obtenus avec R'_{inf}. Une comparaison entre ces deux renommages nécessiterait une étude plus poussée.

Nous pouvons finalement établir une comparaison avec d'autres travaux concernant le problème de Andrews (les plus récents sont [GZ89] et [Qua90]). Nous ne pouvons comparer les temps d'exécution, puisque les démonstrateurs et ordinateurs utilisés sont différents, et nous nous intéressons donc uniquement aux transformations du problème de Andrews sous forme clausale. Le nombre de clauses que nous obtenons — 24 — constitue à ce jour le record dans la catégorie : il a été atteint indépendemment par [Qua90], également avec 3 renommages. La façon dont ces renommages sont calculés n'est pas clairement décrite. Pour autant que l'on puisse en juger (i.e. à partir d'une variante du problème de Andrews, qui est plus simple et dont la transformation est mieux décrite), les formules renommées sont produites par un parcours ascendant, et doivent donc être plus petites que, ou sont des sous-formules de, celles que nous obtenons avec R_{opt}, et donc plus proche des formules obtenues avec sp^+.

Conclusion

Et pressentant violemment la voile.

Rimbaud

Nous obtenons donc, en plus des résultats théoriques concernant les propriétés syntaxiques de R_{opt}, des résultats expérimentaux très satisfaisant, surtout en considérant que R_{opt} n'a pas été écrite "en fonction" du problème de Andrews (comme c'est souvent le cas des systèmes qui parviennent à le résoudre), ni même en fonction des résultats de la section 7.3. Si l'intérêt de R_{opt} ne peut vraiment être démontré que par une longue pratique, ces deux faits suggèrent que nous avons adopté un critère de renommage pertinent, qui semble bien à même d'améliorer les performances des démonstrateurs par résolution sur une majorité de théorèmes. Cette amélioration peut être considérable si l'on compare avec la forme clausale standard, du fait — entre autres — de l'élimination du comportement exponentiel de celle-ci.

Faut-il, alors, rechercher un algorithme de renommage qui serait encore plus performant que R_{opt} (par exemple R'_{inf}) ? Il semble qu'on ne peut guère attendre dans cette direction une amélioration aussi importante que celle obtenue en passant de la forme clausale standard à R_{opt}, et qu'il est pour l'instant plus prometteur d'étudier les relations qui existent entre le renommage et les autres étapes de la mise sous forme clausale, principalement la skolemisation, la linéarisation et la simplification.

Nous avons vu, au cours des expérimentations, l'intérêt qu'il peut y avoir à minimiser le nombre de variables libres des formules renommées — ce qui est un des avantages du parcours descendant dans R_{opt}. Il faudrait donc, autant que possible, skolemiser avant de renommer. Ceci n'est cependant pas possible en dessous d'un signe d'équivalence : il faut d'abord linéariser, ce qui ne peut se faire qu'après avoir effectué certains renommages, pour éviter la complexité exponentielle de la transformation. Ceci impose d'agencer ces transformations de façon à établir un compromis entre la complexité en taille et le nombre de variables libres des sous-formules renommées.

Par ailleurs, il serait sans doute très intéressant de rechercher dans une formule plusieurs occurrences d'une même sous-formule (si possible en tenant compte des propriétés associatives et commutatives des connectifs $\wedge \vee$ et $\Leftrightarrow$), afin de les renommer par un même prédicat de Skolem. On peut même, comme il est suggéré dans [PG86], rechercher des sous-formules φ qui seraient de la forme $\sigma_\varphi \varphi'$, φ' constituant alors

une généralisation de ces sous-formules, qui aurait seule la nécessité d'être définie (ce mécanisme préserve la propriété de la sous-formule). Ce serait particulièrement intéressant en relation avec les simplifications non clausales, en augmentant considérablement leur champ d'application. Il serait également souhaitable d'étendre ces simplifications aux sous-formules de polarité nulle.

Il semble également indispensable d'étendre notre travail de façon à y intégrer l'égalité. Cependant, le parti que l'on pourrait tirer des propriétés de l'égalité en faveur du renommage n'est pas évident. Une possibilité est d'effectuer un autre type de renommage : celui de sous-termes par de nouvelles constantes (ou par un terme $f(x_1..x_n)$ s'il y a des variables libres), que l'on définirait en introduisant une nouvelle équation. Cela reste assez éloigné du renommage de sous-formules. Une autre possibilité, plus en rapport avec celui-ci, serait d'exploiter les égalités pour la recherche de sous-formules identiques, ou instances l'une de l'autre, etc, ainsi que pour les simplifications (voir [BdlTC90]).

Au delà de la forme clausale, il serait intéressant d'étudier plus profondément l'influence des renommages sur les réfutations. Nous avons obtenu en section 7.3 des résultats négatifs sur certains renommages (qu'il semble difficile d'étendre), mais peut-on obtenir des résultats positifs, décrire ce qui se passe lorsqu'on renomme un disjoint, estimer le temps que cela peut faire gagner — ou perdre? Nous avons vu en introduction que le renommage constitue une variante de la règle d'extension de Tseitin. Il est montré dans [Tse68] que, dans le calcul propositionnel, on ne peut majorer la longueur des réfutations par une puissance de la longueur des réfutations minimales que l'on peut obtenir en utilisant la règle d'extension. Celle-ci peut donc permettre de raccourcir exponentiellement une réfutation. Ce résultat ne s'applique évidemment pas au renommage, puisqu'il provient d'une restriction de la règle d'extension.

Le problème qui se pose alors est de comparer la complexité des plus courtes preuves obtenues par renommage avec celles obtenues sans renommage, et avec celles obtenues avec la règle d'extension. Il est probable que les preuves obtenues par renommage ne peuvent elles-même être majorées, en taille, par une puissance de la taille des réfutation obtenues avec la règle d'extension, et que le renommage est loin d'atteindre la puissance de la règle d'extension. Mais le renommage permet-il une amélioration de la complexité des plus courtes preuves, par rapport à la seule règle de résolution? Même si l'amélioration n'était que polynômiale, cela constituerait un résultat extrêmement intéressant en démonstration automatique, et assurerait

l'intérêt théorique du renommage, qui n'est jusqu'à maintenant que suggéré par ses excellentes propriétés au niveau de la transformation sous forme clausale, ainsi que par des résultats expérimentaux.

Bibliographie

[BdlT89] Thierry Boy de la Tour. A locally optimal transformation into clause form using partial formula renaming. Rr 765-i-imag - 90 lifia, institut IMAG, BP 68, 38402 Saint Martin d'Hères cedex, January 1989.

[BdlT90] Thierry Boy de la Tour. Minimizing the number of clauses by renaming. In Mark E. Stickel, editor, *Proceedings of the 10th International Conference on Automated Deduction*, pages 558–572. Springer Lecture Notes in Artificial Intelligence 449, July 1990.

[BdlTC90] Thierry Boy de la Tour and Gilles Chaminade. The use of renaming to improve the efficiency of clausal theorem proving. In P. Jorrand and V. Sgurev, editors, *AIMSA '90, Artificial Intelligence—Methodology Systems Application.* North-Holland, 1990.

[BdlTCC88] Thierry Boy de la Tour, Ricardo Caferra, and Gilles Chaminade. Some tools for an inference laboratory (atinf). In E. Lusk and R. Overbeek, editors, *Proccedings of the 9th International Conference on Automated Deduction*, pages 744–745. Springer Lecture Notes in Computer Science 310, 1988.

[CK73] C. C. Chang and H. J. Keisler. *Model Theory*, volume 73 of *Studies in Logic and the Foundations of Mathematics.* North-Holland, 1973.

[CL73] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, New York, 1973.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the Association for Computing Machinery*, 7(3), 1960.

[Ede84] Elmar Eder. An implementation of a theorem prover based on the connection method. In W. Bibel and B. Petkoff, editors, *AIMSA'84, Artificial Intelligence—Methodology Systems Application*, pages 121–128. North-Holland, September 1984.

[GNOP82] Steven Greenbaum, A. Nagasaka, P. O'Rorke, and David A. Plaisted. Comparison of natural deduction and locking resolution implementations. In D. Loveland, editor, *Proceedings of the 6th Conference on Automated Deduction*, pages 159–171. Springer Lecture Notes in Computer Science 138, 1982.

[GZ89] Angshuman Guha and Hantao Zhang. Andrew's challenge problem: Clause conversion and solutions. *AAR Newsletter*, 14:5–8, December 1989.

[HLO+80] L. Henschen, E. Lusk, R. Overbeek, B.T. Smith, R. Veroff, S. Winker, and L. Wos. challenge problem 1. *SIGART newsletter*, (72):30–31, July 1980.

[Hsi85] Jieh Hsiang. Refutational theorem proving using term-rewriting systems. *Artificial Intelligence*, 25:255–300, 1985.

[JK90] Jean-Pierre Jouannaud and Claude Kirchner. Solving equations in abstract algebras : A rule-based survey of unification. In Jean-Louis Lassez and Gordon Plotkin, editors, *Computational Logic : Essays in honor of Alan Robinson.* MIT-Press, 1990.

[Lov78] Donald W. Loveland. *Automated Theorem Proving: A Logical Basis*, volume 6 of *Fundamental Studies in Computer Science.* North-Holland Publishing Company, 1978.

[Pel86] J.F. Pelletier. Seventy-five problems for testing automatic theorem provers. *Journal of Automated Reasoning*, 2:191–216, 1986.

[PG86] David A. Plaisted and Steven Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation*, 2:293–304, 1986.

[Qua90] Art Quaife. Andrew's challenge problem revisited. *AAR Newsletter*, 15:3–7, May 1990.

[Rob65] J.A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12:23–41, 1965.

[Rus89] Michael Rusinowitch. *De'monstration Automatique : Techniques de Re'e'criture.* Inter Editions, 1989.

[Sko67] Thoralf Skolem. Logico-combinatorial investigation in the satisfiability or provability of mathematical propositions: A simplified proof of a theorem by l. lowenheim and generalizations of the theorem. In J. V. Heijenoort, editor, *From Frege to Godel, a source book in mathematical logic, 1879-1931*, pages 252–263. Harvard University Press, 1967.

[Sti85] M. E. Stickel. An introduction to automated deduction. In W. Bibel and Ph. Jorrand, editors, *Fundamentals of Artificial Intelligence*, pages 75–132. Springer Lecture Notes in Computer Science 138, 1985.

[Tse68] G. S. Tseitin. On the complexity of derivation in propositional calculus. In A. O. Slisenko, editor, *Studies in Constructive Mathematics and Mathematical Logic, Part II*, 1968.

[VG84] Allen Van Gelder. A satisfiability tester for non-clausal propositional calculus. In R.E. Shostak, editor, *Proceedings of the 7th International Conference on Automated Deduction*, pages 101–112. Springer Lecture Notes in Computer Science 170, May 1984.

[Wos88] Larry Wos. *Automated Reasoning: 33 Basic Research Problems.* Prentice-Hall, 1988.

Index

$\sqsubseteq$ 8
$\sqsupseteq$ 8
$\sqsubset$ 8
$\sqsupset$ 8
$\sqcup$ 81
$\lhd$ 82
$\preceq$ 59
ψ^c 8
$|\psi|$ 8
$\models \varphi$ 9
$\psi[R]$ 15

a^{ψ}_{φ} 25, 66
b^{ψ}_{φ} 25, 66
$B(R,\psi)$ 39
$B(\varphi,\psi)$ 39
$B(\varphi,\psi[R])$ 39
$B'(R,\psi)$ 67
c^{ψ}_{φ} 66
$clausale(\varphi)$ 19
d^{ψ}_{φ} 66
$Def_{\psi}(\varphi)$ 15
$d_{\psi}(\varphi)$ 9
$d_{\psi}(R)$ 58
$D_{\Leftrightarrow}(\psi)$ 20
$SFd(\varphi)$ 79
$fnc(\varphi)$ 19
$FV(\varphi)$ 8
$h(\psi)$, $\overline{h}(\psi)$ 26
$h'(\psi)$ 71
$Inf_R(\varphi)$ 14
$\mathcal{L}$ 7
$linéaire(\varphi)$ 18
$\mathcal{M}$ 8
$\mathcal{M} \models \varphi$ 9
$\mathcal{M}, \nu \models \varphi$ 9
$N_{\Leftrightarrow}(\psi)$ 20
$p(\psi)$, $\overline{p}(\psi)$ 23
P^{ψ}_{φ}, $\overline{P}^{\psi}_{\varphi}$ 24
$pgu(t,t')$ 93
φ 8
$\varphi \models \varphi'$ 9
$pol(\varphi,\psi)$ 10
$prénexe(\varphi)$ 19
$PR_{inf}(\psi)$ 76
$PR^{+}_{inf}(\psi)$ 86

ψ 8
$q(\psi)$, $\overline{q}(\psi)$ 64
Q^{ψ}_{φ}, $\overline{Q}^{\psi}_{\varphi}$ 66
$R_{inf}(\psi)$ 41
R'_{inf} 67
R_{opt} 88
$Rnm(R, \psi)$ 15
$SF(\psi)$, $SF^*(\psi)$ 8
$\sigma^s(\varphi, \varphi')$ 10
SkL^{ψ}_{φ}, SkP^{ψ}_{φ} 14
$Sklm(\varphi)$ 18
$Sklm_{\psi}(\varphi)$ 22
$SN(\varphi)$ 42
$SO_{\psi}(\varphi)$, $SO(\psi)$ 21
$struct\_pres(\psi)$ 32
$Sup_R(\varphi)$ 14
$SV_{\psi}(\varphi)$ 21
$V(\psi)$ 22

clause 19
condition de saturation 49
conjoint 8, 68
disjoint 8
expansion conservative 12
forme clausale 19
formule
    close 8
    $\mathcal{L}$-formule 7
    linéaire 18
    satisfaisable 9
    sous-formule 8
    sous-formule directe 8
    sur-formule 8
    valide 9
littéral 19
littéral de Skolem 14
$\mathcal{L}$-modèle 8
polarité 10
prédicat de Skolem 14
renommage 14
    complet dans ψ 41
    q-complet 67
    de φ descendant dans ψ 49
    q-descendant 67
    libre dans ψ 49
    q-libre 67
    optimal dans ψ 41
    q-optimal 67
    p-décroissant 39
    q-décroissant 67
    R_{inf} 41
    R'_{inf} 67
symbole
    de constante 7
    de fonction 7
    de prédicat 7
    propositionnel 7
    de variable 8
$\mathcal{L}$-terme 8
tfm 40

Sommaire

Préface 3

1 Expansion de formules 7

1.1 Notations et préliminaires 7

1.2 Expansion de formules 12

1.3 Le renommage 14

2 La transformation *clausale* 17

2.1 Définition des transformations 17

2.2 Complexités en taille 20

2.2.1 Linéarisation 20

2.2.2 Skolemisation 21

2.2.3 Nombre de clauses 23

2.2.4 Longueur des clauses 26

2.2.5 La forme clausale 28

3 Forme clausale et renommages 31

3.1 Introduction 31

3.2 Le renommage *struct_pres* 34

3.3 Les renommages R_{inf} 39

3.3.1 Définition et propriétés 39

3.3.2 Complexité 44

4 Les renommages descendants **47**
4.1 Stratégies de renommage 47
4.2 Invariance des renommages descendants 50
4.3 Optimalité des renommages descendants 58

5 Minimisation du nombre d'occurrences de littéraux **63**
5.1 Calcul du nombre de littéraux 63
5.2 Propriétés 67
5.2.1 Le théorème fondamental de monotonie 68
5.2.2 Conséquences 69
5.2.3 complexité en taille 70
5.3 Les renommages q-descendants 73

6 Algorithmes de renommage **75**
6.1 Un algorithme pour R_{inf} 75
6.2 Optimalité de PR_{inf} 80
6.3 Modifications pour le calcul de R'_{inf} 83
6.4 Une optimisation 85

7 Comparaisons et expérimentations **89**
7.1 Introduction 89
7.2 Idempotence 90
7.3 Elimination d'inférences triviales 91
7.4 A propos des équivalences 98
7.5 Factorisation contre résolution 98
7.6 A propos des simplifications 100
7.7 Quelques résultats expérimentaux 101

Conclusion **107**

Bibliographie **111**

Index **115**

Résumé

La technique du renommage, appliquée exhaustivement, permet d'obtenir une forme clausale polynômiale. Nous choisissons de l'appliquer partiellement, de façon à minimiser certains critères syntaxiques, principalement le nombre de clauses, tout en conservant une complexité polynômiale. Nous montrons qu'un algorithme efficace permet d'obtenir le nombre optimal de clauses sur les formules linéaires. Enfin, nous étudions l'influence de ces transformations sur les réfutations par la méthode de résolution, autant théoriquement qu'expérimentalement.

mots-clés :

forme clausale, résolution, renommage, nombre de clauses, optimisation.