Conception de Procédures de Décision par Combinaison et Saturation

Duc-Khanh Tran

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR STMIA

Conception de Procédures de Décision par Combinaison et Saturation

THÈSE

présentée et soutenue publiquement le 16 février 2007

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1

(spécialité informatique)

par

Duc-Khanh Tran

Composition du jury

Président :	Dominique Méry
Rapporteurs :	Denis Lugiez, Professeur à l'Université de Provence, France Christopher Lynch, Professeur à l'Université Clarkson, Etats-Unis
Examinateurs :	Silvio Ghilardi, Professeur à l'Université de Milan, Italie Dominique Méry, Professeur à l'Université Henri Poincaré, France Viorica Sofronie-Stokkermans, Habilitée à l'Institut Max-Planck, Allemagne
Directeurs de thèse :	Hélène Kirchner, Directeur de Recherche au CNRS, France Silvio Ranise, Chargé de Recherche à l'INRIA Lorraine, France Christophe Ringeissen, Chargé de Recherche à l'INRIA Lorraine, France

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thloria.

Remerciements

Je voudrais tout d'abord exprimer ma gratitude à Hélène Kirchner, Christophe Ringeissen et Silvio Ranise pour avoir encadré mon travail de thèse. La compétence, la gentillesse et les encouragements de Hélène m'ont été d'une aide précieuse. La rigueur scientifique, la permanente disponibilité et la patience de Christophe m'ont aidé à concrétiser les intuitions à la base de ce travail. L'enthousiasme, la vision pragmatique et la clairvoyance de Silvio ont également apporté des valeurs importantes à mon travail.

Je tiens ensuite à remercier chaleureusement ceux qui ont bien voulu participer au jury de la thèse.

Christopher Lynch qui a accepté de bien vouloir écrire un rapport malgré l'obstacle de la langue, et qui n'a pas hésité à venir des Etats-Unis pour participer à la soutenance. Son travail précurseur a fortement inspiré une grand partie de mon travail.

Denis Lugiez qui a bien voulu s'intéresser à ce travail en acceptant la lourde tâche de rapporteur. Ses commentaires judicieux et ses questions pertinentes me sont très utiles pour la suite de la thèse.

Silvio Ghilardi qui m'a honoré en participant au jury, et qui a fait tous les efforts, malgré son souci de santé, pour participer à la soutenance en vidéo-conférence depuis l'Italie.

Viorica Sofronie-Stokkermans qui m'a fait l'honneur d'examiner cette thèse, en venant d'Allemagne pour participer à la soutenance.

Dominique Mery qui a eu la gentillesse d'accepter d'examiner cette thèse et de présider le jury de la soutenance.

Les équipes Cassis et Prothéo, pour leur accueil, la sympathie de leurs membres et la qualité des échanges scientifiques que j'ai eu avec eux ont été pour moi une source de soutien sur laquelle j'ai pu compter durant ma thèse.

Je tiens également à remercier mes amis au Loria pour les discussions dans lesquelles nous avons partagé l'enthousiasme et parfois aussi l'incertitude de la vérité de notre travail.

Finalement, je remercie Huong, Linh et Minh de me rendre heureux.

à ma famille

Résumé

Beaucoup d'applications des méthodes formelles reposent sur la génération de formules en logique du premier ordre et la preuve de leur satisfiabilité par rapport à une théorie en arrière-plan, qui est souvent obtenu par mélange de plusieurs théories. Dans la littérature, cette forme de satisfiabilité est appelée Satisfiabilité Modulo Théories (SMT). Dans cette thèse, on s'intéresse à la conception de procédures de décision pour les problèmes SMT, en intégrant des techniques de saturation basées sur la réécriture pour des théories finiment axiomatisées et des techniques de combinaison pour des unions de théories.

La première contribution de cette thèse est une reconstruction raisonnée, dans un cadre uniforme, des méthodes de combinaison proposées par Nelson-Oppen, Shostak et d'autres. Ceci est le point de départ pour de nouvelles investigations. Nous introduisons ensuite le concept de canoniseur étendu et dérivons un résultat de modularité pour une nouvelle classe de théories, ce qui contraste avec l'absence de modularité pour la classe de théories considérée par Shostak. La deuxième contribution concerne le problème de la combinaison de procédures basées sur la réécriture en utilisant la méthode de Nelson-Oppen. Nous utilisons la méta-saturation pour développer des techniques de preuve automatique permettant de tester les conditions pour la combinabilité de telles procédures. Lorsque la méta-saturation termine pour une théorie, le résultat obtenu permet de raisonner sur la combinabilité pour cette théorie d'une procédure de satisfiabilité basée sur la réécriture. La troisième contribution de cette thèse est liée à l'intégration des procédures de décision dans les solveurs SMT. Nous considérons le problème de rajouter aux procédures de décision la capacité de construire des justifications en cas d'insatisfiabilité, sans dégradation des performances, en nous focalisant sur la construction modulaire de telles justifications pour une théorie combinée. Pour ce faire, nous étendons la méthode de combinaison de Nelson-Oppen de manière à construire de façon modulaire des justisfications d'insatisfiabilité pour des unions de théories. Nous étudions également comment les justifications obtenues peuvent être reliées à une forme appropriée de minimalité.

Mots clés : Procédures de Décisions, Combinaison, Saturation, Justification.

Table des matières

Table des figures

Chapitre 1

Introduction

L'importance croissante, dans la vie quotidienne, des réseaux et systèmes informatiques pose, de façon toujours plus cruciale, le problème de leur sécurité. Les méthodes formelles basées sur une approche mathématique rigoureuse permettent de prouver des propriétés de logiciels, mais le passage à l'échelle de ces méthodes se heurte encore à des verrous technologiques limitant le champ d'application des outils de preuve. En effet, les systèmes bien connus comme Coq, HOL, Isabelle sont à l'origine des assistants de preuves pour lesquels les preuves sont très peu automatisées et doivent être décrites de façon rigoureuse par les utilisateurs. Ainsi, même dans le cas où on vérifie une propriété d'un système de taille moyenne, la preuve est déjà longue et fastidieuse. De plus, l'utilisation des assistants de preuves exige souvent une expertise en logique et mathématique. Afin de construire des système de preuves nécessitant moins d'interactions et donc plus faciles à utiliser, il est souhaitable de concevoir des outils permettant d'automatiser les preuves. Ces outils sont appelés dans la littérature des *procédures de décision*. L'objectif principal de cette thèse concerne l'étude et l'intégration de procédures de décision dans des systèmes de preuve.

1.1 Problématique

En principe, vérifier la correction d'un programme revient à vérifier la validité des obligations de preuve générées à partir de sa spécification. La spécification d'un programme est souvent décrite dans une certaine logique et les obligations de preuves correspondantes appartiennent à une classe de formules dans cette logique. En pratique, pour vérifier la validité d'une formule on peut utiliser l'approche par réfutation, c'est-à-dire on vérifie si la négation de cette formule est insatisfiable. Si nous disposons de procédures de décision pour traiter le problème de satisfiabilité d'une classe de formules dans la logique considérée, alors nous avons une méthode de vérification automatique de la correction du programme correspondant.

Procédures de décision et leur combinaison

Nous considérons dans le contexte de cette thèse qu'une procédure de décision est un algorithme qui, pour une classe de formules donnée, nous indique leur satisfiabilité. Les procédures de décision se trouvent au cœur de tous les outils modernes d'analyse et de vérification de programmes car leur utilisation d'une part accroît l'efficacité du

système de vérification, et d'autre part libère l'utilisateur de l'interaction, souvent fastidieuse, avec le système. Des procédures de décision sont connues pour plusieurs domaines intéressants en pratique tels que des fragments décidables de l'arithmétique sur les entiers, les réels ainsi que les structures de données apparaissant fréquemment dans les programmes (listes, tableaux ...).

Cependant, la plupart des problèmes de vérification font intervenir des mélanges de différentes théories. Considérons l'exemple d'un problème de vérification dans lequel la vérification de la satisfiabilité fait intervenir la théorie $\mathcal{A}$ des tableaux et le fragment $\mathcal{LA}$ de l'arithmétique linéaire dont les axiomatisations sont les suivantes [1].

$$\mathcal{A} = \begin{cases} \mathsf{select}(\mathsf{store}(A, I, E), I) = E \\ I \neq J \Rightarrow \mathsf{select}(\mathsf{store}(A, I, E), J) = \mathsf{select}(A, J) \end{cases}$$

$$\mathcal{LA} = \begin{cases} X + 0 = X \\ X + (-X) = 0 \\ (X + Y) + Z = X + (Y + Z) \\ X + Y = Y + X \\ X \leq X \\ X \leq Y \vee Y \leq X \\ X \leq Y \wedge Y \leq X \Rightarrow X = Y \\ X \leq Y \wedge Y \leq Z \Rightarrow X \leq Z \\ X \leq Y \Rightarrow X + Z \leq Y + Z \\ 0 \neq 1 \\ 0 \leq 1 \end{cases}$$

Le problème peut être de déterminer la *satisfiabilité* [2] de la formule [3]

$$\mathsf{select}(\mathsf{store}(a, i, \mathsf{select}(a, j)), i) \neq \mathsf{select}(a, i) \ \wedge \ i + j \leq 2j \ \wedge \ j + 4i \leq 5i$$

Pour résoudre ce genre de problème, une première approche naïve est de considérer indépendamment le problème dans chaque théorie composante. Si la procédure de décision de chaque théorie élémentaire retourne un résultat positif, on retourne le résultat final comme positif, et négatif sinon. Essayons d'appliquer cette méthode naïve :

1. la procédure de satisfiabilité de $\mathcal{LA}$ retourne *satisfiable* avec la formule

$$i + j \leq 2j \ \wedge \ j + 4i \leq 5i$$

2. la procédure de satisfiabilité de $\mathcal{A}$ retourne *satisfiable* avec la formule

$$\mathsf{select}(\mathsf{store}(a, i, \mathsf{select}(a, j)), i) \neq \mathsf{select}(a, i)$$

3. la procédure de satisfiabilité de l'union devrait donc retourner *satisfiable*...

Mais en réalité, on peut déduire $i = j$ à partir de la conjonction $i+j \leq 2j \wedge j+4i \leq 5i$. Le premier axiome de $\mathcal{A}$, $i = j$ et $\mathsf{select}(\mathsf{store}(a, i, \mathsf{select}(a, j)), i) \neq \mathsf{select}(a, i)$ nous permettent de déduire $\mathsf{select}(a, i) \neq \mathsf{select}(a, i)$, qui est *insatisfiable*. Le problème vient

1. Les variables apparaissant dans les axiomes sont universellement quantifiées.
2. $\mathcal{A}$ et $\mathcal{LA}$ ainsi que leur combinaison sont décidables (voir par exemple [Opp80]).
3. Les variables apparaissant dans cette formule sont existentiellement quantifiées.

de l'interaction entre les théories composantes, qui est, dans l'exemple, due au partage de variables et du symbole d'égalité.

L'exemple précédent montre la subtilité du problème de combinaison de procédures de décision. Ce problème s'énonce formellement comme suit : étant données n théories T_1, T_2, ..., T_n avec leurs n procédures de décision P_1, P_2, ..., P_n respectivement, peut-on construire une procédure de décision dans la théorie $T_1 \cup T_2 \cup ... \cup T_n$?

Nelson-Oppen A la fin des années 70, Nelson et Oppen [NO79] ont proposé une approche très générale pour construire une procédure de satisfiabilité des formules *sans quantificateurs* dans l'union des théories à signatures *disjointes* $T_1 \cup T_2 \cup ... \cup T_n$. La méthode est basée sur trois étapes : *Purification*, *Test de satisfiabilité* et *Propagation d'égalités*.

1. La *purification* consiste à renommer chaque sous-terme étranger par une variable. Intuitivement, cette étape décompose la formule mixte [4] φ en une conjonction $\varphi_1 \wedge \varphi_2 \wedge ... \wedge \varphi_n$ de n formules *pures* [5] φ_i sur lesquelles on peut appliquer chaque procédure de décision P_i.
2. Le *test de satisfiabilité* vérifie la satisfiabilité dans chaque théorie composante.
3. La *propagation d'égalités* consiste à propager toutes les égalités impliquées dans une théorie composante aux autres théories et revenir à l'étape *Test de satisfiabilité* si nécessaire.

Construite dans un souci de réutilisation, la méthode Nelson-Oppen procède de manière modulaire sur chacune des théories composantes et ensuite les fait communiquer par la *Propagation d'égalités*. Avec l'étape de *Propagation d'égalités* l'interaction entre les théories composantes est prise en compte, le problème de cohérence globale que l'on a mentionné ci-dessus est résolu. La méthode Nelson-Oppen a été implantée dans différents outils de vérification comme CVC [SBD02], ESC [DLNS98].

Shostak En 1984, Shostak a proposé une méthode plus restreinte basée sur la *Clôture de Congruence* [Sho84] pour la combinaison de la théorie de l'égalité et d'autres théories admettant un *canoniseur* et un *solveur*. Bien que la méthode de Shostak soit plus restreinte que celle de Nelson et Oppen, elle a suscité un vif intérêt dans la communauté et a été implantée dans les systèmes de vérification ICS [FORS01], PVS [ORR+96], SVC [BDL96], STeP [BBC+96]. Il y a quelques explications à ce fait. La première raison est l'efficacité. Les études ont montré que la méthode de Shostak est beaucoup plus performante que celle de Nelson et Oppen. La deuxième raison est que malgré les restrictions requises, un nombre significatif de théories intéressantes d'un point de vue pratique font partie de la classe des théories de Shostak.

Malgré la popularité des deux méthodes, les papiers originaux ont manqué de rigueur en terme de correction et de lisibilité. Une liste impressionnante d'articles [CLS96, RS01, BDS02, Kap02, Gan02, CK03b, RS02, CK03a, MZ03] ont été publiés pour corriger les erreurs des deux méthodes, en particulier celle de Shostak. Certaines hypothèses sur les théories composantes ont été introduites pour pallier le problème de correction notamment la *convexité* (pour Shostak) [BDS02, Opp80, Gan02] et la *stable infinité* (pour Nelson-Oppen) [Opp80]. Une direction de recherche intéressante

4. Une formule est mixte si les symboles dans cette formule appartiennent à plusieurs théories.
5. Une formule est pure si tous les symboles dans cette formule appartiennent à une seule théorie.

consiste à définir un cadre uniforme dans lequel la combinaison de Shostak peut s'exprimer comme un raffinement de celle de Nelson-Oppen.

Procédures de décision dérivées par saturation et leur combinaison

Les procédures de satisfiabilité pour les théories modélisant les structures de données telles que les listes ou les tableaux se trouvent au coeur de beaucoup de systèmes de vérification, comme PVS [ORR+96], Simplify [DNS03], ICS [FORS01], CVC [SBD02], CVC Lite [BB04]. La conception, la preuve de correction et l'implantation de telles procédures présentent des défis qui sont loin d'être triviaux. Un des problèmes majeurs est de prouver la correction de ces procédures. De plus, l'implantation de chaque procédure se fait de façon *ad hoc*, avec peu de réutilisation de composants existants. Pour franchir ces obstacles, les auteurs de [ARR03] ont proposé une méthodologie uniforme basée sur la saturation pour construire des procédures de satisfiabilité et prouver leur correction. La preuve de correction se réduit à la preuve de la terminaison d'une application équitable et exhaustive des règles du Calcul de Superposition [NR01]. De plus, de telles procédures peuvent être développées en utilisant les systèmes de preuve implantant le Calcul de Superposition [Sch02, Wei97, RV02]. Les efforts d'ingénierie et de validation de logiciels peuvent être ainsi réutilisés.

Malheureusement, la plupart des problèmes de vérification font intervenir des domaines de calcul pour lesquels la méthodologie basée sur la saturation ne s'applique pas, par exemple l'arithmétique linéaire sur les réels ou plus généralement des théories non finiment axiomatisables. Il y a donc un besoin indiscutable de combiner des procédures de satisfiabilité dérivées par saturation avec d'autres procédures de satisfiabilité construites par d'autres techniques. Une solution consiste à utiliser la méthode de combinaison de Nelson-Oppen [NO79] pour combiner de façon modulaire des procédures composantes. Pour avoir une intégration efficace des procédures de satisfiabilité par saturation dans la combinaison à la Nelson-Oppen, il est important d'étudier les questions suivantes :

- comment assurer que les théories composantes sont stablement infinies ?
- comment produire directement des formules, qui sont propagées d'une procédure à l'autre, une fois que la saturation est terminée ?
- comment traiter efficacement de nouvelles formules qui proviennent d'autres procédures ?

Solveurs de satisfiabilité modulo théories (SMT)

En pratique, pour vérifier la satisfiabilité des formules sans quantificateurs on peut se ramener à la vérification de la satisfiabilité de conjonctions des littéraux *clos* en transformant les obligations de preuve en forme normale disjonctive (DNF). Malgré la correction de la méthode au niveau théorique, cette technique se heurte à certains problèmes en pratique. Premièrement, transformer une formule en DNF est trop coûteux en terme d'espace. Et deuxièmement, si la formule est insatisfiable alors le nombre d'appels de la procédure de décision est égal au nombre de disjoints de la DNF. Ceci augmente rapidement le coût de la vérification de l'insatisfiabilité d'une formule. Pour traiter efficacement des formules ayant une structure booléenne arbitraire, des solveurs SMT (e.g. haRVey [DR03], MathSat [BBC+06], DPLL(T) [GHN+04], ICS [FORS01], CVC-Lite [BB04], et Zapato [BCLZ04]) sont développés en utilisant soit les BDDs

(Binary Decision Diagrams) qui permettent de représenter de manière compacte les formules Booléennes, soit des SAT solveurs qui permettent de trouver un modèle propositionnel de la formule. Les solveurs SMT fonctionnent en combinant une procédure de satisfiabilité pour le calcul propositionnel—un solveur propositionnel—et une procédure de satisfiabilité modulo une théorie T pour des conjonctions de littéraux. L'interaction entre ces deux procédures s'effectue grossièrement de la façon suivante :

1. Faire abstraction des atomes d'une formule ϕ, en les substituant par des variables propositionnelles de manière à obtenir une formule propositionnelle ϕ^p.
2. Appeler le solveur propositionnel sur ϕ^p afin de trouver une affectation α^p de variables propositionnelles.
3. Cette affectation α^p correspond dans la théorie T à une conjonction α de littéraux. Si α est satisfiable dans T, alors on peut conclure que ϕ est satisfiable. Sinon, on ajoute $\neg\alpha^p$ à ϕ^p pour éliminer ce candidat de solution qui n'est pas une solution. Et on retourne à l'étape 2.

L'insatisfiabilité de ϕ est déclarée lorsque le solveur propositionnel ne trouve plus de solution à ϕ^p.

Le principal inconvénient de cette méthode est son efficacité. Le nombre d'appels à la procédure croît exponentiellement avec le nombre de littéraux de la formule ϕ. Ceci rend les performances inacceptables dès lors que la formule est grande. Il est donc primordial de s'intéresser à optimiser cette méthode. Une direction prometteuse consiste à utiliser des procédures de satisfiabilité ayant la capacité de justifier l'insatisfiabilité au moyen d'un "petit" ensemble de littéraux, ce qui permet d'éliminer en un coup un "grand" nombre d'affectations propositionnelles. On réduit ainsi de façon significative le nombre d'itérations nécessaires à la détection de l'insatisfiabilité de ϕ.

1.2 Contribution

Un cadre uniforme pour Nelson-Oppen et Shostak

Nous avons apporté une contribution [RRT04, RRT03] au débat qui consiste à se positionner par rapport aux deux grand schémas de combinaison disjointe (i.e. quand les théories combinées ne partagent pas de symboles) pour la satisfiabilité : Nelson-Oppen et Shostak. Depuis quelques années, certains papiers ont clarifié les subtilités de la combinaison des théories de Shostak en étudiant leurs relations avec les théories de Nelson-Oppen. Malheureusement, ces papiers manquent d'uniformité dans la présentation des résultats ce qui peut troubler les non-experts. Par exemple, certains papiers utilisent du pseudo-code pour décrire les algorithmes de combinaison alors que d'autres adoptent une présentation plus abstraite. Les deux approches ont leurs avantages (et inconvénients) respectifs : le pseudo-code permet de démarrer plus rapidement une implantation, alors qu'un système d'inférence facilite les preuves de correction.

La première contribution de ce travail est de fournir une synthèse des approches de Nelson-Oppen et Shostak pour la combinaison disjointe en utilisant une approche à base de règles dans laquelle beaucoup de résultats récents sont reformulés et prouvés corrects de façon uniforme, rigoureuse et simple. Notre synthèse des schémas de combinaison sert plusieurs objectifs. Premièrement, même si les résultats ne sont pas

nouveaux, nous pensons que les présenter dans un cadre uniforme peut fournir une référence de valeur pour les personnes intéressées par les problèmes de combinaison et plus particulièrement pour les non-experts du domaine. Deuxièmement, cela peut servir comme point de départ pour des recherches supplémentaires. Comme exemple, l'un des problèmes importants en combinant des théories de Shostak est le manque de modularité des solveurs : il n'existe pas de méthode générale pour produire un solveur pour l'union de théories de Shostak à partir des solveurs connus pour les théories composantes [CK03b]. Ce manque de modularité, plus l'observation que la théorie de l'égalité n'est pas une théorie de Shostak, peut nous laisser penser qu'une combinaison *ad hoc* d'une théorie de Shostak et une théorie de Nelson-Oppen constitue un compromis raisonnable entre efficacité et généralité : solveurs et canoniseurs pour les théories de Shostak dérivent efficacement de nouvelles égalités et coopèrent à la Nelson-Oppen. Cette solution peut être facilement spécifiée dans le cadre proposé. Toutefois elle laisse ouverte la question d'un concept permettant d'avoir un résultat modulaire tout en gardant l'efficacité apportée par les solveurs et les canoniseurs. En réponse à cette question, nous proposons le concept de *canoniseur étendu* qui constitue la deuxième contribution de ce travail. Ce nouveau concept présente plusieurs avantages : (i) la construction de canoniseurs étendus est modulaire ; (ii) un canoniseur étendu décide la satisfiabilité pour une large classe de théories, à savoir les théories convexes ; (iii) les canoniseurs étendus peuvent être construits en utilisant des techniques de réécriture dans certains cas intéressants (cf. Chapitre 6).

Tous les résultats ci-dessus supposent que les théories composantes satisfont les hypothèses de convexité et de stable infinité. La convexité nous permet d'avoir une méthode déterministe de Nelson-Oppen tandis que la stable infinité est une condition suffisante pour la correction de la méthode de Nelson-Oppen. Nous avons ensuite travaillé dans le but d'étendre ces résultats au cas où les théories sont convexes mais pas nécessairement stablement infinies. Nous montrons que les procédures de satisfiabilité des théories convexes finiment axiomatisées peuvent être combinées même si ces théories ne sont pas stablement infinies. Dans le même esprit, nous montrons également qu'on peut avoir une construction modulaire de canoniseur étendu pour l'union des théories convexes (pas nécessairement stablement infinies ou finiment axiomatisées) admettant un canoniseur étendu (cf. Chapitre 7). Ce résultat généralise le résultat de modularité de canoniseur étendu pour les théories convexes et stablement infinies.

Combinaison des procédures de décision dérivées par saturation

Lorsque l'on combine des procédures de décision dérivées par saturation en utilisant la méthode de Nelson-Oppen, les questions à se poser sont :

- comment produire directement des formules qui sont propagées d'une procédure à l'autre ?
- comment traiter efficacement de nouvelles formules qui proviennent d'autres procédures ?

Le premier problème peut être décidé en utilisant la procédure de T-satisfiabilité, i.e. pour vérifier qu'une formule ϕ est une conséquence d'un ensemble S de formules dans une théorie T, il suffit de vérifier que $\neg\phi$ est insatisfiable dans T. Mais cette solution n'est évidemment pas efficace, comme observé dans [DNS03]. Afin de franchir cet obstacle, nous montrons qu'une application exhaustive des règles du Calcul de Superposition dérive suffisamment de formules partagées pour la théorie de l'égalité et la théorie

des listes [KRRT05, KRRT06b]. Pour la théorie des tableaux nous montrons qu'une application exhaustive des règles d'une variante du Calcul de Superposition—obtenu en rajoutant au calcul une règle de "fragmentation" ("splitting")—dérive suffisamment de formules partagées (cf. Chapitre 4).

Pour résoudre le deuxième problème, la solution naïve consistant à refaire la saturation sur le nouvel ensemble présente quelques inconvénients. D'un côté il peut être inutile de saturer tout le nouvel ensemble car certaines clauses ne seront pas utilisées, et d'un autre côté il est difficile de faire un retour en arrière lorsque les nouvelles formules reçues causent l'insatisfiabilité. Nous proposons une architecture dans laquelle le Calcul de Superposition fonctionne comme un générateur de lemmes qui sont ensuite utilisés par la Clôture de Congruence. Les procédures de satisfiabilité construites de cette façon peuvent être combinées efficacement car celles-ci sont capables de traiter efficacement les égalités provenant d'autres procédures en utilisant la Clôture de Congruence. Ce qui signifie qu'il n'y a pas besoin d'appeler à nouveau la saturation. Nous montrons que notre architecture est correcte pour des théories modélisant des structures de données comme la théorie des listes et la théorie des tableaux (cf. Chapitre 4).

Décider automatiquement la combinabilité par méta-saturation

Nous avons ensuite étudié la généralisation des résultats du travail précédent [KRRT05, KRRT06b]. Dans cette optique, nous avons défini des conditions suffisantes permettant de vérifier

- si les théories admettent une procédure de satisfiabilité basée sur le calcul de Superposition ;
- si ces procédures de satisfiabilité ont la capacité de produire efficacement des formules partagées ;
- si les théories composantes sont stablement infinies pour que les procédures de satisfiabilité correspondantes puissent être combinées avec d'autre procédures de satisfiabilité en utilisant la méthode de Nelson-Oppen.

De plus, nous avons développé une méthode basée sur la méta-saturation de Lynch et Morawska [LM02] pour vérifier automatiquement ces conditions pour les théories axiomatisées par un ensemble fini de clauses. Nous avons pu identifier, en utilisant notre méthode automatique, une classe de théories satisfaisant ces conditions, qui inclut bien évidemment la théorie de l'égalité, la théorie des listes et leurs combinaisons ([KRRT06a]). Nous étendons par la suite cette méthode à la saturation avec fragmentation (cf. Chapitre 5).

Procédures de décision avec témoin de preuve pour des solveurs SMT

Une technique importante pour obtenir un outil SMT efficace consiste à engendrer des (ensembles représentatifs de) *conflits* comme effet de bord de la procédure de satisfiabilité car leur utilisation permet d'élaguer énormément l'espace de recherche géré par le solveur Booléen. Des travaux ont été réalisés pour étendre, par la production de conflits, des procédures de satisfiabilité pour certaines théories (dont la théorie de l'égalité) [Fon04, dMRS04, NO05, ST05]. Par contre, il n'existe pas (à notre connaissance) de travaux publiés sur la construction modulaire de conflits pour des mélanges de théories, même si des systèmes SMT doivent d'une façon ou d'une autre implanter

cette fonctionnalité. Ainsi, un développeur souhaitant intégrer cette fonctionnalité dans son propre système SMT doit comprendre le code d'autres systèmes (qui n'est pas toujours disponible) afin d'en extraire les idées essentielles et les mettre en œuvre dans son architecture.

Dans [RRT06] et Chapitre 9, nous présentons comment étendre la méthode de combinaison de Nelson-Oppen de manière à construire une procédure de satisfiabilité produisant des conflits pour le mélange de théories (disjointes), lorsque sont connues des procédures appropriées pour les théories composantes. Dans ce but, nous introduisons le concept de *graphe d'explication*, qui code de façon compacte les égalités élémentaires déduites. Nous montrons l'intérêt de ce concept pour construire une procédure de satisfiabilité pour la théorie de l'égalité augmentée par la capacité à produire des conflits. Nous montrons ensuite comment combiner des procédures de satisfiabilité, appelées *moteurs d'explication*, qui ont la capacité de construire des graphes d'explication. Ceci permet de construire une procédure de satisfiabilité avec la même capacité pour le mélange des théories. Pour finir, nous introduisons le concept de *quasi-conflit*, qui nous permet de caractériser précisément une forme de minimalité satisfaite par les explications construites par notre méthode.

Plan de lecture

Après cette introduction générale, nous définissons, dans le Chapitre 2, des notions de base utilisées tout au long du mémoire. Les lecteurs qui sont familiers avec la logique, la théorie des modèles et la réécriture du premier ordre, peuvent commencer directement la lecture au Chapitre 3.

Les cinq Chapitres 4, 5, 6, 7 et 8 présentent l'ensemble des contributions de la thèse.

Le problème de satisfiabilité dans l'union des théories est traité dans les Chapitres 3, 6 et 7. Le Chapitre 3 a pour but de donner aux non-experts un aperçu global de l'ensemble des résultats principaux du problème de satisfiabilité dans l'union des théories. Nous présentons les résultats concernant la combinaison de théories ayant des signatures disjointes ainsi que la combinaison de théories dont les signatures partagent des symboles. Le Chapitre 6 donne un cadre uniforme pour traiter la combinaison de théories convexes et stablement infinies suivant la méthode de Nelson-Oppen ou celle de Shostak. Les résultats décrits dans ce chapitre servent comme point de départ pour une étude plus générale de combinaison de théories convexes et de théories universelles décrite dans le Chapitre 7.

La déduction automatique par paramodulation fait l'objet des Chapitres 4 et 5. Le Chapitre 4 présente les résultats obtenus pour la combinaison des procédures de décision dérivées par saturation. Dans le Chapitre 5, nous présentons une généralisation des résultats abordés dans le Chapitre 4 et également une méthode automatique pour raisonner sur la combinabilité des procédures de décision dérivées par saturation.

Le Chapitre 8 concerne l'intégration des procédures de satisfiabilité avec des solveurs propositionnels. Nous donnons quelques exemples de procédures de satisfiabilité capables de produire des preuves ainsi qu'une méthode pour combiner des procédures de satisfiabilité ayant cette capacité.

Finalement, le Chapitre 9 conclut en donnant des perspectives à cette thèse.

Chapitre 2

Notions préliminaires

Dans ce chapitre, nous introduisons les éléments de base utilisés tout au long de ce manuscrit. Nous commençons par introduire des éléments de syntaxe de la logique du premier ordre comme signature, terme, atome et formule. Nous introduisons certains concepts liés à la réécriture (de termes). Ensuite, nous abordons les aspects sémantiques, en introduisant les notions classiques d'interprétation, satisfiabilité, validité et modèle (d'une théorie). Nous finissons ce chapitre en rappelant des procédures de satisfiabilité pour certaines théories nous intéressant plus particulièrement.

2.1 Syntaxe de premier ordre

Cette section introduit des objets syntaxiques standards utilisés en logique de premier ordre comme les termes, les formules, les substitutions ...

Définition 2.1 (Signature)**.** *Une signature Σ se compose de :*
- *un ensemble Σ^P de symboles de relation avec arité $n \geq 0$, et*
- *un ensemble Σ^F de symboles de fonction avec arité $n \geq 0$.*

Un symbole de constante est un symbole de fonction d'arité nulle. On note Σ^C, l'ensemble de symboles de constante d'une signature. La cardinalité $|\Sigma|$, d'une signature est le nombre de symboles de Σ.

Définition 2.2 (Terme)**.** *Soient Σ une signature et X en ensemble de variables. L'ensemble $T(\Sigma, X)$ de Σ-termes (sur les variables X) est le plus petit ensemble tel que :*
- *chaque variable de X est dans $T(\Sigma, X)$, et*
- *chaque constante de Σ est dans $T(\Sigma, X)$, et*
- *si f est un symbole de fonction d'arité n dans Σ et $t_1, \ldots, t_n$ sont dans $T(\Sigma, X)$ alors $f(t_1, \ldots, t_n)$ est dans $T(\Sigma, X)$.*

Un Σ-terme est clos s'il ne contient pas de variable.

Si t est un terme alors $Var(t)$ désigne l'ensemble des variables apparaissant dans t. Cette notation s'étend naturellement aux ensembles de termes. Un sous-terme d'un terme t est un terme qui apparaît dans t. Un sous-terme strict de t est un sous-terme de t qui est différent de t. Un terme peut être vu comme un arbre fini étiqueté, dont les feuilles sont étiquetées par des variables ou constantes et dont les noeuds internes sont étiquetés par des symboles d'arité positive.

Définition 2.3 (Hauteur de terme)**.** *La hauteur d'un terme t est récursivement défini par :*

- *$hauteur(t) = 0$, si t est une constante ou une variable,*
- *$hauteur(f(t_1, \ldots, t_n)) = 1 + max\{depth(t_i) \mid 1 \leq i \leq n\}$.*

Un terme t est plat si sa hauteur est 0 ou 1.

Définition 2.4 (Position et contexte)**.** *Une position (ou une occurrence) dans un terme t est représentée par une séquence d'entiers positifs décrivant le chemin de la racine de t au sous-terme à cette position, noté $t|_p$. Un terme u a une occurrence dans t si $u = t|_p$ pour une certaine position p.*

Un contexte est un terme avec une position distinguée.

La notation $t[s]_p$ signifie que le terme t contient s comme sous-terme à la position p et $t[s]$ signifie que s est un sous-terme de t.

Définition 2.5 (Substitution)**.** *Une substitution est une application sur $T(\Sigma, X)$ qui est déterminée uniquement par l'image de X. Elle est par conséquent écrite sous la forme $\{x_1 \to t_1, \ldots, x_n \to t_n\}$ lorsqu'il existe un nombre fini de variables dont les images ne sont pas elles-mêmes. L'application d'une substitution $\sigma = \{x_1 \to t_1, \ldots, x_n \to t_n\}$ sur un terme t, noté $\sigma(t)$, est définie récursivement comme suit :*

- *si t est une variable et $t \neq x_i$ pour tout $i = 1, \ldots, n$, alors $\sigma(t) = t$,*
- *si $t = x_i$ pour un certain $i = 1, \ldots, n$, alors $\sigma(t) = t_i$,*
- *si $t = f(u_1, \ldots, u_m)$, où $u_1, \ldots, u_m \in T(\Sigma, X)$ et $f \in \Sigma^F$, alors $\sigma(t) = f(\sigma(u_1), \ldots, \sigma(u_m))$.*

Si σ est une substitution, on appelle les ensembles

$$\mathcal{D}om(\sigma) = \{x \mid x \in X \text{ et } \sigma(x) \neq x\} \quad \text{et} \quad \mathcal{R}an(\sigma) = \{\sigma(x) \mid x \in \mathcal{D}om(\sigma)\}$$

respectivement le domaine et le codomaine de σ.

La substitution vide (substitution identité) est notée ϵ. Si σ et ρ sont des substitutions, la composition $\sigma \circ \rho$ est définie par $\sigma \circ \rho(x) = \rho(\sigma(x))$ (ou $x\sigma\rho$). Une substitution σ est idempotente si $\sigma \circ \sigma = \sigma$. Si $\sigma = \{x_1 \to t_1, \ldots, x_n \to t_n\}$ est une substitution et t est un terme, nous utilisons de façon interchangeable la notation préfixée $\sigma(t)$ ou la notation postfixée $t\sigma$ pour désigner l'application de σ à t.

Définition 2.6 (Remplacement)**.** *Le replacement d'un terme t par un terme t' dans un terme u, noté $u[t \to t']$, est défini récursivement par :*

- *si $u = t$ alors $u[t \to t'] = t'$,*
- *si $u \neq t$ et $u = f(u_1, \ldots, u_n)$ alors $u[t \to t'] = f(u_1[t \to t'], \ldots, u_n[t \to t'])$.*

Les symboles suivants sont utilisés dans le langage de la logique premier ordre mais ne sont inclus dans aucune signature :

- les connecteurs Booléens : $\top, \bot, \wedge, \vee, \neg$;
- les quantificateurs : $\exists, \forall$;
- l'égalité : $=$.

Définition 2.7 (Atome)**.** *Soit Σ une signature. L'ensemble des Σ-atomes est le plus petit ensemble tel que :*

- *$\top$ est un Σ-atome, et*
- *une Σ-égalité, i.e. une expression de la forme $s = t$, où s, t sont des Σ-termes, est un Σ-atome, et*

– *une relation, i.e. une expression de la forme* $P(t_1, \ldots, t_n)$*, où* P *est un symbole de relation d'arité* n *dans* Σ^P *et* $t_1, \ldots, t_n$ *sont des* Σ*-termes, est un* Σ*-atome.*

Un littéral est un Σ*-atome ou la négation d'une* Σ*-atome. La négation de* $\top$ *est* $\bot$*. Une diségalité est la négation d'une égalité, i.e. une expression de la forme* $\neg s = t$*, souvent notée* $s \neq t$*.*

Définition 2.8 (Hauteur de littéral)**.** *La hauteur d'une égalité ou d'une diségalité est définie par :*

$$hauteur(l \bowtie r) = hauteur(l) + hauteur(r)\,,$$

où $\bowtie \in \{=, \neq\}$*. Une égalité est plate si sa hauteur est* 0 *ou* 1*. Une diségalité est plate si sa hauteur est* 0*.*

Définition 2.9 (Formule)**.** *Soit* Σ *une signature. L'ensemble des* Σ*-formules est le plus petit ensemble tel que :*

– *tout* Σ*-atome est une* Σ*-formule, et*
– *si* φ *et* ϕ *sont des* Σ*-formules et* v *est une variable alors* $\neg\varphi$*,* $\varphi \wedge \phi$*,* $\varphi \vee \phi$*,* $\exists v.\varphi$ *et* $\forall v.\varphi$ *sont des* Σ*-formules.*

Une Σ*-formule est close si elle ne contient pas de variable.*

Définition 2.10 (Portée de quantificateurs)**.** *Dans une formule* $Qv.\varphi$*, où* Q *est* $\exists$ *ou* $\forall$*, on dit que* φ *est la portée du quantificateur* Q*.*

Définition 2.11 (Variable libre)**.** *Une variable est libre dans une formule* φ *si au moins une des ses occurrences n'apparaît pas dans la portée d'aucun quantificateur de* φ*.*

Si φ est une formule alors $Var(\varphi)$ désigne l'ensemble de variables libres dans φ. Cette notation s'étend bien évidemment aux ensembles de formules.

Définition 2.12 (Sentence)**.** *Une formule est une sentence si elle ne contient pas de variable libre.*

Définition 2.13 (Forme prénexe)**.** *Une formule est en forme prénexe si elle a la forme*

$$Q_1 v_1 \ldots . Q_n v_n . \varphi$$

où $Q_i \in \{\exists, \forall\}$ *et* φ *est une formule sans quantificateur. Dans ce cas, on dit que* $Q_1 v_1 \ldots . Q_n v_n$ *est le préfixe de quantificateurs et* φ *est la matrice de la forme prénexe.*

Définition 2.14 (Formule universelle et formule existentielle)**.** *Une formule est universelle (respectivement existentielle) si elle est en forme prénexe dont le préfixe de quantificateurs ne contient pas de quantificateurs existentiels (respectivement universels).*

Définition 2.15 (Forme de Skolem)**.** *Une formule est en forme de Skolem si elle est en forme prénexe sans quantificateur existentiel.*

Définition 2.16 (Forme clausale)**.** *Une formule est*

– *une clause si elle est de la forme*

$$l_1 \vee \ldots \vee l_n$$

où l_i *est un littéral dont toutes les variables sont implicitement et universellement quantifiées, pour* $0 \leq i \leq n$ *;*

- *clause de Horn si c'est une clause dont au plus un de ses disjoints est un atome.*
- *clause de Horn stricte si c'est une clause de Horn dont au moins un des disjoints est un atome.*

Une clause peut être aussi vue comme un multi-ensemble de littéraux dont toutes les variables sont universellement quantifiées.

Définition 2.17 (Formes normales)**.** *Une formule sans quantificateurs est*
- *en forme normale disjonctive (ou en DNF) si elle est de la forme*

$$(l_1 \wedge \ldots \wedge l_{k_1}) \vee \ldots \vee (l_n \wedge \ldots \wedge l_{k_1}),$$

- *en forme normale conjonctive (ou en CNF) si elle est de la forme*

$$(l_1 \vee \ldots \vee l_{k_1}) \wedge \ldots \wedge (l_n \vee \ldots \vee l_{k_1}),$$

où l_i est un littéral.

Il est bien connu que toute formule sans quantificateur est équivalente à une formule en DNF (resp. CNF).

Un symbole avec un tilde en tête désigne une séquence, par exemple $\tilde{x}$ désigne la séquence $x_1, \ldots, x_n$ avec $n \geq 0$. Par simplicité, nous considérons également $\tilde{x}$ comme un produit cartésien ou un ensemble. La notation $\varphi(\tilde{x})$, ou $\varphi(x_1, \ldots, x_n)$, signifie que les variables libres de φ sont exactement $x_1, \ldots, x_n$. Pour toute formule $\varphi(x_1, \ldots, x_n)$, $\tilde{\exists}\varphi$ est une abréviation de $\exists x_1 \ldots \exists x_1.\varphi$ (la clôture existentielle de φ) et $\tilde{\forall}\varphi$ est une abréviation de $\forall x_1 \ldots \forall x_1.\varphi$ (la clôture universelle de φ). Nous utilisons de façon interchangeable, par abus de langage, un ensemble de formules et une conjonction de formules dans cet ensemble.

2.2 Réécriture

Nous allons étudier une relation particulière entre les termes que l'on appelle la relation de réécriture ordonnée. Les notions et résultats introduits dans cette section seront utilisés essentiellement dans les Chapitres 4 et 5 dans lesquels la réécriture est la base de notre méthodologie et de nos résultats. Un présentation plus détaillée du sujet se trouve dans [BN98].

Nous introduisons d'abord quelques relations binaires classiques.

Définition 2.18. *Une relation binaire R sur un ensemble T est*
- *réflexive si pour tout $x \in T$, xRx ;*
- *symétrique si pour tout $x, y \in T$, xRy implique yRx ;*
- *antisymétrique si pour tout $x, y \in T$, xRy et yRx implique $x = y$;*
- *transitive si pour tout $x, y, z \in T$, xRy et yRz implique xRz ;*
- *un pré-ordre si elle est réflexive et transitive ;*
- *un ordre si elle est réflexive, transitive et antisymétrique ;*
- *totale si pour tout $x, y \in T$, xRy ou yRx ;*
- *une équivalence si elle est réflexive, transitive et symétrique.*

Étant donnée un pré-ordre $\geq$ sur un ensemble T, la relation d'équivalence induite par $\geq$, notée $\approx_{\geq}$ est définie par

$$\forall x, y \in T, x \approx_{\geq} y \text{ ssi } x \geq y \text{ et } y \geq x$$

L'ordre (strict) $>$ induit par $\geq$ est défini par

$$\forall x, y \in T, x > y \text{ ssi } x \geq y \text{ et } x \not\approx_{\geq} y$$

Définition 2.19 (Ordre bien fondé)**.** *Un ordre $>$ sur un ensemble T est bien fondé (ou noethérien) s'il n'existe pas de suite infinie décroissante d'éléments de T $t_1 > t_2 > \ldots$.*

La construction d'ordres bien fondés peut éventuellement se faire par extension. L'extension lexicographique permet par exemple de comparer des uplets. Soient n ensembles ordonnés $(T_i, >_i)_{i=1,\ldots,n}$. Le produit cartésien ordonné lexicographiquement $(T_1 \times \ldots \times T_n, >^{lex})$ est défini par

$$(s_1, \ldots, s_n) >^{lex} (t_1, \ldots, t_n)$$

s'il existe i, $1 \leq i \leq n$ tel que $s_i >_i t_i$ et $\forall j, 1 \leq j < i, s_j = t_j$. Si $>_i$ est bien fondé pour $i = 1, \ldots, n$, alors $>^{lex}$ est aussi bien fondé.

Pour comparer des objets de taille arbitraire, on introduit l'extension multi-ensemble d'un ordre bien fondé. Un multi-ensemble M sur un ensemble T est une application de T vers les entiers naturels. Soit $>$ un ordre sur T, l'ordre multi-ensemble est défini par

$$L >^{mult} M \text{ si } L \neq M \text{ et } (M(y) > L(y) \Rightarrow \exists x \in T, x > y \text{ et } L(x) > M(x)) .$$

Si $>$ est bien fondé alors $>^{mult}$ est également bien fondé.

Définition 2.20 (Système de réécriture)**.** *Une règle de réécriture est une paire de termes orientée, notée $l \to r$, où l est le membre gauche de la règle et r est son membre droit.*

Un système de réécriture sur les termes est un ensemble de règles de réécriture. La relation de réécriture $\to_R$ associée à un système de réécriture R est définie par : $t \to_R t'$ s'il existe une position p dans t, une règle $l \to r$ dans R et une substitution σ telles que $t|_p = \sigma(l)$ et $t' = t[\sigma(r)]_p$.

Définition 2.21. *Un système de réécriture R est localement confluent (respectivement confluent, terminant) si la relation de réécriture $\to_R$ est localement confluente (respectivement confluente, terminante).*

Nous allons maintenant définir des relations d'ordre qui sont propres à la réécriture.

Définition 2.22. *Un ordre $>$ sur les termes est*

- *un ordre de réécriture s'il est stable par contexte et substitution, i.e. pour tous les termes t, t', u et toute substitution σ*

$$t > t' \Rightarrow u[\sigma(t)] > u[\sigma(t')] ,$$

- *un ordre de réduction si c'est un ordre de réécriture bien fondé,*
- *un ordre de simplification si c'est un ordre de réduction contenant l'ordre sous-terme, i.e. pour tous les termes t, u*

$$t[u] > u .$$

Un ordre de réduction total contient l'ordre sous-terme, et par conséquent est un ordre de simplification.

Soit $\rightarrow$ une relation binaire sur T. On note $\xrightarrow{*}$ la fermeture réflexive et transitive de $\rightarrow$. Un élément t d'un ensemble T est réductible par $\rightarrow$ s'il existe un t' différent de t dans T tel que $t \rightarrow t'$. Dans le cas contraire, il est irréductible. On appelle forme normale de t tout élément t' irréductible tel que $t \xrightarrow{*} t'$.

Définition 2.23. *Soient T un ensemble et $\rightarrow$ une relation binaire sur T. On dit que la relation $\rightarrow$*
- *termine (ou est terminante) si $\rightarrow$ est bien fondée ;*
- *est confluente si pour tout $t, u, v \in T$ si $t \xrightarrow{*} u$ et $t \xrightarrow{*} u$ alors il existe un $t' \in T$ tel que $u \xrightarrow{*} t'$ et $v \xrightarrow{*} t'$;*
- *est localement confluente si pour tout $t, u, v \in T$ si $t \rightarrow u$ et $t \rightarrow v$ alors il existe un $t' \in T$ tel que $u \xrightarrow{*} t'$ et $v \xrightarrow{*} t'$.*

2.3 Théories des modèles de premier ordre

Nous introduisons dans cette section des concepts sémantiques associés aux concepts syntaxiques introduits précédemment. Nous commençons par présenter la notion de structure ensuite nous introduisons des relations entre les structures elles-même (e.g. des homomorphismes), et des relations entre les structures et les formules (e.g. la validité, la satisfiabilité). La plupart des notions introduites proviennent de [Hod94].

2.3.1 Modèle

Définition 2.24 (Structure)**.** *Soit Σ une signature. Une Σ-structure $\mathcal{A}$ est une paire (A, I^Σ) telle que A est appelé le domaine de $\mathcal{A}$, et I^Σ est une fonction qui associe à :*
- *chaque symbole de constante c dans Σ un élément $c^\mathcal{A}$ de A, et*
- *chaque symbole de fonction f d'arité n dans Σ une fonction $f^\mathcal{A}$ de A^n vers A, et*
- *chaque symbole de relation P d'arité n dans Σ une relation sur A^n.*

Une structure est une algèbre si sa signature ne contient pas de symbole de relation. La cardinalité $Card(\mathcal{A})$ d'une structure $\mathcal{A}$ est simplement la cardinalité de A. Une structure est triviale si sa cardinalité est égale à 1, et non triviale dans le cas contraire.

Définition 2.25 (Réduction de structure)**.** *Soient Ω et Σ deux signatures telles que $\Sigma \subseteq \Omega$. La Σ-réduction d'une Ω-structure $\mathcal{A} = (A, I^\Omega)$ est la structure $\mathcal{A}^\Sigma$, obtenue en réduisant I^Ω aux symboles de Σ seulement. On dit aussi que $\mathcal{A}$ est une expansion de $\mathcal{A}^\Sigma$.*

Définition 2.26 (Valuation)**.** *Soient $\mathcal{A}$ une Σ-structure et X un ensemble de variables. Une valuation de X est une application de X dans A.*

Définition 2.27 (Interprétation)**.** *Soient $\mathcal{A}$ une Σ-structure, X un ensemble de variables et α une valuation de X dans A. La paire $(\mathcal{A}, \alpha)$ définit une interprétation, i.e. une application d'un ensemble de Σ-formules φ à $\{\top, \bot\}$ où $Var(\varphi) \subseteq X$.*

Définition 2.28 (Satisfiabilité et validité)**.** *Soient $\mathcal{A}$ une Σ-structure et φ une Σ-formule. On dit que*

- *une valuation α de $Var(\varphi)$ dans A satisfait φ dans $\mathcal{A}$ ou de façon équivalente φ est satisfiable dans $\mathcal{A}$, noté $(\mathcal{A}, \alpha) \models \varphi$, si $(\mathcal{A}, \alpha)$ évalue φ en $\top$ et*
- *$\mathcal{A}$ est un modèle de φ ou de façon équivalente φ est valide dans $\mathcal{A}$, noté $\mathcal{A} \models \varphi$ si $(\mathcal{A}, \alpha) \models \varphi$ pour toute valuation α de $Var(\varphi)$ dans A.*

Dans certain cas, on considère la notation $(\mathcal{A}, \alpha) \models \varphi$ interchangeable avec $\mathcal{A} \models \varphi(a_1, \ldots, a_n)$ si $Var(\varphi) = \{x_1, \ldots, x_n\}$ et $\alpha(x_i) = a_i$ pour $i \in \{1, \ldots, n\}$.

Définition 2.29. *Soient $\mathbb{K}$ une classe de Σ-structures et φ, ϕ des Σ-formules. On dit que*

- *φ est satisfiable dans $\mathbb{K}$ si φ est satisfiable dans une Σ-structure $\mathcal{A} \in \mathbb{K}$;*
- *φ est valide dans $\mathbb{K}$, noté $\mathbb{K} \models \varphi$, si $\mathcal{A} \models \varphi$ pour toute Σ-structure $\mathcal{A} \in \mathbb{K}$;*
- *φ et ϕ sont équisatisfiables dans $\mathbb{K}$ lorsque φ est satisfiable dans $\mathbb{K}$ si et seulement si ϕ est satisfiable dans $\mathbb{K}$.*

Dans le cas où $\mathbb{K}$ est la classe de toutes les Σ-structures, on dit simplement que φ est satisfiable (respectivement valide ou équisatisfiable à ϕ).

Le résultat suivant est bien connu en théorie des modèles et nous l'utiliserons pour prouver le théorème de combinaison du Chapitre 3.

Proposition 2.1. *Si $\mathcal{A}$ est une Σ-structure, $\varphi(\tilde{x})$ est une Σ-formule et α une valuation de $\tilde{x}$ dans A, alors pour toute expansion $\mathcal{A}'$ de $\mathcal{A}$ à la signature $\Sigma' \supseteq \Sigma$*

$$(\mathcal{A}, \alpha) \models \varphi \text{ ssi } (\mathcal{A}', \alpha) \models \varphi\,.$$

Définition 2.30 (Homomorphismes)**.** *Soient $\mathcal{A}$, $\mathcal{B}$ deux Σ-structures. Une application $h : A \to B$ est*

1. *un homomorphisme de $\mathcal{A}$ dans $\mathcal{B}$ si*
   - *pour chaque constante $c \in \Sigma$, $h(c^{\mathcal{A}}) = c^{\mathcal{B}}$, et*
   - *pour tout symbole de relation P d'arité n dans Σ et $(a_1, \ldots, a_n) \in A^n$, si $(a_1, \ldots, a_n) \in P^{\mathcal{A}}$ alors $(h(a_1), \ldots, h(a_n)) \in P^{\mathcal{B}}$, et*
   - *pour tout symbole de fonction f d'arité n dans Σ et $(a_1, \ldots, a_n) \in A^n$,*

   $$h(f^{\mathcal{A}}(a_1, \ldots, a_n)) = f^{\mathcal{B}}(h(a_1), \ldots, h(a_n))\,.$$

2. *un plongement de $\mathcal{A}$ dans $\mathcal{B}$ si*
   - *h est un homomorphisme de $\mathcal{A}$ dans $\mathcal{B}$, et*
   - *h est injectif, et*
   - *pour tout symbole de relation P d'arité n dans Σ et $(a_1, \ldots, a_n) \in A^n$, $(a_1, \ldots, a_n) \in P^{\mathcal{A}}$ si et seulement si $(h(a_1), \ldots, (a_n)) \in P^{\mathcal{B}}$.*
3. *un isomorphisme est un plongement surjectif.*

Un homomorphisme $h : A \to A$ est appelé endomorphisme. Un isomorphisme $h : A \to A$ est appelé automorphisme.

On dit que deux Σ-structures $\mathcal{A}$ et $\mathcal{B}$ sont isomorphes, notée $\mathcal{A} \cong \mathcal{B}$ s'il existe un isomorphisme de $\mathcal{A}$ dans $\mathcal{B}$. On écrit $h : \mathcal{A} \cong \mathcal{B}$ pour noter que h est un isomorphisme de $\mathcal{A}$ dans $\mathcal{B}$.

Définition 2.31 (Sous-structure)**.** *Soient $\mathcal{A}$, $\mathcal{B}$ deux Σ-structures telles que $A \subseteq B$. On dit que $\mathcal{A}$ est une sous-structure de $\mathcal{B}$, noté $\mathcal{A} \subseteq \mathcal{B}$, si l'application définie par l'inclusion de A dans B est un plongement de $\mathcal{A}$ dans $\mathcal{B}$.*

2.3.2 Théorie

Définition 2.32 (Théorie)**.** *Une théorie est un ensemble de sentences. Une Σ-théorie est une théorie dont toutes les sentences sont de signature Σ.*

Toutes les théories considérées tout au long de ce document sont des théories de premier ordre avec égalité, i.e. le symbole $=$ est toujours interprété comme l'identité.

Définition 2.33. *Une Σ-structure $\mathcal{A}$ est un modèle d'une Σ-théorie T, ou un T-modèle, si $\mathcal{A}$ est un modèle de toutes les sentences de T.*

On désigne par $Mod^{\Sigma}(T)$ l'ensemble de tous les Σ-modèles de T. En général, Σ_T désigne la plus petite signature Σ telle que T est une Σ-théorie, et $Mod(T)$ désigne l'ensemble de tous les Σ_T-modèles de T. Sans indication contraire, quand on dit $\mathcal{A}$ est un modèle d'une théorie T, cela veut dire que $\mathcal{A}$ est un Σ_T-modèle de T, noté aussi $\mathcal{A} \in Mod(T)$. Quand Σ est sans ambiguïté dans le contexte, on utilise $Mod(T)$ au lieu de $Mod^{\Sigma}(T)$.

Définition 2.34. *Si T est une Σ-théorie et φ est une Σ-sentence, on dit que φ est une conséquence de T, notée $T \models \varphi$, si $Mod^{\Sigma}(T) \models \varphi$.*

Définition 2.35. *Soit T une Σ-théorie. On dit que*
- *T est consistante si $Mod^{\Sigma}(T)$ est non vide et inconsistante dans le cas contraire ;*
- *T est complète si pour toute Σ-sentence φ soit $T \models \varphi$ soit $T \models \neg\varphi$;*
- *T est non triviale si $Mod^{\Sigma}(T)$ contient une Σ-structure non triviale ;*
- *T est axiomatisée par un ensemble $Ax(T)$ de sentences si toutes les sentences de T sont des conséquences de $Ax(T)$.*

Par simplicité on identifie souvent une Σ-théorie et toutes les Σ-sentences qui sont des conséquences de T.

Définition 2.36 (Satisfiabilité modulo)**.** *Une Σ-formule φ est satisfiable dans une Σ-théorie T, ou de façon équivalente $T \models \tilde{\exists}\varphi$, si elle est satisfiable dans $Mod^{\Sigma}(T)$.*

Définition 2.37 (Validité modulo)**.** *Une Σ-formule φ est valide dans une Σ-théorie T, ou de façon équivalente $T \models \tilde{\forall}\varphi$, si elle est valide dans $Mod^{\Sigma}(T)$.*

Définition 2.38 (Équisatisfiabilité modulo)**.** *Deux Σ-formules sont équisatisfiables dans une Σ-théorie T si elles sont équisatisfiables dans $Mod^{\Sigma}(T)$.*

Par simplicité, nous dirons qu'une formule φ est T-satisfiable (respectivement T-valide, T-équisatisfiable à une formule ϕ) au lieu de dire que φ est satisfiable (respectivement valide, équisatisfiable à une formule ϕ) dans T.

2.4 Procédure de satisfiabilité

Nous nous intéressons dans le contexte de la thèse à l'intégration de procédures de satisfiabilité dans des systèmes de déduction automatique. En général, une procédure de satisfiabilité fonctionne pour un domaine de calcul (ou une théorie) et pour une classe de formule particulière.

Définition 2.39. *Soient T une théorie et Φ une classe de formules. Une procédure de satisfiabilité modulo T, ou procédure de T-satisfiabilité, pour Φ est un algorithme qui détermine la satisfiabilité modulo T, ou T-satisfiabilité, pour toute formule $\phi \in \Phi$.*

Dans les cas où on est amené à décider la satisfiabilité des formules sans quantificateurs, les variables libres dans des formules sont implicitement et existentiellement quantifiées. De façon équivalents ces variables peuvent être considérées comme des constantes. C'est pour cette raison que lorsque l'on considère le problème de satisfiabilité, une formule sans quantificateurs peut être vue comme une formule *close* (*ground* en anglais). Typiquement, lorsque l'on considère des procédures de satisfiabilité dérivées par saturation (Chapitre 4 et Chapitre 5), pour éviter la confusion entre variables libres et variables implicitement et universellement quantifiées, nous utilisons des constantes au lieu de variables existentiellement quantifiées et des formules closes au lieu des formules sans quantificateurs. De plus, toute formule sans quantificateur est équivalente à une formule en DNF, nous pouvons donc, sans perte de généralité, considérer la satisfiabilité des conjonctions (ou ensemble) de littéraux clos au lieu des formules sans quantificateurs arbitraires.

2.5 Exemples de théories

Nous allons présenter quelques théories souvent utilisées en vérification ainsi que leurs procédures de satisfiabilité, à savoir la théorie de l'égalité, des théories de structures de données (listes, tableaux) et l'arithmétique linéaire. Nous avons fait le choix de présenter une procédure de satisfiabilité pour la théorie de l'égalité, qui sert à un double objectif : premièrement elle nous permet d'avoir un comparatif entre l'approche *ad hoc* (voir Figure 2.1) et l'approche uniforme par saturation (voir Chapitre 4) pour construire des procédures de satisfiabilité pour des théories axiomatisées par un ensemble fini de clauses ; et deuxièmement elle est le point de départ pour l'étude de la coopération entre celle-ci et la saturation (présentée dans le Chapitre 4). Pour les théories des structures de données et l'arithmétique linéaire, nous nous contentons d'une présentation informelle des idées principales de leurs procédures de satisfiabilité.

2.5.1 Théorie de l'égalité

La théorie $\mathcal{E}$ de l'égalité n'a aucun axiome. De ce fait, tous les symboles de fonction ou de relation sont libres (ou non interprétés). Le problème de satisfiabilité des formules sans quantificateurs a été montré décidable par Ackerman [Ack54]. Des procédures de satisfiabilité efficaces basées sur la clôture de congruence ont été successivement proposées par Shostak [Sho79], Downey, Sethi et Tarjan [DST80], et Nelson et Oppen [NO80].

A l'origine, la clôture de congruence ($\mathcal{CC}$) a été conçue pour construire les classes d'équivalence de termes modulo un ensemble d'égalités entre termes. De plus, une procédure de satisfiabilité pour les ensembles des littéraux clos peut être construite à partir de la clôture de congruence. En effet, un ensemble de littéraux est insatisfiable si et seulement si nous avons une diségalité entre deux termes qui appartiennent à la même classe d'équivalence. Dans ce qui suit, nous donnons un modèle abstrait [Fon04], qui a été inspiré par [BTV03], des implantations de la clôture de congruence

[DST80, NO80, Sho79, Sho84, Fre99, NO03, Fon04] sans tenir compte des détails de complexité, d'efficacité et d'incrémentalité.

Un algorithme simple de clôture de congruence peut être obtenu par une application exhaustive et non-déterministe des règles décrites dans la Figure 2.1. La configuration initiale est $\mathcal{C}_0; E_0$, où E_0 est un ensemble d'égalités closes et $\mathcal{C}_0$ est une partition de tous les termes dans E_0 telle que chaque terme est dans sa propre classe. La configuration finale est $\mathcal{C}; \emptyset$ où $\mathcal{C}$ représente la Clôture de Congruence de E_0, c'est-à-dire x et y sont dans la même classe dans $\mathcal{C}$ si et seulement si $x = y$ est conséquence de E_0.

Suppression

$$\frac{\mathcal{C} \cup \{A\}; E \cup \{x = y\}}{\mathcal{C} \cup \{A\}; E} \quad \textbf{si } (x, y) \subseteq A$$

Fusion

$$\frac{\mathcal{C} \cup \{A, B\}; E \cup \{x = y\}}{\mathcal{C} \cup \{A \cup B\}; E} \quad \textbf{si } \begin{cases} x \in A \\ y \in B \\ A \neq B \end{cases}$$

Congruence

$$\frac{\mathcal{C} \cup \{A, B\}; E}{\mathcal{C} \cup \{A, B\}; E \cup \{f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)\}}$$

$$\textbf{si } \begin{cases} f(x_1, \ldots, x_n) \in A \\ f(y_1, \ldots, y_n) \in B \\ A \neq B \\ \forall i \in \{1, \ldots, n\}, (x_i, y_i) \in D_i, D_i \in \mathcal{C} \cup \{A, B\} \\ f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \notin E \end{cases}$$

FIGURE 2.1 – Règles de $\mathcal{CC}$

Nous avons le résultat suivant qui est classique dans la littérature.

Théorème 2.1 ([Fon04])**.** *$\mathcal{CC}$ satisfait les propriétés suivantes :*

- *toute $\mathcal{CC}$-dérivation termine, et*
- *si la configuration initiale est $\mathcal{C}_0; E_0$ et la configuration finale est $\mathcal{C}; \emptyset$, alors pour tous les termes x, y dans E, $E \models x = y$ si et seulement si x et y sont dans la même classe dans $\mathcal{C}$.*

Une procédure de satisfiabilité pour des ensembles de littéraux clos peut être facilement obtenue à partir de la clôture de congruence car l'insatisfiabilité d'un ensemble S de littéraux clos est toujours due à une seule diségalité $x \neq y$ telle que x et y sont dans la même classe de congruence induite par le sous-ensemble E de S contenant toutes les égalités.

Dans le Chapitre 4, nous montrons que la satisfiabilité d'un ensemble de littéraux clos peut être également décidée par saturation mais cette méthode n'est bien évidemment pas comparable à la clôture de congruence en terme de performance et de

flexibilité. C'est pour cette raison qu'il est préférable dans certain cas de faire coopérer la clôture de congruence et la saturation pour traiter la satisfiabilité des ensembles de clauses non closes.

2.5.2 Théories de structures de données

Théorie des listes

Soit $\Sigma_{\mathcal{L}} = \{cons, car, cdr\}$. La théorie des listes $\mathcal{L}$ a la signature $\Sigma_{\mathcal{L}}$ et les axiomes suivants :

- un axiome de construction

$$(\forall X)(cons(car(X), cdr(X)) = X)\,,$$

- deux axiomes de sélection

$$(\forall X, Y)(car(cons(X, Y)) = X)$$

$$(\forall X, Y)(cdr(cons(X, Y)) = Y)\,.$$

Le problème de $\mathcal{L}$-satisfiabilité des formules sans quantificateurs a été montré décidable par Shostak [Sho84]. Ce problème peut être également traité par saturation [ARR03] (voir le Chapitre 4 pour plus de détails).

Théorie des tableaux

La théorie $\mathcal{A}$ des tableaux a la signature $\Sigma_{\mathcal{A}} = \{select, store\}$ et les axiomes suivants

$$(\forall A, I, E)(select(store(A, I, E), I) = E)\,,$$

$$(\forall A, I, J, E)(I = J \vee select(store(A, I, E), J) = select(A, J))\,,$$

et éventuellement l'axiome d'extensionalité

$$(\forall A, A')((\forall I)(select(A, I) = select(A', I)) \Rightarrow A = A')\,.$$

Le problème de satisfiabilité des formules sans quantificateurs dans la théorie des tableaux (avec ou sans l'extensionalité) a été montré décidable dans [SBDL01, ARR03].

2.5.3 Arithmétique linéaire

Nous présentons quelques fragments de l'arithmétique, qui sont souvent utilisés en vérification, parmi-eux l'arithmétique de Presburger ou l'arithmétique linéaire sur les entiers, sur les rationnels ou sur les réels.

Arithmétique de Presburger

Considérons la plus petite classe de structure dont le domaine est $\mathbb{N}$ et la signature consiste en

- une constante 0 dont l'interprétation est le zéro des entiers naturels,
- un symbole de fonction S, dont l'interprétation est le successeur,
- un symbole de fonction $+$, dont l'interprétation est l'addition des entiers naturels,

– un symbole de relation $>$, dont l'interprétation est l'ordre standard sur les entiers naturels

L'arithmétique de Presburger est une théorie du premier ordre définie comme l'ensemble des sentences satisfiables dans cette classe de structures.

L'arithmétique de Presburger est intéressante car le problème de validité dans celle-ci est décidable. Presburger a montré ce résultat en utilisant la technique d'élimination des quantificateurs. Presburger a également montré que son arithmétique est consistante et complète.

Le meilleur algorithme d'élimination des quantificateurs a été proposé par Cooper [Coo72], qui a au plus une complexité $2^{2^{2^n}}$, où n est la taille de la formule. Fischer et Rabin [FR74] ont prouvé que toute algorithme d'élimination des quantificateurs pour l'arithmétique de Presburger doit avoir une complexité d'au moins 2^{2^n}.

Pour le problème de validité (sans quantificateur), une meilleure complexité peut être obtenue. En effet, le problème validité sans quantificateur pour l'arithmétique de Presburger est NP-complet [Pap81].

Arithmétique linéaire

L'arithmétique linéaire sur les entiers (resp. rationnels, réels) est une théorie de premier ordre des entiers (resp. rationnels, réels) avec l'addition. La signature de l'arithmétique linéaire est la même que celle de l'arithmétique de Presburger. Seule son domaine diffère de celui de l'arithmétique de Presburger.

En vérification, on est souvent amené à décider le problème de satisfiabilité dans l'arithmétique linéaire. Il existe plusieurs procédures de satisfiabilité pour l'arithmétique linéaire sur les rationnels (resp. réels) : la méthode du point intérieur à la complexité polynomiale tandis que la méthode du simplex et celle de Fourrier-Motzkin ont une complexité exponentielle (voir e.g. [Sch86]).

Chapitre 3

Combinaison de procédures de satisfiabilité

Le principe de modularité est omniprésent en informatique. Il consiste à décomposer un problème en sous-problèmes plus faciles à résoudre. Dans le domaine de la déduction automatique, on dit qu'une propriété $\mathcal{P}$ est *modulaire* si le fait que $\mathcal{P}$ soit vraie dans deux théories T_1 et T_2 implique qu'elle le soit encore dans l'union des théories $T_1 \cup T_2$. La modularité a été étudiée dans plusieurs domaines de la déduction automatique, que ce soit l'unification, le filtrage, la résolution de contraintes, les systèmes de réécriture, le problème du mot ou la satisfiabilité.

L'unification (modulo une théorie équationnelle) est une des premières procédures étudiées en déduction automatique. Une première solution pour ce type de combinaison a été proposée par Stickel dans [Sti81] pour l'unification modulo associativité et commutativité en présence de fonctions non interprétées. Ce travail a inspiré une série de résultats sur la combinaison d'algorithmes d'unification pour des classes de théories plus générales [Kir89, Yel87, Her86, Tid86a, BJSS88, SS89, BS96]. Dans ce contexte, Schmidt-Schauß [SS89] a donné un algorithme de combinaison pour des théories équationnelles arbitraires ayant des signatures disjointes. Ce résultat a été ensuite amélioré par Boudet [Bou93] puis par Baader et Schulz [BS96]. Le filtrage est un problème étroitement lié à l'unification. Les résultats les plus importants sur la combinaison des algorithmes de filtrage sont ceux de Nipkow [Nip91] et Ringeissen [Rin96a, Rin03]. La combinaison en résolution de contraintes a été étudiée par Kirchner et Ringeissen et ensuite amélioré par Baader et Schulz. Kirchner et Ringeissen ont considéré la combinaison des solveurs de contraintes. Dans [Rin92, KR92, Rin93, KR94a, KR94b], ils ont proposé quelques techniques généralisant les résultats existants pour combiner des algorithmes d'unification. Dans [BS95a, BS95b, BS98], Baader et Schulz ont utilisé les notions de structures libres et quasi-libres pour avoir des domaines de contraintes avec des propriétés intéressantes permettant de combiner ces domaines. Le domaine combiné dans lequel la satisfiabilité de contraintes mixtes est vérifiée est construit en prenant le produit libre amalgamé des domaines composants.

La modularité des systèmes de réécriture a été d'abord étudiée dans [Toy87a, Toy87b]. En effet, Toyama a montré, pour le cas disjoint, que la confluence est une propriété modulaire tandis que la terminaison ne l'est pas. Des résultats positifs [Rus87, Mid89, KK90] concernant la terminaison sont obtenus en rajoutant des hypothèses supplémentaires sur la forme des règles de réécriture. Middeldorp et Toyama

[MT91, MT93] se sont également intéressés à la modularité de certaines propriétés lorsque les systèmes de réécriture partagent des symboles de fonction qui sont des constructeurs.

Dans [BT97], Baader et Tinelli ont proposé une méthode de combinaison de procédures de décision du problème du mot pour les théories arbitraires dans le cas disjoint. En ce qui concerne le cas non disjoint, [BT98, DKR94] généralisent les résultats dans [Pig74, Tid86b, SS89, KR94b] aux théories partageant les constructeurs. Fiorentini et Ghilardi [FG03] ont également traité ce problème avec une approche basée sur la réécriture des produits de catégories. Récemment Baader, Ghilardi et Tinelli [BGT04] ont proposé une nouvelle approche pour le problème du mot dans la fusion des logiques modales. Ce travail réutilise et combine certains concepts et techniques développés dans [Ghi04] et [BT98].

Le problème de construction d'interpolants dans des unions de théories présente un grand intérêt en analyse de programme ou en model checking [McM03, McM04, McM05, HJMM04]. Dans [McM04], McMillan a présenté une méthode pour générer des interpolants clos à partir des preuves dans le mélange de l'arithmétique linéaire des rationnels et des fonctions non interprétées. [YM05] a généralisé le résultat de [McM04] en proposant une méthode générale pour générer des interpolants dans des combinaisons disjointes de théories. Cette méthode est basée sur la méthode de combinaison de procédures de satisfiabilité de Nelson-Oppen. Récemment, Sofronie-Stokkermans [SS06] a étudié la génération d'interpolants dans des mélanges non disjoints des extensions locales de théories. La méthode de Sofronie-Stokkermans, qui est fortement inspirée par celle de [YM05], permet de dériver un résultat qui subsume ceux de [McM04] et [YM05].

Dans le cadre de la thèse nous nous intéressons particulièrement à la combinaison des procédures de satisfiabilité dont l'intérêt applicatif se trouve dans des problèmes de vérification de logiciels ou de composants électroniques, où le problème est de décider la validité d'assertions dans des théories axiomatisant les types de données. Bien que les procédures de satisfiabilité sont connues pour (des fragments) des entiers, des réels ou des théories modélisant des structures de données comme les listes, les tableaux, les assertions des programmes sont souvent des formules mixtes composées de symboles de ces théories et de constantes ou symboles de fonction de Skolemisation introduits au cours de certaines transformations. Le premier résultat de combinaison a été donné par Shostak dans [Sho79], qui a considéré l'arithmétique de Presburger et des symboles de fonction libres (ou non interprétés). Shostak a ensuite étendu son résultat à des cas plus généraux dans [Sho84]. Malgré certains énoncés erronés la méthode de combinaison de Shostak présente de réels intérêts dans les applications concrètes, ce qui a inspiré une série de travaux [CLS96, RS01, BDS02, Kap02, Gan02, CK03b, RS02, CK03a, MZ03, RRT04] visant à corriger et généraliser [Sho84]. Une autre méthode plus générale, qui a été proposée par Nelson-Oppen dans [NO79], combine les procédures de satisfiabilité connues pour différentes théories composantes afin d'obtenir une procédure de satisfiabilité pour l'union, à condition que les théories composantes soient stablement infinies et ne partagent pas de symbole autre que l'égalité. Les directions de recherche qui ont pour but d'étendre ou de compléter le résultat de Nelson-Oppen concernent à la fois le cas disjoint et le cas non disjoint. Parmi ceux-ci, les travaux les plus significatifs sur la combinaison disjointe sont [TZ05, Gan02, BGN$^+$06] qui visent à affaiblir la condition de stable infinité des théories composantes. La combinaison non disjointe a

été abordée indépendemment et différemment par Ringeissen et Tinelli [RT03] et par Ghilardi [Ghi04]. Ghilardi et Nicolini et Zucchelli ont ensuite généralisé [Ghi04] à un cadre très général et abstrait dans [GNZ05].

Ce chapitre a pour but de donner un aperçu global de l'ensemble des résultats essentiels sur la décidabilité du problème de satisfiabilité dans l'union des théories pour lesquelles le problème de satisfiabilité est décidable. Nous commençons par présenter un théorème très général sur la combinaison non disjointe, à partir duquel nous pouvons retrouver de façon uniforme les résultats de combinaison dans [TZ05, Gan02, BGN+06, RT03] en explorant différentes classes de théories satisfaisant les conditions d'application de celui-ci.

3.1 Théorème de combinaison

Le théorème de combinaison non disjointe a été énoncé indépendemment par Ringeissen [Rin96b] et par Tinelli et Harandi [TH96], et reformalisé plus tard dans [RT03, MZ03]. Intuitivement, ce théorème dit que la conjonction de deux formules construites sur deux signatures est satisfiable exactement quand ces deux formules sont satisfiables dans deux structures qui partagent la même structure sur la signature commune. Dans le cas où les deux formules partagent des variables libres, les valuations qui les évaluent à vrai dans les structures correspondantes doivent coïncider sur l'ensemble des variables communes.

Théorème 3.1 (Théorème de combinaison non disjointe)**.** *Soient Σ_1 et Σ_2 deux signatures. Soient Φ_i un ensemble de Σ_i-formules et $V_i = Var(\Phi_i)$, pour $i = 1, 2$. Alors les conditions suivantes sont équivalentes*

(i) $\Phi_1 \cup \Phi_2$ *est satisfiable,*

(ii) il existe une Σ_i-interprétation $(\mathcal{M}_i, \alpha_i)$ qui satisfait Φ_i, pour $i = 1, 2$ et il existe un isomorphisme $h : \mathcal{M}_1^{\Sigma_1 \cap \Sigma_2} \cong \mathcal{M}_2^{\Sigma_1 \cap \Sigma_2}$ tel que pour tout $x \in V_1 \cap V_2$, $h(\alpha_1(x)) = \alpha_2(x)$.

Démonstration. $(i) \Rightarrow (ii)$. Supposons que $\Phi_1 \cup \Phi_2$ est satisfiable dans une $(\Sigma_1 \cup \Sigma_2)$-interprétation $(\mathcal{M}, \alpha)$. En prenant la réduction de $\mathcal{M}$ aux signatures composantes et la restriction de α aux sous-domaines correspondants, nous obtenons deux interprétations $(\mathcal{M}^{\Sigma_1}, \alpha_1)$ et $(\mathcal{M}^{\Sigma_2}, \alpha_2)$ telles que $\mathcal{D}om(\alpha_i) = V_i$ pour $i = 1, 2$. Il est évident que $(\mathcal{M}^{\Sigma_1}, \alpha_1)$ et $(\mathcal{M}^{\Sigma_2}, \alpha_2)$ satisfont les conditions de (ii).
$(ii) \Rightarrow (i)$. En supposant (ii), nous allons construire une interprétation $(\mathcal{M}, \alpha)$ qui satisfait $\Phi_1 \cup \Phi_2$. $(\mathcal{M}, \alpha)$ est défini de telle sorte que :

- $M = M_1$,
- pour chaque symbole de constante c :

$$c^{\mathcal{M}} = \begin{cases} c^{\mathcal{M}_1}, & \text{si } c \in \Sigma_1, \\ h^{-1}(c^{\mathcal{M}_2}), & \text{si } c \in \Sigma_2 \setminus \Sigma_1. \end{cases}$$

- pour chaque symbole de fonction f d'arité n :

$$f^{\mathcal{M}}(a_1, \ldots, a_n) = \begin{cases} f^{\mathcal{M}_1}(a_1, \ldots, a_n), & \text{si } f \in \Sigma_1, \\ h^{-1}(f^{\mathcal{M}_2}(h(a_1), \ldots, h(a_n))), & \text{si } f \in \Sigma_2 \setminus \Sigma_1. \end{cases}$$

– pour chaque symbole de relation P d'arité n :

$$(a_1, \ldots, a_n) \in P^{\mathcal{M}} \Leftrightarrow \begin{cases} (a_1, \ldots, a_n) \in P^{\mathcal{M}_1}, & \text{si } P \in \Sigma_1, \\ (h(a_1), \ldots, h(a_n)) \in P^{\mathcal{M}_2}, & \text{si } P \in \Sigma_2 \setminus \Sigma_1. \end{cases}$$

– pour chaque variable x :

$$\alpha(x) = \begin{cases} \alpha_1(x), & \text{si } x \in V_1, \\ h^{-1}(\alpha_2(x)), & \text{si } x \in V_2 \setminus V_1. \end{cases}$$

Par construction, $\mathcal{M}^{\Sigma_1} \cong \mathcal{M}_1$ et $\mathcal{M}^{\Sigma_2} \cong \mathcal{M}_2$. Par la Proposition 2.1, $(\mathcal{M}, \alpha)$ satisfait $\Phi_1 \cup \Phi_2$. □

Dans le cas où les signatures des théories composantes sont disjointes, la situation devient beaucoup plus simple car nous n'avons qu'à considérer le symbole d'égalité et les variables partagées. L'isomorphisme entre les structures de signature commune se réduit simplement à une bijection entre les domaines des structures composantes. Cette observation nous conduit rapidement au résultat suivant.

Théorème 3.2 (Théorème de combinaison disjointe)**.** *Soient Σ_1 et Σ_2 deux signatures telles que $\Sigma_1 \cap \Sigma_2 = \emptyset$. Soient Φ_i un ensemble de Σ_i-formules et $V_i = Var(\Phi_i)$, pour $i = 1, 2$. Alors les conditions suivantes sont équivalentes*

(i) *$\Phi_1 \cup \Phi_2$ est satisfiable,*

(ii) *il existe une Σ_i-interprétation $(\mathcal{M}_i, \alpha_i)$ qui satisfait Φ_i, pour $i = 1, 2$ telles que*
  – *$|M_1| = |M_2|$, et*
  – *pour toute variable $x, y \in V_1 \cap V_2$, $\alpha_1(x) = \alpha_1(y)$ si et seulement si $\alpha_2(x) = \alpha_2(y)$.*

Démonstration. $(i) \Rightarrow (ii)$. Supposons que $\Phi_1 \cup \Phi_2$ est satisfiable dans une $(\Sigma_1 \cup \Sigma_2)$-interprétation $(\mathcal{M}, \alpha)$. En prenant la réduction de $\mathcal{M}$ aux signatures composantes et la restriction de α aux sous-domaines correspondants, nous obtenons deux interprétations $(\mathcal{M}^{\Sigma_1}, \alpha_1)$ et $(\mathcal{M}^{\Sigma_2}, \alpha_2)$ telles que $\mathcal{D}om(\alpha_i) = V_i$ pour $i = 1, 2$. Il est évident que $(\mathcal{M}^{\Sigma_1}, \alpha_1)$ et $(\mathcal{M}^{\Sigma_2}, \alpha_2)$ satisfont les conditions de (ii).
$(ii) \Rightarrow (i)$. On note α^V la restriction de α à un sous-ensemble de variable $V \subseteq \mathcal{D}om(\alpha)$. En supposant (ii), il est facile de voir qu'il existe une bijection h entre $\mathcal{R}an(\alpha_1^V)$ $\mathcal{R}an(\alpha_2^V)$ définie par $h(\alpha_1^V(x)) = \alpha_2^V(x)$. Or $|\mathcal{M}_1| = |\mathcal{M}_2|$, h peut donc s'étendre à une bijection entre M_1 et M_2. Nous pouvons ainsi appliquer le théorème de combinaison non disjointe pour conclure (i). □

Il est important de souligner qu'il s'agit dans notre contexte de décider le problème de satisfiabilité des ensembles de formules sans quantificateurs. Ce qui signifie que les variables libres dans des formules sont implicitement existentiellement quantifiées ou de façon équivalente ces variables peuvent être considérées comme des constantes avec lesquelles on enrichit les signatures. En effet, en Skolemisant les variables libres nous obtenons des formules closes, qui peuvent être construites à partir de la signature originale et des constantes de Skolemisation. C'est pour cette raison que les deux théorèmes de combinaison s'applique également dans des contextes où on utilise des constantes rajoutées à la signature originale au lieu des variables et des interprétations de constantes au lieu de valuations de variables. Les lecteurs souhaitant avoir plus de détails à ce sujet peuvent se référer au livre [Hod94], Section 1.4, page 15.

3.2 Combinaison disjointe

Dans ce qui suit, nous allons classifier les résultats importants relatifs à la combinaison de procédures de satisfiabilité en fonction de critères opérationnels ou théoriques.

Du point de vue opérationnel, nous pouvons classifier les méthodes de combinaison en deux approches : black-box et glass-box. L'approche black-box suppose l'existence d'une procédure de satisfiabilité P_1 pour une théorie T_1 et d'une procédure de satisfiabilité P_2 pour une théorie T_2. La procédure de combinaison fait coopérer les procédures P_1 et P_2 afin d'obtenir une procédure de satisfiabilité pour l'union de T_1 et T_2. Les méthodes de combinaison basées sur la propagation de variables partagées, comme celle développée par Nelson-Oppen [NO79], font partie de cette approche. L'approche glass-box suppose que les théories composantes disposent plus de fonctionalités pour pouvoir ensuite les utiliser dans le schéma de combinaison. L'exemple typique d'une approche glass-box est la combinaison de Shostak [Sho84] dans laquelle les fonctionalités offertes sont beaucoup plus raffinées, e.g. solveurs, canoniseurs.

Du point de vue plus théorique, nous pouvons classifier les approches de combinaison en deux catégories selon les hypothèses à satisfaire par les théories composantes : symétrique ou asymétrique. Dans l'approche symétrique [NO79, BGN$^+$06], les théories composantes satisfont les mêmes hypothèses qui assurent la correction de la procédure de combinaison tandis que l'approche asymétrique [TZ05, RRZ05, FRZ05, FG04, Fon04] suppose des hypothèses très fortes sur une théorie composante donnée et aucune hypothèse sur les autres théories composantes.

Sans perte de généralité nous pouvons considérer seulement la satisfiabilité des ensembles (ou conjonctions) de littéraux puisqu'on peut toujours transformer une formule arbitraire sans quantificateurs en une forme normale disjonctive, i.e. en une disjonction de conjonctions de littéraux.

3.2.1 Nelson-Oppen

En 1979, Nelson-Oppen ont proposé une approche de combinaison symétrique et black-box. A l'origine, cette méthode de combinaison a été conçue pour traiter la combinaison des procédures de satisfiabilité pour les théories à signatures disjointes. Elle fait coopérer les procédures de décision pour les théories du premier ordre pour obtenir une procédure de décision pour l'union des théories. Étant données n signatures $\Sigma_1, \ldots, \Sigma_n$, soient T_i les Σ_i-théories pour $i = 1, \ldots, n$. Supposons qu'il existe n procédures de satisfiabilité $P_1, \ldots, P_n$ telle que pour chaque $i = 1, \ldots, n$, P_i indique la T_i-satisfiabilité des Σ_i-formules sans quantificateurs. La méthode de Nelson-Oppen consiste à déterminer la $(T_1 \cup T_2 \cup ... \cup T_n)$-satisfiabilité des $(\Sigma_1 \cup \Sigma_2 \cup ... \cup \Sigma_n)$-formules sans quantificateurs.

Informellement, la procédure de combinaison à la Nelson-Oppen consiste en trois sous-procédures : *Purification*, *Test de satisfiabilité* et *Propagation d'égalités*.

Purification

L'idée est de transformer la formule *mixte* ϕ considérée en une conjonction $\phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_n$ de formules *pures* dans chaque théorie. Ainsi on peut appliquer des procédures déjà connues dans chacune des théories composantes pour décider la satisfiabilité de

chaque formule *pure*. Pour ce faire, on renomme tous les sous-termes étrangers d'un terme par des nouvelles variables. Cette procédure de renommage est bien évidemment terminante puisque l'ensemble des sous-termes d'une formule est fini. Et $\phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_n$ est satisfiable si et seulement si ϕ est satisfiable. Nous allons introduire quelques notions pour formaliser ces idées.

Définition 3.1. *Soit Σ_i une signature.*
- *Les éléments d'une signature Σ_i sont appelés les i-symboles.*
- *Une variable est appelée i-variable si elle est dans la théorie T_i.*
- *Un Σ-terme est un i-terme si c'est une i-variable, un i-symbole de constante ou une application d'un i-symbole de fonction.*
- *Un i-prédicat est une application d'un i-symbole de prédicat.*
- *Une i-formule atomique est un i-prédicat ou une égalité dont les membres sont des i-termes.*
- *Un i-littéral est une i-formule atomique ou la négation d'une i-formule atomique.*
- *Une occurrence de j-terme t dans un terme ou un littéral est i-étranger si $i \neq j$ et tous les super-termes de t sont des i-termes.*
- *Un i-terme ou i-littéral est* i-pur *si il ne contient que des i-symboles.*

On peut définir l'opération de *Purification* grâce à une application $\mathcal{P}$ de l'ensemble des termes dans l'ensemble des nouvelles variables. Soit φ une formule mixte et ϕ un i-littéral de φ. On remplace toutes les occurrences de j-terme t dans ϕ par $\mathcal{P}(t)$ et on ajoute l'égalité $\mathcal{P}(t) = t$ à la conjonction φ. En répétant cette opération jusqu'à ce que tous les littéraux de φ soient purs, on obtient une nouvelle formule qui est équisatisfiable à la formule de départ.

Test de satisfiabilité

Notons $\phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_n$ la conjonction de formules *pures* obtenues dans chaque théorie à la fin de l'étape précédente. Pour chaque formule i-pure ϕ_i, on applique la procédure de satisfiabilité P_i pour décider la satisfiabilité de ϕ_i. Si une des procédures retourne un résultat *insatisfiable*, la procédure de combinaison termine en retournant le résultat *insatisfiable*. Si toutes les procédures composantes retournent *satisfiable*, alors on passe à l'étape suivante.

Propagation d'égalités

Si on arrive à cette étape, cela signifie que toutes les procédures composantes ont terminé avec le résultat *satisfiable*. Il ne suffit pas de considérer indépendamment chaque théorie composante mais il faut également prendre en compte l'interaction entre les théories. Dans le cas de la combinaison disjointe, la seule interaction repose sur le partage des variables. La *Propagation d'égalités* consiste, comme son nom l'indique, à propager les égalités (ou des disjonctions d'égalités) des *variables partagées*[6] déduites dans une théorie aux autres théories. Si la propagation a lieu, on revient à l'étape *Test de satisfiabilité* pour continuer la procédure de combinaison. Nous allons voir plus en détail comment les égalités (ou les disjonctions d'égalités) sont propagées d'une théorie aux autres.

6. Une variable est partagée si c'est à la fois une i-variable et une j-variable, pour $i \neq j$.

– *Propagation d'une égalité* : supposons que $T_i \models x = y$. On intègre à chaque formule pure cette égalité. C'est-à-dire, pour chaque $j \neq i$, ϕ_j est remplacé par $\phi_j \wedge x = y$ si $T_j \not\models x = y$. Chaque procédure de décision P_j sera appliquée sur la formule $\phi_j \wedge x = y$ et ainsi de suite.
– *Propagation d'une disjonction d'égalités* : supposons que $T_1 \models \bigvee_{k=1}^{n} x_k = y_k$. On propage chaque égalité $x_k = y_k$ dans toutes les théories composantes. On aura autant de branches à considérer que d'égalités dans la disjonction. Si une des branches termine avec le résultat *satisfiable*, la procédure de combinaison termine en retournant *satisfiable*. Si toutes les branches terminent avec le résultat *insatisfiable*, la procédure de combinaison retourne *insatisfiable*.

Nelson-Oppen ont établi que leur méthode de combinaison est correcte si les théories composantes sont stablement infinies.

Définition 3.2 (Théorie stablement infinie)**.** *Une Σ-théorie T est stablement infinie si toute Σ-formule sans quantificateurs T-satisfiable est satisfiable dans un modèle de T dont la cardinalité est infinie.*

En effet, lorsque les théories composantes sont stablement infinies, si les formules pures avec les égalités propagées sont satisfiables alors elles sont satisfiables dans des modèles infinis. On peut donc appliquer le théorème de combinaison disjointe pour déduire que la formule mixte est aussi satisfiable [7].

Beaucoup de théories intéressantes en vérification possèdent la propriété de stable infinité, comme la théorie de l'égalité, la théorie des listes, la théorie des tableaux, l'arithmétique linéaire sur les entiers (les rationnels, les réels). Cependant il existe aussi des théories très simples n'ayant pas cette propriété. Soit $\Sigma = \{a, b\}$ une signature. La théorie T dont l'axiomatisation est $Ax(T) = \{\forall x(x = a \vee x = b)\}$ n'est pas stablement infinie car la cardinalité de tout modèle de T est bornée par 2.

Dans le Chapitre 5, nous présentons une méthode automatique pour vérifier une condition suffisante pour la stable infinité des théories axiomatisées par un ensemble fini de clauses.

Nelson-Oppen déterministe

La propagation d'égalités entre les théories composantes est une approche très intuitive pour prendre en compte l'interaction des théories à combiner. Néanmoins, cette technique pose un problème important dans la pratique. La propagation des disjonctions d'égalités peut dégrader la performance de la procédure de combinaison car elle est une source d'explosion combinatoire. En effet, Nelson-Oppen ont seulement fait l'hypothèse que chaque théorie composante admet une procédure de *satisfiabilité* et donc pour déduire les égalités, on est souvent amené à considérer toutes les relations d'équivalence sur l'ensemble des variables partagées. Or le nombre de telles relations d'équivalence est connu sous le *nombre de Bell* [MZ03] qui croit pllus que exponentiellement avec le nombre de variables partagées. Pour éviter ce problème, Nelson-Oppen considèrent l'hypothèse de convexité sur les théories composantes. Avec une telle hypothèse, nous avons une version complètement déterministe de la propagation des égalités. En effet, si une disjonction d'égalités entre variables est déduite dans une

7. En fait, ici nous utilisons implicitement le théorème "Upward Löwenheim-Skolem" [Hod94], qui stipule que si une théorie a un modèle infini alors elle a un modèle de toute cardinalité supérieure.

théorie alors nécessairement un de ses disjoints doit être déduit. Il n'est donc plus nécessaire de considérer des disjonctions d'égalité entre variables partagées.

Définition 3.3 (Théorie convexe)**.** *Une Σ-théorie T est convexe si pour toute conjonction Γ de Σ-littéraux et pour toute disjonction $\bigvee_{i=1}^{n} x_i = y_i$*

$$T \cup \Gamma \models \bigvee_{i=1}^{n} x_i = y_i \text{ ssi } T \cup \Gamma \models x_k = y_k \text{ pour un certain } k \in \{1, \ldots, n\}\,.$$

Les exemples de théories convexes sont la théorie de l'égalité, la théorie des listes, l'arithmétique sur les rationnels, les réels. La théorie des tableaux et l'arithmétique sur les entiers sont des exemples de théories non convexes.

3.2.2 Extensions de Nelson-Oppen

Dans le cas de la combinaison disjointe, des travaux ayant pour but d'affaiblir l'hypothèse de stable infinité ont été motivés par l'existence de plusieurs théories intéressantes en vérification qui ne sont pas stablement infinies, comme par exemple la théorie des entiers modulo n. Il est important de savoir que l'hypothèse de stable infinité sur toutes les théories composantes n'est qu'une condition suffisante pour laquelle le théorème de combinaison disjointe s'applique (dans ce cas les cardinalités des modèles des théories composantes sont toutes égales). Si nous cherchons à étendre la méthode de Nelson-Oppen aux théories non stablement infinies, nous devons classifier les théories composantes par rapport à la possibilité d'exhiber un modèle de cardinalité finie de la théorie. Dans cet esprit, [TZ05] et [BGN+06] ont proposé des solutions de deux styles différents : asymétrique et symétrique.

Cas asymétrique

Tinelli et Zarba [TZ05] ont été les premiers à proposer une approche de combinaison asymétrique qui ne suppose pas l'hypothèse de stable infinité sur toutes les théories composantes. Ils ont proposé une procédure de combinaison qui consiste à propager des égalités entre variables partagées conjointement avec une contrainte de cardinalité minimale des modèles. Cette procédure est correcte pour l'union d'une théorie *shiny* et d'une théorie arbitraire.

Définition 3.4 (Théorie "shiny")**.** *Une Σ-théorie T est*

- *stablement finie si toute Σ-formule sans quantificateurs T-satisfiable est satisfiable dans un modèle de T dont la cardinalité est finie.*
- *"smooth" si pour toute Σ-formule sans quantificateurs ϕ, pour tout modèle $\mathcal{M}$ de T qui satisfait ϕ et pour toute cardinalité $\kappa > |M|$, il existe un modèle $\mathcal{M}'$ de T qui satisfait ϕ tel que $|M'| = \kappa$.*
- *"shiny" si T est stablement finie et smooth et il existe une fonction calculable $mincard_T$ dont l'application à chaque conjonction T-satisfiable Γ de Σ-littéraux retourne une cardinalité minimal k d'un T-modèle qui satisfait Γ.*

Il résulte directement de la définition de théories shiny que lorsqu'une formule est satisfiable dans une théorie shiny, on peut calculer une cardinalité minimale telle qu'il existe toujours un modèle de cardinalité supérieur qui satisfait cette formule. Par conséquent, si on souhaite combiner une théorie shiny avec une théorie arbitraire

il suffit de communiquer la cardinalité minimale conjointement avec les égalités entre variables partagées déduites pour chaque formule satisfiable dans la théorie shiny. Si la contrainte de cardinalité minimale et les égalités entre variables partagées sont satisfiables dans l'autre théorie on est sûr d'avoir deux modèles qui satisfont les conditions d'applicabilité du théorème de combinaison disjointe.

Dans le même esprit, [RRZ05] traite la combinaison des théories modélisant les structures de données (e.g. les listes ou les tableaux) avec des théories d'éléments, qui ne sont pas nécessairement stablement infinies. Cette méthode de combinaison est basée sur la notion de "politeness", qui est en fait une généralisation de la notion de "shininess" à un cadre multi-sorté.

Cas symétrique

[BGN+06] a proposé une approche symétrique qui classifie les théories par rapport à la décidabilité du problème de satisfiabilité dans des modèles finis et infinis. La procédure de combinaison consiste à propager les égalités entre variables partagées et une contrainte de cardinalité maximale des modèles finis. Cette méthode de combinaison est correcte si les théories composantes ont la propriété d'être fortement $\exists_\infty$-décidables, que nous allons définir.

Définition 3.5. *On dit que le problème de satisfiabilité dans un modèle de cardinalité κ est décidable dans une théorie T si pour toute formule sans quantificateur Γ nous savons si $T \cup \Gamma$ est satisfiable dans un modèle de cardinalité κ.*

Définition 3.6 (Théorie fortement $\exists_\infty$-décidable)**.** *Une théorie T est fortement $\exists_\infty$-décidable si*

- *le problème de satisfiabilité dans T est décidable, et*
- *le problème de satisfiabilité dans un modèle de cardinalité κ est décidable dans T pour toute cardinalité $\kappa \in \mathbb{N} \cup \{\infty\}$.*

Il est facile de voir que l'hypothèse de $\exists_\infty$-décidabilité forte nous donne la possibilité de tester si une formule est satisfiable dans un modèle infini de la théorie. Lorsque tous les modèles composants sont infinis, nous pouvons appliquer la méthode de Nelson-Oppen. Dans le cas contraire, on est sûr que la cardinalité de tous les modèles de l'union doit être bornée par un entier positif. Il suffit donc de faire un test de type force brute sur tous les modèles de cardinalité inférieure à cette borne.

A noter que notre définition de $\exists_\infty$-décidabilité forte est légèrement différente de celle de [BGN+06] dans lequel les auteurs ont défini la $\exists_\infty$-décidabilité forte comme suit : une théorie est $\exists_\infty$-décidable si (i) le problème de satisfiabilité dans T est décidable ; et (ii) le problème de satisfiabilité dans un modèle infini est décidable dans T ; et (ii) il est décidable de vérifier si une structure finie est un modèle de T. Du point de vue de décidabilité, notre définition est légèrement plus faible que celle de [BGN+06]. Les deux définitions deviennent strictement équivalentes lorsqu'on considère les théories universelles (cf. Chapitre 7). Du point de vue opérationnel, notre définition de $\exists_\infty$-décidabilité forte permet de dériver une méthode pour combiner de façon modulaire les procédures de satisfiabilité composantes tandis que celle de [BGN+06] fournit une méthode de test de type force brute, à l'aide d'un solveur SMT pour la théorie de l'égalité, des modèles finis dont la cardinalité est bornée par un entier positif.

3.2.3 Shostak

Contrairement à l'approche de Nelson-Oppen qui est intuitive et générale, Shostak utilise une approche de type glass-box qui est plus subtile et complexe mais les expérimentations ont montré qu'elle est plus performante que celle de Nelson-Oppen. La clé de la méthode de Shostak est la clôture de congruence pour les égalités closes dans les théories dites *canonisable* et *solvable*. Nous n'allons pas présenter l'algorithme original de Shostak mais donner seulement des points clés de l'approche pour une bonne compréhension du problème. Les hypothèses sur les théories à combiner sont les suivantes :

- les littéraux considérés sont clos,
- les signatures des théories ne doivent pas contenir de symboles de prédicat,
- les signatures des théories doivent être disjointes,
- les théories doivent être *canonisables*,
- les théories doivent être *solvables*.

Avant de définir ce qu'est une théorie de Shostak, nous définissons la notion de théorie solvable.

Définition 3.7 (Solveur)**.** *Un solveur pour une Σ-théorie T est une fonction solve qui prend en entrée une égalité $s = t$ et :*

- *si $T \models s \neq t$ alors solve retourne $faux$*
- *ou bien, solve$(s = t)$ retourne une substitution $\sigma = \{x_1 \leftarrow t_1, ..., x_n \leftarrow t_n\}$ telle que :*
  - *$x_i \in Var(s = t)$ pour tout i,*
  - *$x_i \notin Var(t_j)$ pour tout i, j*
  - *l'équivalence suivante est T-valide :*

$$s = t \leftrightarrow (\exists \tilde{y}) \bigwedge_{i=1}^{n} x_i = t_i$$

où $\tilde{y}$ est l'ensemble des nouvelles variables introduites et :

$$\tilde{y} = \bigcup_{i=1}^{n} Var(t_i) \setminus Var(s = t)$$

Une théorie est solvable si elle admet un solveur.

L'utilisation de la fonction *solve* a deux buts. Le premier est de déterminer rapidement l'insatisfiabilité si la fonction retourne $faux$. Le deuxième est que la fonction *solve* permet d'identifier grâce à la notion de canoniseur, les classes d'équivalence de variables qui sont utilisées dans la clôture de congruence.

Définition 3.8 (Canoniseur)**.** *Un canoniseur d'une Σ-théorie Th est une fonction idempotente canon : $T(\Sigma^F, X) \rightarrow T(\Sigma^F, X)$ telle que $Th \models a = b$ ssi canon$(a) \equiv$ canon(b).*

La fonction *canon* prend en entrée un terme et renvoie sa forme canonique. Cette forme canonique représente une classe d'équivalence de termes. L'utilisation d'un canoniseur permet d'identifier les classes d'équivalence de termes modulo la théorie et par conséquent de résoudre le problème du mot qui consiste à déterminer si deux termes sont égaux ou non.

Nous adoptons ici la nouvelle définition des théories de Shostak, donnée par [Gan02], dans laquelle la propriété de convexité (cf Définition 3.3) est une hypothèse fondamentale permettant d'assurer la méthode de combinaison de Shostak.

Définition 3.9 (Théorie de Shostak)**.** *Une Σ-théorie est une théorie de Shostak si :*
- *elle est solvable,*
- *elle admet un canoniseur,*
- *elle est convexe.*

Dans [Sho84], Shostak a proposé une procédure de satisfiabilité pour l'union d'une théorie Shostak et la théorie de l'égalité. Étant donnée une conjonction d'égalités et d'inégalités mixtes, la procédure applique d'abord le *solveur* à chaque égalité. Si le résultat est *faux*, elle retourne *insatisfiable*. Sinon, une substitution est obtenue, on l'applique ensuite aux autres égalités et inégalités. Le canoniseur est appelé sur chaque terme substitué. La clôture de congruence est utilisée pour fusionner les termes dans la même classe d'équivalence et ainsi de suite... Shostak prétendait également que la classe des théories de Shostak est close par l'opération d'union : en faisant l'union de deux théories de Shostak on obtient une théorie de Shostak (c'est-à-dire les canoniseurs peuvent être combinés pour obtenir un canoniseur de l'union des théories et les solveurs peuvent être également combinés). Cette assertion est malheureusement fausse dans le cas général. En effet, il n'est pas toujours possible de combiner les solveurs pour obtenir un solveur pour l'union des théories [CK03b].

3.3 Combinaison non disjointe

Nous abordons dans cette section les résultats principaux de la combinaison non disjointe : celui de Ringeissen et Tinelli [RT03] et celui de Ghilardi [Ghi04]. Les deux méthodes diffèrent dans la façon de construire un modèle pour l'union de théories qui satisfait la formule mixte dont on cherche à décider la satisfiabilité.

La méthode de Ringeissen et Tinelli est fortement inspirée par les résultats du domaine de résolution de contraintes sur des domaines prédéfinis, très souvent des algèbres libres, où l'unification est utilisée pour résoudre les équationes. Ringeissen et Tinelli exploitent le fait que deux structures libres sur les bases de même cardinalité sont isomorphes pour identifier les hypothèses pour lesquelles le théorème de combinaison non disjointe (cf. Théorème 3.1) s'applique. Ils ont également introduit la notion de théorie stablement libre qui capture la notion de stable infinité dans le cas de la combinaison disjointe.

Ghilardi adopte une approche plus orientée théorie des modèles. Il s'appuie sur la notion de clôture existentielle de structures pour capturer la notion de stable infinité du cas disjoint. La construction d'un modèle pour l'union de théories et de la formule mixte dans la méthode de Ghilardi est basée sur le théorème de Robinson pour la Consistence Jointe [8]. Cette méthode fonctionne pour des théories qui sont compatibles par rapport à une sous-théorie admettant une modèle-complétion [9]. Un cas particulier de la compatibilité des théories composantes par rapport à une sous-théorie est le

8. Le théorème de Robinson sur la Consistence Jointe (en anglais "Robinson's Joint Consistency Theorem") [Hod94] stipule que l'union de deux théories consistantes partageant une théorie complète est consistante.

9. En fait, toutes les structures d'une modèle-complétion sont des clôtures existentielles.

suivant : les théories composantes sont des extensions conservatives d'une modèle-complétion d'une sous-théorie sur la signature commune. La notion de compatibilité se réduit à la notion de stable infinité dans le cas de la combinaison disjointe.

Il est intéressant de noter que dans le cas de la théorie de l'égalité, la notion de modèle libre sur une base infinie et celle de clôture existentielle de modèle coïncident. Cependant, ces deux notions sont très différentes en théorie des modèles car il existe des théories qui sont stablement libres mais sans admettre de modèle-complétion (voir e.g. [Ghi04] pour plus de détails).

3.3.1 Ringeissen et Tinelli

Ringeissen et Tinelli [RT03] ont travaillé dans le but de généraliser la méthode de Nelson-Oppen. En résumé, leur approche revient à identifier des conditions pour lesquelles le théorème de combinaison non disjointe s'applique : ces conditions permettent d'exhiber deux modèles composants dont les réductions à la signature commune sont isomorphes. Les résultats de cette approche sont fortement liés aux propriétés des structures libres sur leurs réductions et leurs produits amalgamés (ou leurs fusions d'après la terminologie de [RT03]).

Définition 3.10 (Identification de variables)**.** *Soit un ensemble de variables V. L'ensemble des identifications de V est défini par :*

$$ID(V) = \{\xi \in SUB(V) \| \mathcal{R}an(\xi) \subseteq V \setminus \mathcal{D}om(\xi)\}$$

où $SUB(V)$ désigne l'ensemble de substitutions idempotentes dont le domaine est un sous-ensemble de V. Pour chaque identification $\xi \in ID(V)$, on associe l'ensemble de contraintes :

$$\xi_{\neq} = \bigcup_{u,v \in V\xi, u \neq v} \{u \neq v\}$$

qui signifie que toutes les variables qui ne sont pas identifiées par ξ doivent prendre des valeurs deux à deux distinctes.

Définition 3.11 (Instanciation)**.** *Soient un ensemble de variables V et une signature finie $\Sigma = \bigcap_{i=1}^{n} \Sigma_i$. L'ensemble des Σ-instanciations de V est défini par :*

$$IN^{\Sigma}(V) = \{\rho \in SUB(V) \| Ran(\rho) \subseteq \mathcal{T}(\Sigma, X) \backslash V\}$$

où $X \cap V = \emptyset$. Pour chaque instanciation $\rho \in IN^{\Sigma}(V)$, on associe l'ensemble :

$$iso_{\rho}^{\Sigma} = \bigcup_{v \in Var(V\rho), f_i \in \Sigma_F} \{\forall \tilde{u}_i \; v \neq f_i(\tilde{u}_i)\}$$

où $\tilde{u}_i$ est un sous-ensemble des variables non-partagées et propres à T_i.

Soient $\phi_1,\ \phi_2$ deux conjonctions de Σ_i-littéraux purs pour $i = 1, 2$ et $\Sigma = \Sigma_1 \cap \Sigma_2$. La *semi-procédure de satisfiabilité* fonctionne en trois étapes :

Instanciation Générer le couple $< \gamma_1,\ \gamma_2 > = < \phi_1\rho \wedge iso_{\rho}^{\Sigma},\ \phi_2\rho \wedge iso_{\rho}^{\Sigma} >$ pour une certaine instanciation $\rho \in IN^{\Sigma}(V)$ où $V = (Var(\phi_1) \cap Var(\phi_2))$.

Identification Générer le couple $< \varphi_1,\ \varphi_2 > = < \gamma_1\xi \wedge \xi_{\neq},\ \gamma_2\xi \wedge \xi_{\neq} >$ pour une certaine identification $\xi \in ID(Var(V\rho))$.

Test de satisfiabilité Si φ_1, φ_1 sont satisfiables dans respectivement T_1 et T_2 alors retourner *satisfiable*.

Cette procédure n'est qu'une *semi-procédure* de satisfiabilité car la règle *Instanciation* nous dit que l'on essaie d'instancier de façon non-déterministe les variables partagées par des termes partagés. Cet ensemble de termes partagés est potentiellement infini. Si on arrive à trouver une instanciation et une identification de variables partagées qui rendent chaque Σ_i-formule T_i-satisfiable alors la conjonction de départ est *satisfiable*. Dans le cas contraire, la procédure ne termine pas. L'hypothèse faite sur les théories composantes est qu'elles doivent être *N-O-combinables* [RT03]. L'hypothèse de N-O-combinabilité garantit que la satisfiabilité dans l'union $T_1 \cup T_2$ d'une conjonction de formules pures $\phi_1 \wedge \phi_2$ peut se réduire à la satisfiabilité de ϕ_1 dans T_1 et de ϕ_2 dans T_2 en rajoutant des restrictions par rapport à la signature commune, i.e. des formules qui correspond à $\phi_i\rho \wedge iso_\rho^\Sigma$, pour $i = 1, 2$. Ringeissen et Tinelli ont ensuite proposé un catalogue de théories *N-O-combinables*, qui sont très souvent des théories pour lesquelles si une formule est satisfiable dans celles-ci alors elle est satisfiable dans un modèle libre sur une base de cardinalité infinie par rapport à la signature commune. Dans ce cas on parle de théories stablement libres. Dans le cas disjoint, la notion de théories stablement libres se réduit simplement aux théories stablement infinies.

3.3.2 Ghilardi

Dans [Ghi04], Ghilardi a généralisé la méthode de combinaison de Nelson-Oppen au cas non disjoint où les théories composantes partageant une sous-théorie ayant des propriétés étroitement liées à l'élimination des quantificateurs. La procédure de combinaison, qui consiste à propager des formules sans quantificateurs construites à partir de la signature commune, étend celle de Nelson-Oppen dans le sens où elle fait échanger des formules sans quantificateurs de signature commune au lieu d'échanger des disjonctions d'égalités entre variables partagées. La méthode de Ghilardi a été motivée par l'observation suivante.

Dans la méthode de combinaison de Nelson-Oppen, nous considérons une Σ_1-théorie T_1 et une Σ_2-théorie T_2 telles que Σ_1 et Σ_2 sont disjointes. Nous avons à décider la satisfiabilité de l'ensemble

$$(T_1 \cup \Gamma_1) \cup (T_2 \cup \Gamma_2)$$

où Γ_i est un ensemble de $\Sigma_i(\tilde{a})$-littéraux clos, $\Sigma_i(\tilde{a})$ étant Σ_i auquel on ajoute de nouvelles constantes libres partagées $\tilde{a}$ [10], pour $i = 1, 2$. Soit $\Sigma_0 = \Sigma_1 \cap \Sigma_2$ (qui se réduit à l'ensemble vide dans le cas disjoint). Si $(T_1 \cup \Gamma_1) \cup (T_2 \cup \Gamma_2)$ a un modèle $\mathcal{M}$, alors nous pouvons prendre le diagramme de Robinson $Diag(\mathcal{A})$ [11] de la Σ_0-sous-structure de $\mathcal{M}$ engendrée par $\tilde{a}$ et conclure que les deux ensembles

$$(T_1 \cup \Gamma_1 \cup Diag(\mathcal{A})) \text{ et } (T_2 \cup \Gamma_2 \cup Diag(\mathcal{A}))$$

10. On parle de variables partagées dans certain contexte. Si nous nous intéressons au problème de satisfiabilité des formules sans quantificateurs alors les variables libres sont existentiellement quantifiées et elles peuvent être donc considérées comme des constantes libres.

11. Le diagramme de Robinson d'une $\Sigma(\tilde{c})$-structure $\mathcal{A}$ est l'ensemble de tous les $\Sigma(\tilde{c})$-littéraux qui sont satisfiable dans $\mathcal{A}$.

sont également satisfiables. Et pour toute paire $a_i, a_j \in \tilde{a}$, nous avons $\mathcal{M} \models a_i = a_j$ ou $\mathcal{M} \models a_i \neq a_j$. Or il existe un nombre fini de tels arrangements de constantes. Si la satisfiabilité de $(T_1 \cup \Gamma_1) \cup (T_2 \cup \Gamma_2)$ est équivalente à la satisfiabilité locale de deux ensembles $(T_1 \cup \Gamma_1 \cup Diag(\mathcal{A}))$ et $(T_2 \cup \Gamma_2 \cup Diag(\mathcal{A}))$ alors nous avons une procédure de combinaison.

Malheureusement la satisfiabilité locale de $(T_1 \cup \Gamma_1 \cup Diag(\mathcal{A}))$ et $(T_2 \cup \Gamma_2 \cup Diag(\mathcal{A}))$ n'est pas une condition suffisante pour la satisfiabilité de $(T_1 \cup \Gamma_1) \cup (T_2 \cup \Gamma_2)$. Pour résoudre ce problème Nelson-Oppen ont supposé que T_1 et T_2 sont stablement infinies, ce qui signifie que tout modèle de T_i se plonge dans un modèle infini de T_i, pour $i = 1, 2$. Ghilardi avait donc noté que la théorie universelle T_0 (la théorie de l'égalité dans le cas disjoint) peut s'étendre à une théorie T_0^* (la théorie d'un ensemble infini dans le cas disjoint) sur la même signature Σ_0. En théorie des modèles, T_0^* est appelée une modèle-complétion de T_0 (T_i est stablement infinie signifie que tout modèle de T_i se plonge dans un modèle de $T_i \cup T_0^*$).

Définition 3.12 (Modèle-complétion)**.** *Soient T une Σ-théorie et T^* une Σ-théorie contenant T ($T^* \supseteq T$). On dit que T^* est une modèle-complétion de T si*

- *tout modèle de T peut être plongé dans un modèle de T^*, et*
- *pour tout modèle $\mathcal{M}$ de T, $T^* \cup Diag(\mathcal{M})$ est une théorie complète.*

En appliquant le théorème de Robinson pour la Consistence Jointe, l'hypothèse que tout modèle de T_i se plonge dans un modèle de $T_i \cup T_0^*$ est suffisante pour que la satisfiabilité de $(T_1 \cup \Gamma_1 \cup Diag(\mathcal{A}))$ et $(T_2 \cup \Gamma_2 \cup Diag(\mathcal{A}))$ implique celle de $(T_1 \cup \Gamma_1) \cup (T_2 \cup \Gamma_2)$. En fait, $T_0^* \cup Diag(\mathcal{A})$ est une théorie complète.

Pour garantir la terminaison de la méthode, Ghilardi a supposé également que la théorie T_0 est localement finie.

Définition 3.13 (Théorie localement finie)**.** *Une théorie universelle T_0 de signature Σ_0 est localement finie si et seulement si pour toute ensemble fini $\tilde{c}$ de constantes libres, on peut calculer un nombre fini de $\Sigma_0(\tilde{c})$-termes clos $t_1, \ldots, t_{n_{\tilde{c}}}$ tels que pour tout $\Sigma_0(\tilde{c})$-terme u nous avons $T_0 \models u = t_i$, pour un certain $i \in \{1, \ldots, n_{\tilde{c}}\}$.*

Intuitivement, T_0 est localement finie signifie qu'il existe un nombre fini de classes d'équivalences modulo T_0 sur les termes, ce qui nous permet d'avoir que $Diag(\mathcal{A})$ est fini.

Pour résumer, les hypothèses dont on a besoin pour la méthode de Ghilardi sont :

(i) il existe une Σ_0-théorie universelle T_0, qui est incluse à la fois dans T_1 et T_2 ;

(ii) T_0 admet une modèle-complétion T_0^* ;

(iii) tout modèle de T_i se plonge dans un modèle de $T_i \cup T_0^*$;

(iv) T_0 est localement finie.

Lorsqu'une théorie T_i satisfait les conditions (i), (ii) et (iii) on dit que T_i est T_0-compatible. Intuitivement, les conditions (i), (ii) et (iii) garantissent que la satisfiabilité locale à chaque théorie composante se transfère à l'union et la condition (iv) sert à assurer la terminaison de la procédure de combinaison.

L'approche de Ghilardi est intéressante non seulement parce qu'elle est très générale mais également parce que les hypothèses faites sur les théories composantes présentent une relation étroite avec l'élimination des quantificateurs comme démontré dans le résultat suivant.

Proposition 3.1 ([Hod94])**.** *Soient T une Σ-théorie et $T^* \supseteq T$ une autre Σ-théorie. Alors T^* est une modèle-complétion de T si*

- *tout modèle de T peut être plongé dans un modèle de T^*, et*
- *T^* admet l'élimination des quantificateurs.*

Récemment, Ghilardi, Nicolini et Zucchelli [GNZ05] ont généralisé [Ghi04] à un cadre très général de la logique d'ordre supérieur. Ils considèrent le problème de satisfiabilité dans des unions de fragments algébriques (au lieu de théories du premier ordre). D'après [GNZ05] un fragment algébrique consiste en : (i) un langage typé d'ordre supérieur ; (ii) un ensemble récursif de termes construits à partir du dernier langage ; et (iii) une classe de structures. Ainsi [GNZ05] a généralisé la notion de T_0-compatibilité et celle de T_0-finité locale à respectivement la compatibilité entre fragments algébriques et la *noethérianité* du fragment partagé par les fragments composants. De façon similaire au cas des théories du premier ordre, la compatibilité entre fragments garantit que la satisfiabilité locale à chaque fragment composant se transfère à l'union tandis que la noethérianité du fragment partagé sert à assurer la terminaison de la procédure de combinaison. Une méthode de combinaison est aussi proposée. Celle-ci est fortement inspirée de la méthode de Nelson-Oppen.

Chapitre 4

Combinaison des procédures de satisfiabilité dérivées par saturation

Les procédures de satisfiabilité pour les théories modélisant les structures de données telles que les listes, les tableaux se trouvent au coeur de beaucoup de systèmes de vérification, comme PVS [ORR+96], Simplify [DNS03], ICS [FORS01], CVC [SBD02], CVC Lite [BB04]. La conception, la preuve de correction et l'implantation de telles procédures présentent des défis qui sont loin d'être triviaux. Un des problèmes majeurs est de prouver la correction de ces procédures. De plus, l'implantation de chaque procédure se fait de façon *ad hoc*, avec peu de réutilisation de composants existants. Pour franchir ces obstacles, les auteurs de [ARR03] ont proposé une méthodologie uniforme basée sur la saturation pour construire des procédures de satisfiabilité et prouver leur correction. La preuve de correction se réduit à la preuve de la terminaison d'une application équitable et exhaustive des règles du Calcul de Superposition [NR01]. De plus, de telles procédures peuvent être implantées en utilisant les systèmes de preuve implantant le Calcul de Superposition [Sch02, Wei97, RV02]. Les efforts d'ingénierie et de validation de logiciels peuvent être ainsi réutilisés.

Malheureusement, la plupart des problèmes de vérification font intervenir différents domaines de calcul (ou théories) pour lesquels la méthodologie basée sur la saturation peut ne pas s'appliquer. Il y a donc un besoin indiscutable de construire une procédure de satisfiabilité pour l'union des théories en réutilisant et combinant les procédures de satisfiabilité existantes pour les théories composantes. C'est dans cet esprit qu'a été conçue la méthode de combinaison de Nelson-Oppen [NO79]. En adoptant cette méthode de combinaison, il est intéressant d'étudier comment y intégrer efficacement les procédures de satisfiabilité basées sur la saturation. Les problèmes à résoudre sont :

- comment produire directement des formules qui sont propagées d'une procédure à l'autre ;
- comment traiter efficacement de nouvelles formules qui proviennent d'autres procédures.

Le premier problème, qui est équivalent au problème de T-validité, peut être décidé en utilisant la procédure de T-satisfiabilité, i.e. pour vérifier qu'une formule ϕ est une conséquence d'un ensemble S de formules dans une théorie T il suffit de vérifier que

$\neg\phi$ est insatisfiable dans T. Mais cette solution n'est évidemment pas efficace, comme observé dans [DNS03]. Afin de franchir cet obstacle, nous montrons qu'une application exhaustive des règles du Calcul de Superposition, sous certaines hypothèses sur l'ordre utilisé, dérive suffisamment de formules partagées pour la théorie de l'égalité et la théorie des listes. Pour la théorie des tableaux nous montrons que —toujours sous certaines hypothèses sur l'ordre utilisé— une application exhaustive des règles d'une variante du Calcul de Superposition—obtenu en rajoutant au calcul une règle de "splitting"—dérive suffisamment de formules partagées.

Pour résoudre le deuxième problème, i.e. le traitement efficace de nouvelles formules, la solution naïve consistant à refaire la saturation sur le nouvel ensemble présente quelques inconvénients. D'un côté il peut être inutile de saturer tout le nouvel ensemble car certaines clauses ne seront pas utilisées et de l'autre côté il est difficile de faire un retour en arrière lorsque les nouvelles formules reçues causent l'insatisfiabilité. Nous proposons une architecture dans laquelle le Calcul de Superposition fonctionne comme un générateur de lemmes qui sont ensuite utilisés par la clôture de congruence. Les procédures de satisfiabilité construite de cette manière peuvent être combinées efficacement car celles-ci sont capables de traiter efficacement les égalités provenant d'autres procédures en utilisant la clôture de congruence. Ceci signifie qu'il n'y a pas besoin d'appeler à nouveau la saturation. Nous montrons que notre architecture est correcte pour les théories modélisant des structures de données, à savoir la théorie des listes et la théorie des tableaux.

Dans ce qui suit, nous commençons par un bref rappel de la méthodologie de construction de procédures de satisfiabilité par saturation. Nous abordons par la suite le problème de génération et de traitement efficace des formules partagées en considérant une par une la théorie de l'égalité, la théorie des listes et la théorie des tableaux.

4.1 Dériver des procédures de satisfiabilité par saturation

La dérivation des procédures de satisfiabilité par saturation fonctionne de la manière suivante. Considérons une théorie T axiomatisée par un ensemble fini $Ax(T)$ de clauses et un ensemble S qui ne contient que des littéraux plats et clos. Pour décider la satisfiabilité de S dans la théorie T, on applique exhaustivement les règles du Calcul de Superposition sur $Ax(T) \cup S$. Si on dérive la clause vide, alors S n'est pas satisfiable dans T, sinon S est satisfiable dans T. La correction de telles procédures est garantie par la correction des règles du Calcul de Superposition. Bien que le Calcul de Superposition soit réfutationnellement complet (ce qui veut dire que pour tout ensemble insatisfiable de clauses, un nombre fini d'applications des règles de ce calcul dérivera la clause vide), il ne termine pas en général. Ceci a pour conséquence que nos procédures décrites ci-dessous ne sont que des semi-procédures de satisfiabilité. Pour obtenir des procédures de satisfiabilité, nous devons montrer pour la théorie T la propriété suivante : pour tout ensemble S de littéraux plats et clos, toute application exhaustive des règles du Calcul de Superposition ne dérive qu'un ensemble fini de clauses.

4.1.1 Calcul de Superposition

Le Calcul de Superposition a été conçu dans le but de développer une version plus spécialisée et donc plus efficace de la Résolution de Robinson spécifiquement pour la logique du premier ordre avec égalité. En effet, pour traiter le prédicat d'égalité dans la logique du premier ordre, nous pouvons utiliser les axiomes de congruence, i.e. réflexivité, symétrie, transitivité et stabilité par contexte. Mais cette façon de procéder n'est bien évidemment pas efficace car la Résolution génère beaucoup trop de clauses non nécessaires. Pour surmonter ce problème, Robinson et Wos [RW69] ont proposé d'introduire une nouvelle règle de Paramodulation dédié au traitement implicite des axiomes de congruence. Les avantages et désavantages de ce nouveau calcul ont conduit à un volume important de travaux à la fois théoriques et pratiques dans la communauté autour de la déduction automatique basée sur la Paramodulation. Dans ce qui suit, nous allons brièvement survoler les idées théoriques et les fondements du Calcul de Superposition.

Une caractéristique fondamentale du Calcul de Superposition est l'utilisation d'un ordre de réduction $\succ$, qui est total sur l'ensemble de tous les termes clos. L'ordre $\succ$ est étendu aux littéraux de sorte que seules les instances maximales des littéraux sont considérées à chaque application d'une règle du calcul.

Nous utilisons le Calcul de Superposition utilisée dans [ARR03], enrichi par une règle Orientation (cf. Figures 4.1 et 4.2). Étant donné un ensemble S de clauses, une règle de clôture dans la figure 4.1 ajoute clauses dans sa conclusion à l'ensemble S tandis qu'une règle de simplification simplifie clauses dans S ou supprime clauses de S.

Définition 4.1. *Soit I un système d'inférence. On dit que*
- *une clause C est redondante par rapport à I (ou I-redondante) et par rapport à un ensemble S de clauses si C est dans S ou si S peut être dérivé de $S \cup \{C\}$ par une séquence d'applications des règles de I.*
- *une inférence de I est redondante par rapport à un ensemble S de clauses si sa conclusion est redondante par rapport à I et par rapport à S.*

Définition 4.2. *Un ensemble S de clauses est saturé par rapport à un système d'inférence I (ou I-saturé) si toutes les inférences de I avec prémisses dans S sont redondantes par rapport à S.*

Définition 4.3. *Une dérivation par rapport à un système d'inférence I (ou I-dérivation) est une séquence $S_0, S_1, \ldots, S_i, \ldots$ d'ensemble de clauses, où à chaque étape une inférence de I est appliquée pour générer et ajouter ou supprimer ou modifier une clause.*

Une I-dérivation est caractérisée par sa limite, définie comme l'ensemble de clauses persistantes

$$S_\infty = \bigcup_{j \geq 0} \bigcap_{i > j} S_i \,.$$

Une I-dérivation $S_0, S_1, ..., S_i, ...$ avec sa limite S_∞ est équitable par rapport à $\mathcal{SP}$ si pour toute inférence de I avec prémisses dans S_∞, il existe un $j \geq 0$ tel que cette inférence est redondante dans S_j.

La limite d'une I-dérivation équitable est aussi appelée une saturation par I.

Théorème 4.1 ([NR01])**.** *Si $S_0, S_1, \ldots$ est une $\mathcal{SP}$-dérivation équitable par rapport à $\mathcal{SP}$, alors*

(i) sa limite S_∞ est saturée par rapport à $\mathcal{SP}$, et

(ii) S_0 est insatisfiable si et seulement si la clause vide est dans S_j pour un certain j, et

(iii) si une telle $\mathcal{SP}$-dérivation équitable est finie, i.e. elle a la forme $S_0, \ldots, S_n$, alors S_n est saturé et équivalent à S_0.

On dit que $\mathcal{SP}$ est réfutationnellement complet car il est toujours possible de dériver la clause vide avec une $\mathcal{SP}$-dérivation finie à partir d'un ensemble insatisfiable de clauses (cf. (ii) du Théorème 4.1).

Superposition $-$ droite

$$\frac{\Gamma \Rightarrow \Delta, l[u'] = r \quad \Pi \Rightarrow \Sigma, u = t}{\sigma(\Gamma, \Pi \Rightarrow \Delta, \Sigma, l[t] = r)} \quad \textbf{si } (i), (ii), (iii), (iv)$$

Superposition $-$ gauche

$$\frac{\Gamma, l[u'] = r \Rightarrow \Delta \quad \Pi \Rightarrow \Sigma, u = t}{\sigma(l[t] = r, \Gamma, \Pi \Rightarrow \Delta, \Sigma)} \quad \textbf{si } (i), (ii), (iii), (iv)$$

Reflection

$$\frac{\Gamma, u' = u \Rightarrow \Delta}{\sigma(\Gamma \Rightarrow \Delta)} \quad \textbf{si } (v)$$

Factorisation

$$\frac{\Gamma \Rightarrow \Delta, u = t, u' = t'}{\sigma(\Gamma, t = t' \Rightarrow \Delta, u = t')} \quad \textbf{si } (i), (vi)$$

σ est l'unificateur principal de u et u', u' n'est pas une variable dans Superposition $-$ droite et Superposition $-$ gauche, L est un littéral, et les conditions suivantes sont satisfaites :

(i) $\sigma(u) \not\preceq \sigma(t)$,

(ii) $\forall L \in \Pi \cup \Sigma : \sigma(u = t) \not\preceq \sigma(L)$,

(iii) $\sigma(l[u']) \not\preceq \sigma(r)$,

(iv) $\forall L \in \Gamma \cup \Delta : \sigma(l[u'] = r) \not\preceq \sigma(L)$,

(v) $\forall L \in \Gamma \cup \Delta : \sigma(u' = u) \not\prec \sigma(L)$,

(vi) $\forall L \in \Gamma : \sigma(u) \not\preceq \sigma(L)$, $\forall L \in \{u' = t'\} \cup \Delta : \sigma(u = t) \not\prec \sigma(L)$.

FIGURE 4.1 – Règles de Clôture de $\mathcal{SP}$.

4.1.2 Dérivation uniforme de procédures de satisfiabilité par saturation

Étant donnée une théorie T axiomatisée par un ensemble fini $Ax(T)$ de clauses. La méthodologie basée sur la saturation consiste en les étapes suivantes :

Subsomption

$$\frac{S \cup \{C, C'\}}{S \cup \{C\}}$$

$$\textbf{si} \begin{cases} \exists \theta, \theta(C) \subseteq C' \\ \nexists \rho, \rho(C') \equiv C \end{cases}$$

Simplification

$$\frac{S \cup \{C[l'], l = r\}}{S \cup \{C[\theta(r)], l = r\}}$$

$$\textbf{si} \begin{cases} l' \equiv \theta(l) \\ \theta(l) \succ \theta(r) \\ \forall L \in C[\theta(l)] : L \succ (\theta(l) = \theta(r)) \end{cases}$$

Orientation

$$\frac{S \cup \{c = c'\}}{S[c \to c']} \quad \textbf{si } c \succ c'$$

Suppression

$$\frac{S \cup \{\Gamma \Rightarrow \Delta, t = t\}}{S}$$

où C et C' sont des clauses est S est un ensemble de clauses.

FIGURE 4.2 – Règles de Simplification de $\mathcal{SP}$.

(i) *L'aplatissement* consiste à transformer un ensemble de littéraux clos en un ensemble équisatisfiable qui ne contient que des littéraux plats et clos ;

(ii) *Le choix de l'ordre* se fait de manière à considérer un ordre de réduction, total sur l'ensemble de termes clos.

(iii) *La preuve de terminaison* consiste à assurer que toute saturation des axiomes de la théorie avec un ensemble arbitraire de littéraux plats et clos ne génère qu'un nombre fini de clauses.

En conséquence, si T est une théorie pour laquelle cette méthodologie s'applique, alors une procédure de T-satisfiabilité peut être construite en implantant l'aplatissement et en utilisant un prouveur automatisant le Calcul de Superposition avec un ordre convenable. Si l'ensemble final de clauses retourné par le prouveur contient la clause vide, alors la procédure de T-satisfiabilité retourne insatisfiable ; sinon, elle retourne satisfiable.

Dans ce qui suit, nous supposons que

Hypothèse 4.1. *Les ensembles de littéraux dont on cherche à décider la satisfiabilité dans une théorie ne contiennent que des littéraux plats et clos.*

En ce qui concerne l'ordre utilisé dans le Calcul de Superposition, nous supposons que :

Hypothèse 4.2. *$\succ$ vérifie les hypothèses suivantes :*
- *$\succ$ est un ordre de réduction total sur l'ensemble de termes clos, et*
- *$t \succ c$ pour chaque constante c et pour tout terme t qui est différent d'une constante et d'une variable, et*
- *$t \succ t'$ si t est un terme dont le symbole en tête est interprété et t est un terme dont le symbole en tête est non interprété et t, t' ne sont pas des constantes ou variables, et*
- *$\succ$ est étendu aux égalités en les considérant comme des multi-ensembles des termes, et puis aux clauses en les considérant comme des multi-ensembles d'égalités, en considérant l'extension multi-ensemble, et*
- *les diségalités soient plus grandes que les égalités.*

Par souci d'efficacité, nous supposons que

Hypothèse 4.3. *La règle* Orientation *est la plus prioritaire. Elle s'applique dès qu'une égalité entre constantes apparaît au cours de la saturation.*

Pour compléter la dérivation de procédures de satisfiabilité par saturation, nous devons montrer pour chaque théorie T la propriété suivante.

Définition 4.4 (Propriété de saturation finie)**.** *Soit $Ax(T)$ l'ensemble fini des axiomes de la théorie T. On dit que T a la propriété de saturation finie par rapport à $\mathcal{SP}$ si pour tout ensemble S de littéraux plats et clos, toute saturation de $Ax(T) \cup S$ ne génère qu'un nombre fini de clauses.*

Pour illustrer la dérivation de procédures de satisfiabilité par saturation, nous considérons la théorie de l'égalité, la théorie des listes, la théorie des tableaux et montrons que toute saturation des axiomes de ces théories avec un ensemble arbitraire de littéraux plats et clos ne génère qu'un nombre fini de clauses et ainsi $\mathcal{SP}$ fournit, pour ces théories, une procédure de satisfiabilité.

Nous commençons d'abord par la théorie $\mathcal{E}$ de l'égalité qui n'a aucun axiome.

Proposition 4.1. *Soit S un ensemble fini de littéraux plats et clos. Les clauses se trouvant dans la saturation de S par rapport à $\mathcal{SP}$ ont nécessairement une des formes suivantes :*

(i) *la clause vide ;*

(ii) $c \bowtie c'$ *;*

(iii) *égalité de la forme* $f(c_1, \ldots, c_n) = c$,

où $c, c_1, \ldots, c_n$ *sont des constantes et* $\bowtie \in \{=, \neq\}$

Démonstration. La preuve est faite par induction sur la longueur de la dérivation. Pour le cas de base, il est évident que S contient bien les clauses des formes mentionnées. Pour le cas d'induction, considérons l'application de différents types d'inférences :
- Superposition – droite : (ii) et (ii) donnent (ii) ; (ii) et (iii) donnent (iii) ; (iii) et (iii) donnent (ii).
- Superposition – gauche : (ii) et (ii) donnent (ii).

- Reflection : ne s'applique pas aux clauses unitaires.
- Factorisation : ne s'applique pas aux clauses unitaires.
- Orientation : ne modifie pas la forme des clauses.
- Subsomption : ne génère pas de nouvelles clauses.
- Simplification : idem que Superposition droite.
- Suppression : ne génère pas de nouvelles clauses.

□

Puisqu'il y a un nombre fini de constantes et de symboles de fonctions, le nombre de clauses générées dans une saturation doit être fini, et le résultat suivant est une conséquence directe de la Proposition 4.1.

Corollaire 4.1. *$\mathcal{SP}$ est une procédure de satisfiabilité pour la théorie $\mathcal{E}$ de l'égalité.*

Nous considérons maintenant une théorie équationnelle qui est la théorie $\mathcal{L}$ des listes [Sho84], axiomatisée par l'ensemble

$$Ax(\mathcal{L}) = \left\{ \begin{array}{l} car(cons(X,Y)) = X \\ cdr(cons(X,Y)) = Y \\ cons(car(X), cdr(X)) = X \end{array} \right\}$$

où X et Y sont des variables universelles.

Proposition 4.2. *Soit S un ensemble fini de littéraux plats et clos. Les clauses se trouvant dans la saturation de $Ax(\mathcal{L}) \cup S$ par rapport à $\mathcal{SP}$ ont nécessairement une des formes suivantes :*

(i) la clause vide ;

(ii) les axiomes dans $Ax(\mathcal{L})$, i.e. les clauses de la forme :
  (a) $car(cons(X,Y)) = X$;
  (b) $cdr(cons(X,Y)) = Y$;
  (c) $cons(car(X), cdr(X)) = X$;

(iii) littéraux plats et clos de la forme :
  (a) $c \bowtie c'$;
  (b) $f(c_1, \ldots, c_n) = c$;
  (c) $car(a) = b$;
  (d) $cdr(a) = b$;
  (e) $cons(a,b) = c$;

(iv) égalités de la forme :
  (a) $cons(b, cdr(a)) = a$;
  (b) $cons(car(a), b) = a$;

où X, Y sont des variables, $a, b, c, c_1, \ldots, c_n$ sont des constantes, f est une fonction non interprétée et $\bowtie \in \{=, \neq\}$.

Démonstration. La preuve est faite par induction sur la longueur de la dérivation. Pour le cas de base, il est évident que $Ax(\mathcal{L}) \cup S$ contient bien les clauses des formes mentionnées. Pour le cas d'induction, nous considérons l'application de différents types d'inférences comme pour la théorie de l'égalité. □

Corollaire 4.2. *$\mathcal{SP}$ est une procédure de satisfiabilité pour la théorie $\mathcal{L}$ des listes.*

Un exemple de théorie axiomatisée par un ensemble de clauses non unitaires est la théorie $\mathcal{A}$ des tableaux axiomatisée par l'ensemble

$$Ax(\mathcal{A}) = \left\{ \begin{array}{l} select(store(A, I, E), I) = E \\ I = J \vee select(store(A, I, E), J) = select(A, J) \end{array} \right\}$$

où A, I, J, E sont des variables universelles.

Proposition 4.3. *Soit S un ensemble fini de littéraux plats et clos. Les clauses se trouvant dans la saturation de $Ax(\mathcal{A}) \cup S$ par rapport à $\mathcal{SP}$ ont nécessairement une des formes suivantes :*

(i) la clause vide ;

(ii) les axiomes dans $Ax(\mathcal{A})$, i.e. les clauses de la forme :

- *(a) $select(store(A, I, E), I) = E$;*
- *(b) $I = J \vee select(store(A, I, E), J) = select(A, J)$;*

(iii) littéraux plats et clos de la forme :

- *(a) $c_1 \bowtie c_2$;*
- *(b) $f(c_1, \ldots, c_n) = c$;*
- *(c) $store(a, i, e) = a'$;*
- *(d) $select(a, i) = e$;*

(iv) clauses non unitaires de la forme :

- *(a) $c_1 = c'_1 \vee \ldots \vee c_n = c'_n$;*
- *(b) $c \neq c' \vee c_1 = c'_1 \vee \ldots \vee c_n = c'_n$;*
- *(c) $f(c_1, \ldots, c_n) = c \vee i_1 = i'_1 \vee \ldots \vee i_n = i'_n$;*
- *(d) $select(a, X) = select(a', X) \vee X = i''_1 \vee \ldots \vee X = i''_n \vee i_1 = i'_1 \vee \ldots \vee i_m = i'_m$;*
- *(e) $select(a, i) = e \vee i = i' \vee i_1 = i'_1 \vee \ldots \vee i_n = i'_n$;*
- *(f) $store(a, i, e) = a' \vee i_1 = i'_1 \vee \ldots \vee i_n = i'_n$;*

où X, A, I, J, E sont des variables, $a, a', e, i, i' i''_1, \ldots, i''_n, i_1 i'_1, \ldots, i_n, i'_n, c, c', c_1, c'_1, \ldots, c_n, c'_n$ sont des constantes, f est un symbole de fonction non interprété et $\bowtie \in \{=, \neq\}$.

Démonstration. La preuve est faite par induction sur la longueur de la dérivation. Pour le cas de base, il est évident que $Ax(\mathcal{A}) \cup S$ contient bien les clauses des formes mentionnées. Pour le cas d'induction, nous considérons l'application de différents types d'inférences comme pour la théorie de l'égalité et des listes. Une différence avec ces deux cas est que nous avons des clauses non unitaires dans la dérivation. Par conséquent nous utilisons le fait que l'ordre utilisé par le Calcul de Superposition "privilégie" toujours les littéraux négatifs par rapport aux littéraux positifs. □

Corollaire 4.3. *$\mathcal{SP}$ est une procédure de satisfiabilité pour la théorie $\mathcal{A}$ des tableaux.*

4.2 Générer efficacement des formules partagées

L'idée principale de la méthode de combinaison de Nelson-Oppen est de propager les égalités entre constantes déduites d'une procédure de satisfiabilité à l'autre. Comme expliqué précédemment, il n'y a aucune difficulté à implanter la déduction des égalités entre constantes en utilisant la procédure de satisfiabilité elle-même. Cependant pour avoir une implantation efficace de la méthode de Nelson et Oppen, il est important, comme mentionné dans [DNS03], d'avoir des procédures de satisfiabilité ayant la capacité de produire directement les égalités à échanger lorsque celles-ci terminent avec le résultat satisfiable. Dans ce cas, nous dirons que telles procédures sont *complètes vis-à-vis de la déduction.*

4.2.1 Complétude vis-à-vis de la déduction

Nous allons dans un premier temps définir formellement la notion de complétude vis-à-vis de la déduction.

Définition 4.5 (Clause élémentaire)**.** *Une clause est élémentaire si elle est une disjonction d'égalité entre constantes. Une égalité élémentaire est une clause élémentaire unitaire.*

Définition 4.6 (Procédure de satisfiabilité complète vis-à-vis de la déduction)**.** *Une procédure de satisfiabilité pour une théorie T est complète vis-à-vis de la déduction par rapport aux clauses élémentaires si pour tout ensemble T-satisfiable S de littéraux plats et clos, elle retourne, à part* satisfiable*, un ensemble de clauses élémentaires S_e tel que pour toute clause élémentaire C, nous avons $T \models S \Rightarrow C$ si et seulement si $S_e \models C$.*

Pour montrer que le Calcul de Superposition est une procédure de satisfiabilité complète vis-à-vis de la déduction pour une théorie T nous devons prouver la propriété suivante.

Propriété 4.1. *Soit T une théorie axiomatisée par un ensemble fini $Ax(T)$ de clauses. Alors*

- *pour tout ensemble T-satisfiable S de littéraux plats et clos, et*
- *pour toute clause élémentaire C,*

nous avons $T \models S \Rightarrow C$ si et seulement si $S_e \models C$, où S_e est le sous-ensemble de la saturation de $Ax(T) \cup S$ par $\mathcal{SP}$, qui contient toutes les clauses élémentaires.

Comme la propriété doit être prouvée pour chaque théorie T, nous étendons la méthodologie par saturation (voir Section 4.1) avec l'étape suivante :

(iv) *complétude vis-à-vis de la déduction* : tout ensemble de clauses saturé par $\mathcal{SP}$ (qui ne contient pas la clause vide) contient un sous-ensemble d'égalités (resp. de clauses) élémentaires, qui implique toutes les égalités (resp. clauses) élémentaires qui sont une conséquence de l'ensemble initial de clauses.

Théorie de l'égalité Nous commençons par montrer que $\mathcal{SP}$ est une procédure de satisfiabilité complète vis-à-vis de la déduction pour la théorie de l'égalité.

Proposition 4.4. *Soient S un ensemble satisfiable de littéraux plats et clos et S' une saturation de S par $\mathcal{SP}$. Soit $c = c'$ une égalité élémentaire, alors les propriétés suivantes sont équivalentes :*

(i) $S \models c = c'$,

(ii) $S_e \models c = c'$, *où* S_e *est le sous-ensemble de* S', *qui contient toutes les égalités élémentaires.*

Démonstration. (ii) $\Rightarrow$ (i) est trivial. Nous allons prouver (i) $\Rightarrow$ (ii). Puisque S et S' sont équivalents, $S \models c = c'$ si et seulement si $S' \models c = c'$. Ce qui veut dire aussi que $S' \cup \{c \neq c'\}$ est insatisfiable. Par la complétude réfutationnelle du Calcul de Superposition, on peut dériver la clause vide à partir de $S' \cup \{c \neq c'\}$ utilisant le système d'inférence $\mathcal{SP}$. Or S' étant déjà saturé et ne contenant pas de clause vide, les inférences qui permettent de dériver la clause vide doivent avoir comme prémisses des clauses dans S' et $c \neq c'$ ou des clauses dérivées à partir de S' et $c \neq c'$. Considérons maintenant les types d'inférences avec de telles prémisses. Soit C une clause dans S'. S'il y a une inférence utilisant C et $c \neq c'$, alors C doit être une égalité entre constantes ou variables. Or d'après les formes listées dans la Proposition 4.1, C ne doit pas contenir de variable. Par conséquent, C est une égalité élémentaire et ainsi la conclusion d'une telle inférence est une diségalité entre constantes. Ce qui veut dire que le sous-ensemble S_e et $c \neq c'$ suffisent pour dériver la clause vide. Ou de façon équivalente, $S_e \models c = c'$. □

La Proposition 4.4 nous permet de conclure le résultat suivant.

Corollaire 4.4. *$\mathcal{SP}$ est une procédure de satisfiabilité complète vis-à-vis de la déduction pour la théorie de l'égalité.*

Théorie des listes Considérons maintenant la théorie des listes.

Proposition 4.5. *Soient S un ensemble $\mathcal{L}$-satisfiable de littéraux plats et clos et S' une saturation de $Ax(\mathcal{L}) \cup S$ par $\mathcal{SP}$. Soit $c = c'$ une égalité élémentaire, alors les propriétés suivantes sont équivalentes :*

(i) $T \models S \Rightarrow c = c'$,

(ii) $S_e \models c = c'$, *où* S_e *est le sous-ensemble de* S', *qui contient toutes les égalités élémentaires.*

Démonstration. Nous prouvons juste (i)$\Rightarrow$ (ii) car (ii)$\Rightarrow$ (i) est trivial. De façon similaire à la théorie de l'égalité. S'il y a une inférence utilisant une C dans S' et $c \neq c'$, alors C doit être une égalité entre constantes ou variables. Or au vu des formes listées dans la Proposition 4.2, C ne peut pas contenir de variable. Ce qui implique que C est une égalité élémentaire. Nous pouvons donc conclure $S_e \models c = c'$, comme pour le cas de la théorie de l'égalité. □

La Proposition 4.5 nous permet de conclure le résultat suivant.

Corollaire 4.5. *$\mathcal{SP}$ est une procédure de satisfiabilité complètes vis-à-vis de la déduction pour la théorie des listes.*

4.2.2 Étendre la méthode aux théories non équationelles

Pour la théorie des tableaux, le résultat sur la complétude vis-à-vis la déduction ne peut plus s'appliquer. En effet, puisque nous pouvons avoir des clauses non unitaires qui contiennent plus d'une égalité dans une dérivation, certaines inférences sont bloquées en raison de l'ordre utilisé par le Calcul de Superposition. Ce qui implique que la saturation peut ne pas contenir une clause élémentaire qui est une conséquence de l'ensemble de clauses en entrée. Considérons l'exemple suivant.

Exemple 4.1. *Considérons l'ensemble*

$$S = \left\{ \begin{array}{l} select(a', i') = e' \\ store(a, i, e) = a' \\ select(a, i') = e' \\ i \neq i' \end{array} \right\}$$

La saturation de

$$Ax(\mathcal{A}) \cup S = \left\{ \begin{array}{l} select(store(A, I, E), I) = E \\ I = J \vee select(store(A, I, E), J) = select(A, J) \\ select(a', i') = e' \\ store(a, i, e) = a' \\ select(a, i') = e' \\ i \neq i' \end{array} \right\}$$

donne

$$S' = \left\{ \begin{array}{l} select(store(A, I, E), I) = E \\ I = J \vee select(store(A, I, E), J) = select(A, J) \\ select(a', i') = e' \\ store(a, i, e) = a' \\ select(a, i') = e' \\ i \neq i' \\ select(a, i') = e' \\ select(a, J) = select(a', J) \vee J = i \\ select(a, i') = e \vee i = i' \\ e' = e \vee i = i' \end{array} \right\}$$

Il est facile de voir que $\{i \neq i', e' = e \vee i = i'\} \models e' = e$*, d'où nous avons* $S' \models e' = e$*. Mais le sous-ensemble* $\{e' = e \vee i = i'\}$ *des clauses élémentaires de* S' *n'implique pas* $e' = e$*, si on considère l'ordre* $\succ$ *tel que* $e' \succ e \succ i \succ i'$*. Par conséquent* $\mathcal{SP}$ *n'est pas une procédure de satisfiabilité complète vis-à-vis de la déduction pour la théorie des tableaux.*

Pour franchir cet obstacle, nous proposons une solution inspirée de la méthode d'analyse de cas (en anglais "splitting") décrite dans [RV01]. L'idée de l'analyse de cas est de ramener la réfutation d'un ensemble de clauses $S \cup \{A \vee B\}$ à la réfutation de l'ensemble $S \cup \{A \vee p, \neg p \vee B\}$. En procédant ainsi on dit que $S \cup \{A \vee p, \neg p \vee B\}$ est obtenu à partir de $S \cup \{A \vee B\}$ par fragmentation binaire. Dans notre contexte, l'analyse explicite de cas nous permet de fragmenter des clauses en introduisant de

nouvelles constantes propositionnelles qui seront considérées comme les plus petits éléments par rapport à l'ordre $\succ$ utilisé dans le Calcul de Superposition. Ainsi nous transformons des clauses qui peuvent contenir plusieurs égalités en des clauses qui ne contiennent qu'une seule égalité et par conséquent obtenons la complétude vis-à-vis de la déduction.

Nous allons avoir besoin de définir un ordre $\succ$ avec les caractéristiques suivantes.

Hypothèse 4.4. *$\succ$ a toutes les caractéristiques de l'ordre utilisé dans $\mathcal{SP}$ (voir Hypothèse 4.2) et satisfait de plus les restrictions suivantes :*

- *$\succ$ est total sur l'ensemble des constantes propositionnelles, et*
- *p et $\neg p$ sont plus petites que les égalités pour toute constante propositionnelle p, et*
- *$\neg p \succ p$ pour toute constante propositionnelle p.*

Soit $\mathcal{SP}'$ le système d'inférence obtenu en ajoutant à $\mathcal{SP}$ (paramétré par $\succ$) les deux règles d'inférence suivantes :

Fragmentation

$$\frac{S' \cup \{A \vee B\}}{S' \cup \{A \vee p, \neg p \vee B\}} \quad \textbf{si } Vars(A) \cap Vars(B) = \emptyset$$

Resolution

$$\frac{A \vee p \quad \neg p \vee B}{A \vee B} \quad \textbf{si} \begin{cases} \forall L \in A : p \not\preceq L \\ \forall L \in B : \neg p \not\preceq L \end{cases}$$

Il est important de noter que $\mathcal{SP}'$ préserve la complétude réfutationnelle car celui-ci n'est rien d'autre que la Résolution enrichie par la superposition pour traiter l'égalité et la fragmentation pour faire l'analyse de cas.

Théorème 4.2. *$\mathcal{SP}'$ est réfutationellement complet, i.e. il est toujours possible de dériver la clause vide avec une $\mathcal{SP}'$-dérivation finie à partir d'un ensemble insatisfiable de clauses construites à partir de la signature composée du symbole d'égalité et de constantes propositionnelles.*

Aussi, les notions de saturation, de propriété de saturation finie peuvent facilement s'étendre à $\mathcal{SP}'$. Dans ce cas, on dit qu'une théorie a la propriété de saturation finie par rapport à $\mathcal{SP}'$.

Proposition 4.6. *Pour toute $\mathcal{SP}'$-dérivation*

$$S_0, S_1, \ldots, S_i, S_{i+1}, \ldots$$

telle que S_0 est un ensemble de clauses, nous avons :

(i) l'ensemble S_i est satisfiable si et seulement si S_{i+1} l'est aussi ;

(ii) si C est une clause qui ne contient pas de constante propositionnelle qui apparaît dans S_i, alors $S_i \cup \{C\}$ est satisfiable si et seulement si $S_{i+1} \cup \{C\}$ l'est aussi.

Démonstration. (i). Il est facile de voir que $S \cup \{A \vee p, \neg p \vee B\}$ et $S \cup \{A \vee B\}$ sont équisatisfiables car tout modèle de $S \cup \{A \vee p, \neg p \vee B\}$ est aussi un modèle de $S \cup \{A \vee B\}$ et tout modèle de $S \cup \{A \vee B\}$ peut être étendu pour qu'il devienne un modèle de $S \cup \{A \vee p, \neg p \vee B\}$.
(ii). Tout modèle de $S \cup \{A \vee p, \neg p \vee B\} \cup \{C\}$ est un modèle de $S \cup \{A \vee B\} \cup \{C\}$. Puisque C ne contient pas de constante propositionnelle, tout modèle $\mathcal{M}$ de $S \cup \{A \vee B\} \cup \{C\}$ peut être étendu à un modèle $\mathcal{M}'$ de $S \cup \{A \vee p, \neg p \vee B\} \cup \{C\}$ en définissant ce dernier de telle sorte que $\mathcal{M}' \models p \leftrightarrow \neg\forall\tilde{x}A$. □

Comme nous avons étendu la signature de départ avec les constantes propositionnelles, il est nécessaire d'adapter la notion de clause élémentaire pour prendre en compte cette extension.

Définition 4.7 (Clause quasi-élémentaire)**.** *Une clause est quasi-élémentaire si elle a une des formes suivantes :*
- $[\neg]p_1 \vee \ldots \vee [\neg]p_n$ *;*
- $c = c' \vee [\neg]p_1 \vee \ldots \vee [\neg]p_n$ *;*

où c, c' sont des constantes, $p_1, \ldots, p_n$ sont des constantes propositionnelles, $n \geq 0$ et $[\neg]$ signifie que $\neg$ est optionnel.

Puisque la saturation d'un ensemble de clauses par $\mathcal{SP}'$ peut contenir des clauses quasi-élémentaires, pour rendre $\mathcal{SP}'$ complète vis-à-vis de la déduction, nous avons besoin de faire un post-traitement, qu'on appelle *defragmentation*, décrit par le système d'inférence $Defrag$, formé de la seule règle d'inférence suivante :

Defragmentation

$$\frac{A \vee p \quad \neg p \vee B}{A \vee B} \text{ **si** } A \vee p, \neg p \vee B \text{ sont quasi-élémentaires}$$

Il est facile de voir que $Defrag$ préserve la satisfiabilité.

Définition 4.8. *Si I est un système d'inférence et S est un ensemble fini de clauses, on désigne par $I(S)$ la limite d'une I-dérivation à partir de S. Si I' est un système d'inférence et $I(S)$ est fini, alors $I'(I(S))$ désigne la limite d'une I'-dérivation à partir de $I(S)$.*

Lemme 4.1. *Soient S_{qe} un ensemble de clauses quasi-élémentaires qui est $\mathcal{SP}'$-saturé et S_e le sous-ensemble de la $Defrag$-saturation de S_{qe}, qui contient toutes les clauses élémentaires. Alors, pour toute clause élémentaire C nous avons $S_{qe} \models C$ si et seulement si $S_e \models C$.*

Démonstration. La condition suffisante est triviale. Pour prouver la condition nécessaire, soit $C \equiv c_1 = c'_1 \vee \ldots c_n = c'_n$. Raisonnons par réfutation, $S_{qe} \models c_1 = c'_1 \vee \ldots c_n = c'_n$ si et seulement si $S_{qe} \cup \{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$ est insatisfiable. Et donc il est possible de dériver la clause vide à partir de $S_{qe} \cup \{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$ par $\mathcal{SP}'$. Nous considérons deux cas de figure : (i) la clause vide est dérivée en utilisant seulement les règles de $\mathcal{SP}$, et (ii) la clause vide est dérivée en utilisant les règles de $\mathcal{SP}'$.

(i) Si la clause vide est dérivée en utilisant seulement les règles de $\mathcal{SP}$, alors les clauses utilisées ne doivent pas contenir les constantes propositionnelles. Ceci signifie que le sous-ensemble S_e de S_{qe} contenant toutes les clauses élémentaires, conjointement avec $\{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$ suffit pour dériver la clause vide. Ou de façon équivalente $S_e \models c_1 = c'_1 \vee \ldots c_n = c'_n$.

(ii) Dans ce cas, la clause vide est dérivée par une suite d'inférences terminant par Resolution. Ceci est due au fait que les constantes propositionnelles sont les plus petites par rapport à l'ordre utilisé (Hypothèse 4.4) et par conséquent avant de dériver la clause vide il est nécessaire de dériver des clauses contenant seulement des constantes propositionnelles pour lesquelles Resolution s'applique.
Soit $\mathcal{SP}'(S_{qe} \cup \{c_1 \neq c'_1, \ldots, c_n \neq c'_n\})$ la saturation de $S_{qe} \cup \{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$ par $\mathcal{SP}'$. Puisque les constantes propositionnelles ne sont qu'introduites par Fragmentation, si la clause vide est dérivé par $\mathcal{SP}'$ à partir de $S_{qe} \cup \{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$ alors nécessairement la dérivation de la clause vide termine par une Superposition − gauche suivie par une Resolution. Les clauses utilisées par ces deux inférences doivent avoir les formes $A \vee P, \neg A, \neg P$, où A est une égalité entre constantes (ou symétriquement une diségalité) et P est une constantes propositionnelle. Supposons que A est une égalité entre constantes (le cas où A est une diségalité peut être traité de façon similaire). Dans ce cas A et $\neg A$ sont dans

$$Defrag(\mathcal{SP}'(S_{qe} \cup \{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}))\ ^{12}.$$

Nous allons prouver que A et $\neg A$ sont dans $\mathcal{SP}'(S_e \cup \{c_1 \neq c'_1, \ldots, c_n \neq c'_n\})$. En effet, puisque S_{qe} est déjà $\mathcal{SP}'$-saturé, $\mathcal{SP}'$ ne dérive pas de nouvelle clause quasi-élémentaire de la forme $c = c' \vee [\neg]p_1 \vee \ldots \vee [\neg]p_n$ à partir de $S_{qe} \cup \{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$. Ceci implique que le sous-ensemble de $Defrag(\mathcal{SP}'(S_{qe} \cup \{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}))$ contenant toutes les clauses élémentaires est égal à S_e. Ce qui veut dire aussi que A est dans S_e. Pour $\neg A$, il est facile de voir que $\mathcal{SP}'$ ne dérive pas de nouvelle diségalité à partir de $(S_{qe} \setminus S_e) \cup \{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$. Donc $\neg A$ ne peut être dérivé qu'à partir de $S_e \cup \{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$.
Nous avons donc que A et $\neg A$ sont dans $\mathcal{SP}'(S_e \cup \{c_1 \neq c'_1, \ldots, c_n \neq c'_n\})$, qui doit contenir la clause vide. De façon équivalente, $S_e \models c_1 = c'_1 \vee \ldots c_n = c'_n$.

□

Théorie des tableaux En réadaptant la preuve de la Proposition 4.3, nous pouvons montrer que la saturation d'un ensemble de littéraux plats et clos avec les axiomes de la théorie par $\mathcal{SP}'$ ne contient que des clauses de certaines formes.

Proposition 4.7. *Soit S un ensemble fini de littéraux plats et clos. Les clauses se trouvant dans la saturation de $Ax(\mathcal{A}) \cup S$ par rapport à $\mathcal{SP}'$ ont seulement une des formes suivantes :*

(i) la clause vide ;

(ii) les axiomes dans $Ax(\mathcal{A})$, i.e. les clauses de la forme :

(a) $select(store(A, I, E), I) = E$;

(b) $I = J \vee select(store(A, I, E), J) = select(A, J)$;

12. Par la Définition 4.8 $Defrag(\mathcal{SP}'(S))$ signifie la $Defrag$-saturation de la $\mathcal{SP}'$-saturation de S.

(iii) littéraux plats et clos de la forme :

(a) $c_1 \bowtie c_2$ *;*

(b) $f(c_1, \ldots, c_n) = c$ *;*

(c) $store(a, i, e) = a'$ *;*

(d) $select(a, i) = e$ *;*

(iv) clauses non unitaires de la forme :

(a) $c \bowtie c' \vee [\neg]p_1 \vee \ldots \vee [\neg]p_n$ *;*

(b) $f(c_1, \ldots, c_n) = c \vee [\neg]p_1 \vee \ldots \vee [\neg]p_n$ *;*

(c) $select(a, i) = e \vee [\neg]p_1 \vee \ldots \vee [\neg]p_n$ *;*

(d) $store(a, i, e) = a' \vee [\neg]p_1 \vee \ldots \vee [\neg]p_n$ *;*

(e) $select(a, X) = select(a', X) \vee X = i''_1 \vee \ldots \vee X = i''_n \vee [\neg]p_1 \vee \ldots \vee [\neg]p_n$ *;*

(f) $[\neg]p_1 \vee \ldots \vee [\neg]p_n$ *;*

où X, A, I, J, E *sont des variables,* $a, a', e, i, i''_1, \ldots, i''_n, c, c', c_1, \ldots, c_n$ *sont des constantes,* $p_1, \ldots p_n$ *sont des constantes propositionnelles,* f *est une fonction non interprétée,* $\bowtie \in \{=, \neq\}$*,* $n > 0$ *et* $[\neg]$ *signifie que* $\neg$ *est optionnel.*

La Proposition 4.7 nous permet de prouver le résultat suivant.

Proposition 4.8. *Soient* S *un ensemble* $\mathcal{A}$*-satisfiable de littéraux plats et clos et* S_{qe} *le sous-ensemble de la saturation de* $Ax(\mathcal{A}) \cup S$ *par* $\mathcal{SP}'$*, qui contient toutes les clauses quasi-élémentaires. Alors, pour toute clause élémentaire* C *nous avons* $Ax(\mathcal{A}) \cup S \models C$ *si et seulement si* $S_{qe} \models C$*.*

Démonstration. Soit S' la saturation de $Ax(\mathcal{A}) \cup S$ par $\mathcal{SP}'$. Et soit $C \equiv c_1 = c'_1 \vee \ldots c_n = c'_n$ une clause élémentaire. Raisonnons par réfutation, $Ax(\mathcal{A}) \cup S \models c_1 = c'_1 \vee \ldots c_n = c'_n$ si et seulement si $Ax(\mathcal{A}) \cup S \cup \{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$ est insatisfiable. Il est donc possible de dériver la clause vide à partir de $Ax(\mathcal{A}) \cup S \cup \{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$ par $\mathcal{SP}'$. Il résulte de la Proposition 4.6 et la complétude réfutationnelle de $\mathcal{SP}'$ qu'il est aussi possible dériver la clause vide à partir de $S' \cup \{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$ par $\mathcal{SP}'$. Puisque S est $\mathcal{A}$-satisfiable, la clause vide ne peut être dérivée qu'en commençant par une inférence utilisant une clause C' dans S' et une clause dans $\{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$. Analysons une telle inférence, i.e. entre C' et $c_k \neq c'_k$. Si une telle inférence a lieu, C' doit être nécessairement une clause contenant soit des égalités entre constantes ou variables soit des (négations de) constantes propositionnelles soit les deux en même temps, autrement l'inférence n'est pas possible. Or, au vu des formes de clauses listées dans la Proposition 4.7, C' ne peut être qu'une clause quasi-élémentaire. Maintenant, la conclusion de cette inférence doit contenir au plus une diségalité et éventuellement des (négations des) constantes propositionnelles. Et encore une fois, pour avoir une inférence avec ce type de clause, il faut utiliser une autre clause quasi-élémentaire. Cela signifie que le sous-ensemble S_{qe} de S', qui contient toutes les clauses quasi-élémentaires, et $\{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$ suffisent pour dériver la clause vide. Ou de façon équivalente, $S_{qe} \models c_1 = c'_1 \vee \ldots c_n = c'_n$. □

La Proposition 4.8 et le Lemme 4.1 nous permettent d'obtenir le résultat suivant concernant la complétude vis-à-vis de la déduction pour la théorie des tableaux.

Corollaire 4.6. $\mathcal{SP}'$ *est une procédure de satisfiabilité complète vis-à-vis de la déduction pour la théorie des tableaux.*

4.3 Traiter efficacement de nouvelles formules partagées

Nous avons vu précédemment que le Calcul de Superposition (avec ou sans fragmentation) est une procédure de satisfiabilité complète vis-à-vis de la déduction pour une large classe de théories. Ceci permet d'améliorer la combinaison à la Nelson-Oppen car les formules partagées à propager sont produites directement dans la saturation. Dans cette partie, nous étudions le problème du traitement efficace des formules provenant d'autres procédures. A première vue, une solution naïve consiste à saturer à nouveau l'ensemble saturé avec les nouvelles formules reçues. Mais elle présente plusieurs inconvénients : d'un côté il peut être inutile de saturer tout le nouvel ensemble car certaines clauses ne seront pas utilisées et de l'autre côté il est difficile de faire un retour en arrière lorsque les nouvelles formules reçues causent l'insatisfiabilité.

Nous avons constaté que pour décider la satisfiabilité d'un ensemble $Ax(T) \cup S$ pour une théorie équationnelle T et tout ensemble S de littéraux plats et clos, le Calcul de Superposition se comporte comme la clôture de congruence enrichie avec la superposition entre les littéraux dans S et les axiomes dans $Ax(T)$. Or les formules partagées à considérer ne sont que des clauses élémentaires. Ce qui signifie que la propagation des clauses élémentaires déduites et le traitement de nouvelles clauses élémentaires peuvent être éventuellement délégués à la clôture de congruence. Cette observation nous a conduit à considérer une architecture dans laquelle le Calcul de Superposition fonctionne comme un générateur de lemmes qui sont ensuite utilisés par la clôture de congruence. Des procédures de satisfiabilité construites de cette façon peuvent être combinées efficacement car elles sont non seulement complètes vis-à-vis de la déduction mais aussi capables de traiter efficacement les égalités provenant d'autres procédures sans avoir à faire appel à la saturation. La combinaison se fait au niveau de la Clôture de Congruence.

Notre architecture fonctionne de façon suivante :

(i) Saturer l'ensemble $Ax(T) \cup S$ pour obtenir un ensemble S'.

(ii) Extraire le sous-ensemble S_g de S' qui ne contient que des clauses closes.

(iii) Envoyer S_g à la clôture de congruence pour décider sa satisfiabilité.

(iv) La propagation des égalités élémentaires se fait au niveau de la clôture de congruence.

Pour prouver la correction de notre méthode, nous devons prouver, pour une théorie T, la propriété suivante.

Propriété 4.2. *Soit T une théorie axiomatisée par un ensemble fini $Ax(T)$ d'égalités. Soient S un ensemble T-satisfiable de littéraux plats et clos et S' une saturation de $Ax(T) \cup S$ par $\mathcal{SP}$. Si $c = c'$ est une égalité élémentaire, alors les propriétés suivantes sont équivalentes :*

(i) $T \cup S \cup \{c = c'\}$ est satisfiable,

(ii) $S_g \cup \{c = c'\}$ est satisfiable, où S_g est le sous-ensemble de S', qui contient tous les clauses closes.

Pour illustrer notre méthode, nous considérons dans un premier temps la théorie des listes.

Proposition 4.9. *Soient S un ensemble $\mathcal{L}$-satisfiable de littéraux plats et clos et S' une saturation de $Ax(\mathcal{L}) \cup S$ par $\mathcal{SP}$. Soit $c = c'$ une égalité élémentaire, alors les propriétés suivantes sont équivalentes :*

(i) $S \cup \{c = c'\}$ *est* $\mathcal{L}$*-satisfiable,*

(ii) $S_g \cup \{c = c'\}$ *est satisfiable, où* S_g *est le sous-ensemble de* S'*, qui contient tous les littéraux clos.*

Démonstration. (i) ⇒ (ii) est trivial. Nous allons prouver (ii) ⇒ (i) par contradiction. Puisque $Ax(\mathcal{L}) \cup S$ et S' sont équivalents, $S \cup \{c = c'\}$ est $\mathcal{L}$-insatisfiable si et seulement si $S' \cup \{c = c'\}$ l'est aussi. Par la complétude réfutationnelle du Calcul de Superposition, on peut dériver la clause vide à partir de $S' \cup \{c = c'\}$ utilisant le système d'inférence $\mathcal{SP}$. Or S' est déjà saturé et ne contient pas de clause vide, une des inférences qui permettent de dériver la clause vide doit utiliser $c = c'$. Considérons une inférence entre $c = c'$ et un littéral l dans S' qui a pour conclusion l'. S'il y a une telle inférence, alors l doit avoir une des formes (iii), (iv) dans la Proposition 4.2, qui sont toutes closes. Analysons les différentes formes de l' :

- $c = c'$ et $c = c''$ donne $c = c''$;
- $c = c'$ et $c \neq c''$ donne $c \neq c''$;
- $c = c'$ et $f(c_1, \ldots, c, \ldots, c_n) = c''$ donne $f(c_1, \ldots, c', \ldots, c_n) = c''$;
- $c = c'$ et $car(c) = c''$ donne $car(c') = c''$;
- $c = c'$ et $cdr(c) = c''$ donne $cdr(c') = c''$;
- $c = c'$ et $cons(c, d) = c''$ donne $cons(c', d) = c''$;
- $c = c'$ et $cons(d, c) = c''$ donne $cons(d, c') = c''$;
- $c = c'$ et $cons(c, cdr(d)) = c''$ donne $cons(c', cdr(d)) = c''$;
- $c = c'$ et $cons(d, cdr(c)) = c''$ donne $cons(d, cdr(c')) = c''$;
- $c = c'$ et $cons(car(c), d) = c''$ donne $cons(car(c'), d) = c''$;
- $c = c'$ et $cons(car(d), c) = c''$ donne $cons(car(d), c') = c''$;

Maintenant, nous montrons que si l' est utilisé avec un littéral non clos dans une inférence, alors la conclusion de cette inférence peut être obtenue par une dérivation utilisant seulement des clause closes.

Prenons le cas où l a la forme $car(c) = c''$ et l' a la forme $car(c') = c''$. Alors nous avons deux possibilité pour dériver $cons(c'', cdr(c')) = c'$:

1. avec l' et $cons(car(X), cdr(X)) = X$: $car(c') = c''$ et $cons(car(X), cdr(X)) = X$ se superposent pour générer $cons(c'', cdr(c')) = c'$.

2. avec S_g et $c = c'$: $car(c) = c''$ est dans S' et on doit déjà avoir une superposition entre $car(c) = c''$ et $cons(car(X), cdr(X)) = X$ pour générer $cons(c'', cdr(c)) = c$. Maintenant, la règle Orientation s'applique à $cons(c'', cdr(c)) = c$ en utilisant $c = c'$ pour dériver $cons(c'', cdr(c')) = c'$. Et comme S' est déjà saturé $cons(c'', cdr(c)) = c$ doit être dans $S_g \subset S'$. Cela signifie que S_g et $c = c'$ suffisent pour dériver $cons(c'', cdr(c')) = c'$.

La preuve avec d'autres formes de l et l' se fait de la même façon et nous obtenons le même résultat. Ceci veut dire que le sous-ensemble S_g de S', qui ne contient que des littéraux clos, avec $c = c'$ suffit pour dériver la clause vide. □

Pour la théorie des tableaux, si on se place toujours dans le contexte de la logique mono-sortée la méthode ne s'applique plus. Considérons l'exemple suivant.

Exemple 4.2. *Considérons l'ensemble*

$$S = \left\{ \begin{array}{l} select(a'', i') = e \\ store(a, i, e) = a' \\ select(a, i') = e' \\ i \neq i' \\ e \neq e' \end{array} \right\}$$

et la $\mathcal{SP}$-saturation de

$$Ax(\mathcal{A}) \cup S = \left\{ \begin{array}{l} select(store(A, I, E), I) = E \\ I = J \vee select(store(A, I, E), J) = select(A, J) \\ select(a'', i') = e \\ store(a, i, e) = a' \\ select(a, i') = e' \\ i \neq i' \\ e \neq e' \end{array} \right\}$$

donne

$$S' = \left\{ \begin{array}{l} select(store(A, I, E), I) = E \\ I = J \vee select(store(A, I, E), J) = select(A, J) \\ select(a'', i') = e \\ store(a, i, e) = a' \\ select(a, i') = e' \\ i \neq i' \\ e \neq e' \\ select(a, J) = select(a', J) \vee J = i \end{array} \right\}$$

Maintenant, considérons l'ensemble $S' \cup \{a'' = a'\}$ dont la $\mathcal{SP}$-saturation contient nécessairement la clause vide. En effet, une superposition entre $select(a'', i') = e$ et $a'' = a'$ donne $select(a', i') = e$ qui superpose avec $select(a, J) = select(a', J) \vee J = i$ pour dériver $select(a, i') = e \vee i' = i$. Une superposition entre ce dernier avec $select(a, i') = e'$ donne $e' = e \vee i' = i$ qui conjointement avec $\{i \neq i', e \neq e'\}$ dérive la clause vide. Cependant, pour dériver la clause vide nous avons besoins de la clause $select(a, J) = select(a', J) \vee J = i$ qui n'est évidemment pas close. Ceci signifie que notre méthode ne marche plus pour des égalités entres des tableaux.

En revanche, cette situation disparaît si on se place dans le contexte de la logique multi-sortée dans lequel nous avons trois sortes *Array*, *Elem* et *Index* et la combinaison concerne les deux sortes *Elem* et *Index*. En effet, dans l'exemple précédent, l'égalité $a'' = a'$ est concerne deux constantes deux sorte *Array*. Or lorsqu'on combine une théorie de structure de données (d'une sorte $\mathcal{S}$) avec une théorie d'éléments, il n'est pas de propager les égalités entre constantes de sorte $\mathcal{S}$. Nous considérons donc seulement les égalités entre constantes de sorte différente de $\mathcal{S}$. Cette observation nous conduit au résultat suivant.

Proposition 4.10. *Soient S un ensemble $\mathcal{A}$-satisfiable de littéraux plats et clos et S' une saturation de $Ax(\mathcal{A}) \cup S$ par $\mathcal{SP}$. Soit $c = c'$ une égalité élémentaire, où c, c' sont tous les deux de sorte Elem ou tous les deux de sorte Index, alors les propriétés suivantes sont équivalentes :*

(i) $\mathcal{A} \cup S \cup \{c = c'\}$ est satisfiable,

(ii) $S_g \cup \{c = c'\}$ *est satisfiable, où* S_g *est le sous-ensemble de* S'*, qui contient tous les clauses closes.*

Démonstration. (i) ⇒ (ii) est trivial. Nous allons prouver (ii) ⇒ (i) par contradiction. Puisque $Ax(\mathcal{A}) \cup S$ et S' sont équivalents, $\mathcal{A} \cup S \cup \{c = c'\}$ est insatisfiable si et seulement si $S' \cup \{c = c'\}$ l'est aussi. Par la complétude réfutationnelle du Calcul de Superposition, on peut dériver la clause vide à partir de $S' \cup \{c = c'\}$ utilisant le système d'inférence $\mathcal{SP}$. Or S' est déjà saturé et ne contient pas de clause vide, une des inférences qui permettent de dériver la clause vide doit utiliser $c = c'$. Considérons une inférence entre $c = c'$ et un littéral l dans S' qui a pour conclusion l'. Si c, c' sont de sorte $Index$ [13], alors l doit être de la forme (iii.a) ou (iii.c) ou (iii.d) ou (iv.a) ou (iv.b) ou (iv.d) ou (iv.e) ou (iv.f) dans la Proposition 4.3. Par conséquent l' doit aussi être l'une de ces formes. De façon similaire à la théorie des listes, nous pouvons montrer qu'une inférence utilisant une clause contenant des variables génère une clause close qui peut être dérivée en utilisant seulement d'autres clauses closes. Or pour dériver la clause vide, il est nécessaire d'appliquer Reflection sur une diségalités $u \neq u$, qui est close. Ce qui veut dire que le sous-ensemble S_g de S', qui ne contient que des clauses closes, avec $c = c'$ suffit pour dériver la clause vide. □

Notons que le résultat ci-dessus se généralise sans aucune difficulté à $\mathcal{SP}'$. Il est facile de voir que la clause vide est dérivée par les deux règles Reflection et Resolution. Or au vu des formes de clauses listées dans la Proposition 4.7, les clauses utilisées par ces deux règles sont toutes closes.

4.4 Discussion

Nous avons d'abord considéré la combinaison des procédures de satisfiabilité dérivées par saturation avec d'autres procédures de satisfiabilité arbitraires par la méthode de Nelson-Oppen. Pour avoir une intégration plus efficace de la combinaison Nelson-Oppen, il est important que les procédures composantes soient complètes vis-à-vis de la déduction. Nous avons montré que le Calcul de Superposition fournit une procédure de satisfiabilité complète vis-à-vis de la déduction pour la théorie de l'égalité et la théorie des listes. Pour la théorie des tableaux nous avons montré qu'une variante du Calcul de Superposition, à savoir le Calcul de Superposition avec une règle d'analyse explicite de cas, conjointement avec un post-traitement, nous permettent d'avoir un résultat similaire. Nous avons utilisé la version d'analyse de cas sans retour en arrière (splitting without backtracking) au lieu de la version classique avec retour en arrière, implantée dans Spass [Wei97, Wei01]. Notre choix est motivé par plusieurs raisons. Premièrement, l'analyse de cas sans retour en arrière est plus simple, comme démontré dans [RV01]. Deuxièmement, l'analyse de cas sans retour en arrière a été implantée dans le système de preuve VAMPIRE [RV02]. Les expérimentations dans [RV01] ont montré que la Résolution et la Superposition avec l'analyse de cas sans retour en arrière se montre très efficace. Ceci a pour conséquence que nous pouvons obtenir facilement et efficacement la complétude vis-à-vis de la déduction en utilisant un système de preuve comme VAMPIRE combiné avec un post-traitement simple qui réalise la défragmentation.

13. Les constantes de sorte *Elem* n'apparaissent pas dans les clauses non closes (cf. Proposition 4.3).

Nous avons ensuite montré que le Calcul de Superposition peut aussi être utilisé avec la clôture de congruence pour construire des procédures de satisfiabilité pour la théorie des listes et la théorie des tableaux au lieu d'utiliser seulement le Calcul de Superposition. Notre travail a été motivé par le souhait d'avoir à la fois l'expressivité offerte par la méthodologie par saturation et l'efficacité et la flexibilité de la clôture de congruence. De plus, avec une telle architecture, nous pouvons intégrer de façon encore plus efficace des procédures de satisfiabilité dans la méthode de combinaison de Nelson-Oppen car celles-ci peuvent traiter efficacement de nouvelles formules partagées. D'un point de vue opérationnel, notre architecture fonctionne comme un prouveur de théorème basé sur l'instanciation (voir e.g. [GK03, GK04] pour plus de détails) où interviennent un module d'instanciation et un solveur de satisfiabilité close. Néanmoins, il y a une différence fondamentale entre notre architecture et un prouveur basée sur l'instanciation. En effet, dans la méthode basée sur l'instanciation, les instances de clauses sont générées par un système d'inférence conçu spécifiquement pour l'instanciation tandis que notre architecture utilise le Calcul de Superposition pour générer les instances de clauses. La correction de notre méthode doit être prouvée pour chaque théorie considérée. Nous sommes persuadés que notre méthode peut employer un module d'instanciation à la place du Calcul de Superposition afin d'obtenir un résultat similaire. Le choix du Calcul de Superposition est justifié par la réutilisation possible d'implantations efficaces de celui-ci, développés depuis longtemps. [Sch02, Wei97, RV02].

Jusqu'à maintenant, nous avons étudié de façon "ad hoc" mais très similaire la propriété de saturation finie et la complétude vis-à-vis de la déduction pour chaque théorie considérée. Ce travail nous a conduit à déterminer une classe de théories pour laquelle ces résultats s'appliquent et par la suite à développer une méthode automatique et uniforme pour la preuve de propriété de saturation finie et de complétude vis-à-vis de la déduction. Nous détaillerons la méthode utilisée et les résultats obtenus concernant cette direction de recherche dans le chapitre qui suit.

Chapitre 5

Une méthode de preuve automatique par méta-saturation

Nous avons vu dans le chapitre 4, comment le Calcul de Superposition nous permet de dériver directement des procédures de satisfiabilité pour la théorie de l'égalité, des listes et des tableaux. La preuve de correction de la méthode se réduit à la preuve de terminaison de toute saturation d'un ensemble arbitraire de littéraux plats et clos avec les axiomes de la théorie considérée. Nous avons également montré que de telles procédures de satisfiabilité sont capables de produire directement des (disjonctions de) égalités entre constantes, qui sont des conséquences de l'ensemble de litéraux en entrée. Comme la théorie de l'égalité, des listes et des tableaux sont toutes stablement infinies, ces procédures de satisfiabilité peuvent être combinées efficacement avec d'autres procédures de satisfiabilité en utilisant la méthode de combinaison de Nelson-Oppen.

Nous avons pu voir aussi que pour chaque théorie considérée la preuve de correction des méthodes proposées est faite de façon "ad hoc", malgré une forte ressemblance dans le déroulement. Un objectif est donc de développer une méthode automatisant, pour une théorie axiomatisée par une ensemble fini de clauses, la preuve de

- la propriété de saturation finie, qui est fondamentale pour avoir une procédure de satisfiabilité ;
- la stable infinité, qui est cruciale pour la correction de la méthode de combinaison à la Nelson-Oppen ;
- la propriété de complétude vis-à-vis de la déduction ;
- la correction de la coopération de la clôture de congruence et de la saturation.

Nous montrons que nous pouvons adopter la méta-saturation, développée par Lynch et Morawska [LM02], pour réaliser ces objectifs.

Nous introduisons d'abord le concept de clause variable-active qui nous permet d'établir des critères syntaxiques pour déterminer, pour une théorie axiomatisée par un ensemble fini de clauses, la propriété de saturation finie, la stable infinité et la complétude vis-à-vis de la déduction. Ensuite nous revisitons la méta-saturation et son utilisation pour prouver les propriétés mentionnées. En ce qui concerne la coopération de la clôture de congruence et de la saturation, nous introduisons le concept de clause constante-active qui nous permet de définir des critères syntaxiques afin de déterminer automatiquement si cette coopération s'applique à une théorie donnée. Finalement, nous étendons la méthode à la saturation par rapport au Calcul de Superposition

avec fragmentation, décrite dans le chapitre 4.

5.1 Propriété de variable-inactivité

Nous commençons par donner les observations qui nous ont conduit à définir la propriété de variable-inactivité—un concept fondamental dans notre méthode automatique pour déterminer la stable infinité, la complétude vis-à-vis de la déduction et la modularité de la propriété de saturation finie. Par un souci de clarté, nous considérons d'abord uniquement les théories équationnelles.

Soit T une théorie axiomatisée par un ensemble fini $Ax(T)$ d'égalités et un ensemble arbitraire T-satisfiable S de littéraux plats et clos.

Si nous souhaitons montrer que T est stablement infini, nous devons montrer que S est satisfiable dans un modèle infini de T. Nous avons observé [KRRT06a] qu'un ensemble de formules n'a pas de modèle infini si celui-ci implique une égalité de la forme $X = t$, où X est une variable qui n'apparaît pas dans t. Si nous réussissons à montrer que la saturation de tout ensemble de littéraux ne contient pas d'égalité de la forme $X = t$, nous avons alors un critère syntaxique pour déterminer la stable infinité.

En ce qui concerne la complétude vis-à-vis de la déduction, nous devons montrer que pour toute égalité élémentaire $c = c'$, $T \models S \Rightarrow c = c'$ si et seulement si le sous-ensemble de la saturation de $Ax(T) \cup S$, qui contient toutes les égalités élémentaires, implique $c = c'$. Or par la complétude réfutationnelle de $\mathcal{SP}$, si $T \models S \Rightarrow c = c'$ alors la saturation de $Ax(T) \cup S \cup \{c \neq c'\}$ dérive nécessairement la clause vide. Comme S est satisfiable, pour dériver la clause vide il est nécessaire d'utiliser $c \neq c'$. S'il y a une inférence entre $c \neq c'$ et un littéral C, alors C doit contenir seulement des égalités entre variable ou constantes. Si C contient des variables, C a la forme $X = c$ et ainsi subsume des égalités élémentaires. Par conséquent nous n'avons plus la complétude vis-à-vis de la déduction. Il est donc nécessaire d'imposer des conditions pour exclure ce type de littéral de la saturation de $Ax(T) \cup S$.

Pour la modularité de la propriété de saturation finie, étant données deux théories équationnelles T_1, T_2 ayant la propriété de saturation finie, nous devons considérer toutes les inférences croisées entre les théories et montrer qu'elles ne génèrent qu'un nombre fini de littéraux pendant une saturation de $Ax(T_1) \cup Ax(T_2) \cup S$ pour tout ensemble S de littéraux plats et clos. Puisque les signatures des théories sont disjointes, les inférences croisées entre les théories ne se produisent qu'à travers les constantes, les variables ou les symboles de fonction non interprétés. Il est facile de voir que les inférences aux constantes ou aux symboles non interprétés ne génèrent qu'un nombre fini de littéraux. Seules les inférences aux variables peuvent éventuellement générer un nombre infini de littéraux. Or les inférences aux variables se font seulement lorsqu'on a des littéraux de la forme $X = t$, où X est une variable qui n'apparaît pas dans t. Il suffit donc d'exclure ce type de clause pour avoir la modularité de la propriété de saturation finie.

Toutes ces observations nous ont conduit à introduire le concept de clause variable-active, qui est formellement défini comme suit.

Définition 5.1 (Clause variable-active)**.** *Une clause C est variable-active par rapport à un ordre $\succ$ si*

– *C contient un littéral de la forme $X = t$, où $X \notin Var(t)$, et*

– *il existe une substitution λ telle que $\lambda(X = t)$ est maximal dans $\lambda(C)$ et $\lambda(X)$ est maximal dans $\lambda(X = t)$ par rapport à $\succ$.*

Dans ce qui suit, une clause est dite variable-active si elle est variable-active par rapport à l'ordre utilisé dans $\mathcal{SP}$.

Proposition 5.1. *Les propriétés suivantes sont équivalentes :*

(i) C est une clause variable-active.

(ii) C contient un littéral maximal de la forme $X = t$, où $X \notin Var(t)$.

Démonstration. (i) $\Rightarrow$ (ii). Si C ne contient pas d'égalité de la forme $X = t$, alors C n'est pas variable-active. Si C contient des littéraux de la forme $X = t$ mais ils ne sont pas maximaux dans C alors C n'est pas variable-active. Supposons que C contient un littéral maximal de la forme $X = t$ mais $X \in Var(t)$, alors par propriété de sous-terme, $\lambda(X)$ ne serait pas maximal dans $\lambda(X = t)$ pour toute substitution λ. Donc C ne serait pas variable-active.
(ii) $\Rightarrow$ (i). Supposons que C contient un littéral maximal de la forme $X = t$, où $X \notin Var(t)$. Alors nous pouvons choisir la substitution vide ϵ qui satisfait la définition de clause variable-active. □

Définition 5.2 (Propriété de variable-inactivité)**.** *Soit T une théorie axiomatisée par un ensemble fini $Ax(T)$ de clauses. On dit que T a la propriété de variable-inactivité si pour tout ensemble S de littéraux plats et clos, toute saturation de $Ax(T) \cup S$ par $\mathcal{SP}$ ne contient pas de clause variable-active.*

5.2 Méta-saturation revisitée

La méta-saturation [LM02] a été conçue pour un double objectif : prouver automatiquement la terminaison de la saturation de l'ensemble des axiomes d'une théorie avec un ensemble arbitraire de littéraux plats et clos ; et dériver également la complexité d'une telle saturation. Étant donnée une théorie T axiomatisée par un ensemble fini $Ax(T)$ de clauses, la méta-saturation pour la théorie T simule le processus de la saturation de $Ax(T) \cup S$, pour tout un ensemble S arbitraire de littéraux plats et clos. La simulation consiste à saturer, par rapport au système d'inférence $m\mathcal{SP}$ (voir les figures 5.1 et 5.2), l'ensemble des axiomes $Ax(T)$ avec un ensemble G_0^T, que nous allons définir dans un instant, qui intuitivement schématise tout ensemble fini de littéraux construits à partir des symboles dans la signature de T. Si la saturation de $Ax(T) \cup G_0^T$ par rapport à $m\mathcal{SP}$ termine alors la saturation de $Ax(T) \cup S$ termine également pour toute ensemble S de littéraux plats et clos et ainsi le Calcul de Superposition fournit une procédure de satisfiabilité pour T. Dans ce qui suit, nous allons voir les principales idées de la méta-saturation.

Définition 5.3 (Contrainte de constante)**.** *Une* contrainte de constante atomique *a la forme $const(t)$ et elle est vraie si t est une constante. Une* contrainte de constante *a la forme $const(t_1) \wedge \ldots \wedge const(t_n), n \geq 0$.*

Définition 5.4 (Clause contrainte)**.** *Une clause avec contrainte de constante, ou simplement clause contrainte, a la forme $C \parallel \phi$, où C est une clause et ϕ est une contrainte de constante.*

Une substitution λ satisfait une contrainte de constante ϕ si $\lambda(\phi)$ est vrai. Une variable x est contrainte dans une clause contrainte $C \parallel \phi$ si ϕ contient $const(x)$, sinon elle est non contrainte cette clause. Il est facile de voir que les variables non contraintes correspondent aux variables universellement quantifiées tandis que les variables contraintes correspondent aux constantes.

Définition 5.5 (Instance contrainte)**.** *Soit λ une substitution. On dit que $\lambda(C)$ est une* instance contrainte *de $C \parallel \phi$ si $\mathcal{D}om(\lambda) = Var(\phi)$ et $\mathcal{R}an(\lambda)$ ne contient que des constantes. Une clause contrainte est close si une de ses instances contraintes est close.*

Il est facile de voir que si une des instances contraintes d'une clause contrainte est close alors toutes ses instances contraintes sont également closes. Par conséquent une clause contrainte est non close si une de ses instances contraintes contient au moins une variable. Nous utilisons la notation $Var(C \parallel \phi)$ pour désigner l'ensemble des variables non contraintes de C.

Définition 5.6. *Soit G_0^T défini comme suit :*

$$G_0^T = \{x = y \parallel const(x) \wedge const(y)\} \cup \{x \neq y \parallel const(x) \wedge const(y)\} \cup \bigcup_{f \in \Sigma_T} \{f(x_1, \ldots, x_n) = x_0 \parallel \bigwedge_{i=0}^{n} const(x_i)\},$$

où Σ_T est la signature de T.

Avant de présenter le système d'inférence $m\mathcal{SP}$ décrivant la méta-saturation, nous avons besoin d'étendre l'ordre $\succ$ utilisé dans $\mathcal{SP}$ aux clauses contraintes car celui-ci manipule les clauses contraintes.

Hypothèse 5.1. *$\succ$ est étendu de façon à ce qu'une clause contrainte C est plus grande qu'une clause contrainte D si toutes les instances closes de C sont plus grandes que les instances closes de D.*

Le système d'inférence $m\mathcal{SP}$ (voir les figures 5.1 et 5.2) est presque identique à $\mathcal{SP}$, sauf que les clauses traitées ont maintenant une contrainte de constante, qui est héritée par la conclusion d'une inférence. Les conditions d'application des règles d'inférence sont aussi légèrement modifiées. Ceci est dû au fait que nous ne pouvons pas simuler toutes les simplifications, suppressions ou encore subsomptions car nous ne pouvons pas imposer que certains littéraux, dont dépendent ces inférences, persistent au cours de la saturation. Aussi la règle Orientation n'est pas présent dans $m\mathcal{SP}$.

Nous définissons la méta-saturation d'une théorie T axiomatisée par un ensemble fini $Ax(T)$ d'axiomes comme étant la saturation de $Ax(T) \cup G_0^T$ par rapport à $m\mathcal{SP}$.

Définition 5.7 (Propriété de méta-saturation finie)**.** *Soit $Ax(T)$ l'ensemble fini des axiomes de la théorie T. On dit que T a la propriété de méta-saturation finie par rapport à $\mathcal{SP}$ si toute saturation de $Ax(T) \cup G_0^T$ par $m\mathcal{SP}$ est finie.*

Une clause contrainte est variable-active par rapport à un ordre $\succ$ si une de ses instances contraintes est variable-active par rapport à $\succ$.

Proposition 5.2. *Soit $C \parallel \phi$ une clause contrainte. Alors, les conditions suivantes sont équivalentes :*

Superposition − droite

$$\frac{\Gamma \Rightarrow \Delta, l[u'] = r \parallel \phi \quad \Pi \Rightarrow \Sigma, u = t \parallel \varphi}{\sigma(\Gamma, \Pi \Rightarrow \Delta, \Sigma, l[t] = r \parallel \phi \wedge \varphi)} \quad \textbf{si } (i), (ii), (iii), (iv)$$

Superposition − gauche

$$\frac{\Gamma, l[u'] = r \Rightarrow \Delta \parallel \phi \quad \Pi \Rightarrow \Sigma, u = t \parallel \varphi}{\sigma(l[t] = r, \Gamma, \Pi \Rightarrow \Delta, \Sigma \parallel \phi \wedge \varphi)} \quad \textbf{si } (i), (ii), (iii), (iv)$$

Reflection

$$\frac{\Gamma, u' = u \Rightarrow \Delta \parallel \phi}{\sigma(\Gamma \Rightarrow \Delta \parallel \phi)} \quad \textbf{si } (v)$$

Factorisation

$$\frac{\Gamma \Rightarrow \Delta, u = t, u' = t' \parallel \phi}{\sigma(\Gamma, t = t' \Rightarrow \Delta, u = t' \parallel \phi)} \quad \textbf{si } (i), (vi)$$

σ est l'unificateur principal de u et u', u' n'est pas une variable dans Superposition − droite et Superposition − gauche, L est un littéral, et les conditions suivantes sont satisfaites :

(i) $\sigma(u) \not\preceq \sigma(t)$,
(ii) $\forall L \in \Pi \cup \Sigma : \sigma(u = t) \not\preceq \sigma(L)$,
(iii) $\sigma(l[u']) \not\preceq \sigma(r)$,
(iv) $\forall L \in \Gamma \cup \Delta : \sigma(l[u'] = r) \not\preceq \sigma(L)$,
(v) $\forall L \in \Gamma \cup \Delta : \sigma(u' = u) \not\prec \sigma(L)$,
(vi) $\forall L \in \Gamma : \sigma(u) \not\preceq \sigma(L)$, $\forall L \in \{u' = t'\} \cup \Delta : \sigma(u = t) \not\prec \sigma(L)$.

FIGURE 5.1 – Règles de Clôture de $m\mathcal{SP}$.

(i) une des instances contraintes de $C \parallel \phi$ est variable-active ;

(ii) toute instance contrainte de $C \parallel \phi$ est variable-active.

Démonstration. Si une variable X apparaît dans une instance contrainte de $C \parallel \phi$, alors X doit être non contrainte dans $C \parallel \phi$, i.e. $const(X)$ n'est pas dans ϕ. Il est facile de voir que si une variable non contrainte X apparaît dans $C \parallel \phi$, alors X apparaît dans toutes les instances contraintes de $C \parallel \phi$ comme une variable. □

Théorème 5.1 ([LM02])**.** *Soit T une théorie axiomatisée par un ensemble fini $Ax(T)$ de clauses, qui est saturé par rapport à $\mathcal{SP}$. Soit G^T_∞ une saturation finie de $Ax(T) \cup G^T_0$ par $m\mathcal{SP}$. Si G^T_∞ ne contient pas de clause variable-active alors pour tout ensemble S de Σ_T-littéraux plats et clos, chaque clause dans la saturation de $Ax(T) \cup S$ par $\mathcal{SP}$ a une des formes suivantes :*

(A) une disjonction d'égalités entre constantes $c_1 = c'_1 \vee \ldots \vee c_n = c'_n$, tel que $n \geq 1$;

Subsomption

$$\frac{S \cup \{C, C' \parallel \phi\}}{S \cup \{C\}}$$

$$\textbf{si} \begin{cases} C \in Ax(T), \exists \theta : \theta(C) \subseteq C'.\text{ou} \\ C \text{ et } C' \parallel \phi \text{ sont des renomages l'un de l'autre} \end{cases}$$

Simplification

$$\frac{S \cup \{C[l'] \parallel \phi, l = r\}}{S \cup \{C[\theta(r)] \parallel \phi, l = r\}}$$

$$\textbf{si} \begin{cases} l = r \in Ax(T) \\ l' \equiv \theta(l) \\ \theta(l) \succ \theta(r) \\ \forall L \in C[\theta(l)] : L \succ (\theta(l) = \theta(r)) \end{cases}$$

Suppression

$$\frac{S \cup \{\Gamma \Rightarrow \Delta, t = t \parallel \phi\}}{S}$$

$$\frac{S \cup \{\Gamma \Rightarrow \Delta \parallel \phi\}}{S} \quad \textbf{si } \phi \text{ est insatisfiable}$$

où C et C' sont des clauses est S est un ensemble de clauses.

FIGURE 5.2 – Règles de Simplification de $m\mathcal{SP}$.

(B) une clause de la forme $C \vee c_1 = c'_1 \vee \ldots \vee c_n = c'_n$*, tel que* $n \geq 0$ *et* C *est une instance contrainte d'une clause* C' *dans* G^T_∞

où $c_1, \ldots, c_n, c'_1, \ldots, c'_n, c''_1, \ldots, c''_m$ *sont des constantes.*

Il est important de noter que dans [LM02], le Théorème 5.1 s'énonce avec une troisième forme de clause, à savoir

$$f(c''_1, \ldots, c''_m) = c'' \vee c_1 = c'_1 \vee \ldots \vee c_n = c'_n$$

telle que $m \geq 1$, $n \geq 0$, $f \notin \Sigma_T$ et $c_1, \ldots, c_n, c'_1, \ldots, c'_n, c''_1, \ldots, c''_m$ sont des constantes. De cette façon, l'ensemble S peut être un ensemble de littéraux plats et clos et pas seulement un ensemble de littéraux plats et clos sur la signature de T.

Nous adoptons la nouvelle formulation pour deux raisons. La première est que nous nous intéressons à prouver la combinabilité d'une seule théorie, et non le mélange de celle-ci avec la théorie de l'égalité. La deuxième raison est que ce nouvel énoncé permet de corriger une imperfection dans l'énoncé original. En effet, lorsqu'on considère des symboles de fonction non interprétés, il se peut que la méta-saturation ne simule pas toutes les inférences de la saturation. Considérons l'exemple suivant.

Exemple 5.1. *Soit T la théorie axiomatisée par*

$$Ax(T) = \left\{ \; X = Y \vee Y = Z \vee Z = X \; \right\}$$

G_0^T contient les clauses suivantes

$$\left\{ \begin{array}{l} x = y \parallel const(x) \wedge const(y) \\ x \neq y \parallel const(x) \wedge const(y) \end{array} \right\}$$

La saturation G_∞^T de $Ax(T) \cup G_0^T$ par $m\mathcal{SP}$ est $Ax(T) \cup G_0^T$.

Maintenant, considérons l'ensemble $S = \{f(c) = c\}$. La saturation de $Ax(T) \cup S$ génère la clause $f(Y) = c \vee Y = Z \vee Z = c$. Il est facile de voir que cette clause n'est une instance contrainte d'aucune clause dans G_∞^T. De plus elle n'est subsumée par aucune autre clause. Par conséquent, G_∞^T ne schématise pas la saturation de $Ax(T) \cup S$ par $\mathcal{SP}$.

Aussi, par rapport au résultat énoncé dans [LM02], le Théorème 5.1 suppose que la méta-saturation ne dérive pas de clause variable-active. En effet, si la méta-saturation dérive une clause variable-active, il se peut qu'elle ne schématise plus toutes les saturations. L'exemple suivant illustre bien cette situation.

Exemple 5.2. *Soit T la théorie axiomatisée par*

$$Ax(T) = \left\{ \begin{array}{l} X = Y \vee Y = Z \vee Z = X \\ f(X) = g(X) \end{array} \right\}$$

G_0^T contient les clauses suivantes

$$\left\{ \begin{array}{l} x = y \parallel const(x) \wedge const(y) \\ x \neq y \parallel const(x) \wedge const(y) \\ f(x) = y \parallel const(x) \wedge const(y) \\ g(x) = y \parallel const(x) \wedge const(y) \end{array} \right\}$$

La saturation G_∞^T de $Ax(T) \cup G_0^T$ par $m\mathcal{SP}$ est égale $Ax(T) \cup G_0^T$.

Considérons l'ensemble $S = \{f(c) = c'\}$. La saturation de $Ax(T) \cup S$ génère la clause $f(Y) = c' \vee Y = Z \vee Z = c$ qui n'est pas schématisée au niveau de la méta-saturation. La raison est que $X = Y \vee Y = Z \vee Z = X$ est une clause variable-active.

Dans ce qui suit, quand l'on parle de littéral (resp. formule, clause) on fait référence à un Σ_T-littéral (resp. Σ_T-formule, Σ_T-clause) lorsque la théorie T est non ambigüe dans le contexte.

Le corollaire suivant, qui résulte directement du Théorème 5.1, met en évidence le lien entre méta-saturation et saturation.

Corollaire 5.1. *Si T est une théorie ayant la propriété de méta-saturation finie alors T a la propriété de saturation finie.*

Les exemples suivants servent à illustrer les idées de la méta-saturation.

Exemple 5.3. *La théorie $\mathcal{E}$ n'a pas d'axiomes et donc $G_0^{\mathcal{E}}$ contient les clauses suivantes :*

$$\left\{ \begin{array}{l} x = y \parallel const(x) \wedge const(y) \\ x \neq y \parallel const(x) \wedge const(y) \end{array} \right\}$$

Il est facile de voir que $G_0^{\mathcal{E}}$ est déjà saturé par rapport à $m\mathcal{SP}$ et par conséquent $G_\infty^{\mathcal{E}} = G_0^{\mathcal{E}}$. Il résulte du Corollaire 5.1 que $\mathcal{SP}$ est une procédure de satisfiabilité pour $\mathcal{E}$.

Exemple 5.4. *$G_0^{\mathcal{L}}$ contient les clauses dans $G_0^{\mathcal{E}} \cup Ax(\mathcal{L})$ et les clauses suivantes :*

$$\left\{ \begin{array}{l} car(x) = y \parallel const(x) \wedge const(y), \\ cdr(x) = y \parallel const(x) \wedge const(y), \\ cons(x,y) = z \parallel const(x) \wedge const(y) \wedge const(z) \end{array} \right\}$$

En appliquant la saturation de $G_0^{\mathcal{E}} \cup Ax(\mathcal{L})$ par $m\mathcal{SP}$, nous obtenons l'ensemble $G_\infty^{\mathcal{L}}$ contenant des clauses dans $G_\infty^{\mathcal{E}} \cup Ax(\mathcal{L}) \cup G_0^{\mathcal{L}}$ et les clauses :

$$\left\{ \begin{array}{l} cons(car(x), y) = z \parallel const(x) \wedge const(y) \wedge const(z) \\ cons(x, cdr(y)) = z \parallel const(x) \wedge const(y) \wedge const(z) \end{array} \right\}$$

Encore une fois, par le Corollaire 5.1, $\mathcal{SP}$ est une procédure de satisfiabilité pour $\mathcal{L}$.

Bien que la saturation termine pour la théorie des tableaux, la méta-saturation ne termine pas, contrairement à ce qui est indiqué dans [LM02].

Exemple 5.5. *$G_0^{\mathcal{A}}$ contient des clauses dans $G_0^{\mathcal{E}}$ et les clauses :*

$$\left\{ \begin{array}{l} select(x,y) = z \parallel const(x) \wedge const(y) \wedge const(z) \\ store(x,y,z) = t \parallel const(x) \wedge const(y) \wedge const(z) \wedge const(t) \end{array} \right\}$$

La saturation de $Ax(\mathcal{A}) \cup G_0^{\mathcal{A}}$ contient une clause de la forme

$$select(x,t) = select(z,t) \vee y = t \parallel const(x) \wedge const(y) \wedge const(z)$$

Cette clause superpose avec un renommage d'elle-même, i.e. une clause de la forme $select(x',t') = select(z',t') \vee y' = t' \parallel const(x') \wedge const(y') \wedge const(z')$, génèrera une clause d'une nouvelle forme

$$select(x,t) = select(z,t) \vee y = t \vee y' = t \parallel const(x) \wedge const(y) \wedge const(z) \wedge const(y')$$

Cette nouvelle clause se superpose encore avec son renommage pour générer une clause d'une nouvelle forme. Nous avons ainsi une dérivation infinie des clauses de la forme $select(x,t) = select(z,t) \vee y_1 = t \vee \ldots \vee y_n = t \parallel const(x) \wedge const(y) \wedge const(z) \wedge const(y_1) \wedge \ldots \wedge const(y_n)$.

Dans ce qui suit, nous aurons besoin du concept de clause variable-active pour développer une méthode automatique pour vérifer la stable infinité, la complétude vis-à-vis de la déduction. Le résultat suivant nous permet de déterminer automatiquement la propriété de variable-inactivité d'une théorie par méta-saturation.

Théorème 5.2. *Soit T une théorie axiomatisée par un ensemble fini $Ax(T)$ de clauses. Si la méta-saturation de T ne contient pas de clause variable-active, alors T a la propriété de variable-inactivité.*

Démonstration. Il est facile de voir qu'une clause variable-active doit être de la forme (B) dans le Théorème 5.1. Ce qui signifie que si T n'a pas la propriété de variable-inactivité, alors la saturation de $Ax(T) \cup G_0^T$ par $m\mathcal{SP}$ contient nécessairement une clause variable-active. □

5.3 Stable infinité

La méthode de combinaison de Nelson-Oppen nous permet de combiner les procédures de satisfiabilité pour la classe des théories stablement infinies. L'hypothèse de stable infinité est cruciale pour la correction de cette méthode de combinaison. C'est pour cette raison qu'il est important d'avoir une méthode automatique pour prouver la stable infinité des théories axiomatisées par un ensemble fini de clauses. Nous allons montrer que la méta-saturation peut être utilisée pour développer une telle méthode.

Rappelons que $\exists^{\geq n}$ désigne la formule $\exists x_1 \dots x_n . \bigwedge_{j \neq k}(x_j \neq x_k)$, et $\exists^{\leq n}$ désigne la formule $\forall x_0 \dots x_n . \bigvee_{j \neq k}(x_j = x_k)$. Et on note par $\exists^{\infty}$ la formule $\{\exists^{\geq n} | n \geq 2\}$. Il est facile de voir que $\exists^{\geq n}$, $\exists^{\leq n}$, et $\exists^{\infty}$ imposent à la cardinalité de chacun de leurs modèles d'être respectivement au moins n, au plus n et infinie.

Lemme 5.1. *Soit T une théorie du premier ordre. Si T n'a pas de modèle infini, alors il existe un entier positif n tel que pour chaque modèle $\mathcal{M}$ de T, la cardinalité de $\mathcal{M}$ est inférieure ou égale à n.*

Démonstration. Supposons par contradiction que pour tout entier positif n, T a un modèle de cardinalité au moins n. Nous allons prouver que T a un modèle infini. Pour cela, il nous faut juste prouver que $T \cup \exists^{\infty}$ a un modèle. Par compacité, il suffit de prouver que tout sous-ensemble fini Γ_0 de $T \cup \exists^{\infty}$ a un modèle. En effet, il existe un n_0 tel que

$$\Gamma_0 \subseteq T \cup \{\exists^{\geq 2}, \exists^{\geq 3}, \dots, \exists^{\geq n_0}\}.$$

Or T a un modèle avec au moins n_0 éléments dans son domaine. Ce qui implique que $T \cup \{\exists^{\geq 2}, \exists^{\geq 3}, \dots, \exists^{\geq n_0}\}$ a un modèle. Par compacité, Γ_0 a un modèle. □

Définition 5.8 (Contrainte de modèle fini)**.** *Une clause est une contrainte de modèle fini si c'est une disjonction d'égalités entre variables.*

Lemme 5.2. *Soit T une théorie consistante. Si T n'a pas de modèle infini alors T implique nécessairement une contrainte de modèle fini.*

Démonstration. Supposons que T n'a pas de modèle infini. Soit $\mathcal{M}$ un modèle de T. Alors, nous avons $\mathcal{M} \models \exists^{\leq \kappa}$, où κ est la cardinalité de $\mathcal{M}$. Par le Lemme 5.1, il existe un entier positive n tel que la cardinalité de chaque modèle de T est inférieure ou égale à n. Ceci implique qu'il existe un nombre fini de modèles de T, modulo isomorphisme. On note C la clause $\bigvee_{i=1}^{r} \exists^{\leq \kappa_i}$, où $\mathcal{M}_i$ est un modèle de T, $\kappa_i = |M_i|$ (pour $i = 1, ..., r$), et les quantificateurs universels sont implicites dans $\exists^{\leq \kappa_i}$. Il est évident que $T \models C$ où C est une contrainte de modèle fini. □

Lemme 5.3. *Soit T une théorie consistante, axiomatisée par un ensemble fini $Ax(T)$ de clauses. S'il existe un ensemble fini T-satisfiable S de littéraux clos tel que $T \cup S$ implique une contrainte de modèle fini, alors toute saturation de $Ax(T) \cup S$ contient nécessairement une clause variable-active.*

Démonstration. Supposons qu'il existe une contrainte de modèle fini telle que $T \cup S \models C$. En raisonnant par réfutation, $T \cup S \models C$ si et seulement si $S \wedge \neg C$ n'est pas T-satisfiable. Donc, il doit être possible de dériver la clause vide par application de $\mathcal{SP}$ à partir de $Ax(T) \cup S \cup Sk(\neg C)$, où $C \equiv \exists^{\leq n}$ et $Sk(\neg C)$ étant l'ensemble $\{c_i^{sk} \neq c_j^{sk} | \ 0 \leq i \neq j \leq n\}$ tel que c_i^{sk}, c_j^{sk} sont des constantes de Skolem (souvenons

que chaque variable universelle implicite dans C devient existentielle dans $\neg C$). Pour obtenir une saturation de $Ax(T) \cup S \cup Sk(\neg C)$, il est possible d'obtenir d'abord une saturation S' de $Ax(T) \cup S$ par $\mathcal{SP}$ et puis de considérer toutes les inférences possibles entre les clauses de S' et celles de $Sk(\neg C)$. Aucun autre type d'inférence est possible pour dériver la clause vide car S' est déjà saturé par rapport à $\mathcal{SP}$ et $Sk(\neg C)$ ne contient que des clauses négatives. Considérons les inférences entre une clause $c_i^{sk} \neq c_j^{sk}$ et une clause $s = t \vee D$ dans S'. Ce type d'inférence n'est possible que si s, t sont des variables ou constantes. Si s, t sont des constantes alors l'inférence n'est pas possible car c_i^{sk}, c_j^{sk} sont des constantes de Skolem qui n'apparaissent pas dans $s = t \vee D$. Supposons que t est une variable, alors nous pouvons ainsi dériver la clause $\sigma(s \neq c_i^{sk} \vee D)$, où σ est l'unificateur principal de t et c_j^{sk}. Or à partir de cette clause, si on souhaite dériver la clause vide il faut que Reflection s'applique à $\sigma(s \neq c_i^{sk})$. Ce qui implique que s doit être une variable. s, t ne doivent pas apparaître comme sous-termes dans D car autrement $s = t$ ne serait pas maximal et ne serait pas sélectionné. De plus, D ne doit pas contenir une diségalité car sinon, $s = t$ ne serait pas maximal et ne serait pas sélectionné. Par la Proposition 5.1, $s = t \vee D$ est variable-active, et donc T n'a pas la propriété de variable-inactivité. □

Nous sommes maintenant prêt à énoncer le résultat qui nous donne une méthode automatique pour vérifier la stable infinité des théories axiomatisées par un ensemble fini de clauses.

Théorème 5.3. *Soit T une théorie consistante telle que*
- *T est axiomatisée par un ensemble fini $Ax(T)$ de clauses, qui est saturé par rapport à $\mathcal{SP}$, et*
- *la méta-saturation de T ne contient pas de clause variable-active.*

Alors T est stablement infinie.

Démonstration. Par contradiction, supposons que T n'est pas stablement infinie. Alors il existe un ensemble T-satisfiable S de littéraux clos tel que $T \cup S$ n'a pas de modèle infini. Par le Lemme 5.2, $T \cup S$ implique une contrainte de modèle fini. Il résulte du Lemme 5.3 que la saturation de $Ax(T) \cup S$ contient une clause variable-active C. Or C doit être de la forme (B) dans le Théorème 5.1. Par conséquent la méta-saturation de T contient une clause variable-active, ce qui est en contradiction avec l'hypothèse du théorème. □

Exemple 5.6. *Nous pouvons voir que dans l'exemple 5.3 (respectivement 5.4), $G_\infty^{\mathcal{E}}$ (respectivement $G_\infty^{\mathcal{L}}$) ne contient pas de clause variable-active et ainsi par le Théorème 5.3, la théorie de l'égalité (respectivement la théorie des listes et la théorie des tableaux) est stablement infinie.*

5.4 Complétude vis-à-vis de la déduction

Comme discuté précédemment, pour avoir une intégration efficace de procédures de satisfiabilité dans la méthode de combinaison de Nelson-Oppen, il est indispensable que ces procédures soient capables de dériver directement les égalités entre constantes, qui sont ensuite propagées à d'autres procédures. Dans le chapitre 4, nous avons

prouvé de façon "ad hoc" que $\mathcal{SP}$ est une procédure de satisfiabilité complète vis-à-vis de la déduction pour la théorie de l'égalité et la théorie des listes. Nous allons généraliser ce résultat aux théorie de Horn.

Lemme 5.4. *Soit T une théorie telle que*
- *T est axiomatisée par un ensemble fini $Ax(T)$ de clauses de Horn, et*
- *T a la propriété de variable-inactivité.*

Soient S un ensemble T-satisfiable de littéraux plats et clos et S' une saturation de $Ax(T) \cup S$ par $\mathcal{SP}$. Soit $c = c'$ une égalité élémentaire, alors les propriétés suivantes sont équivalentes :

(i) $T \models S \Rightarrow c = c'$,

(ii) $S_e \models c = c'$, *où S_e est le sous-ensemble de S', qui contient toutes les égalités élémentaires.*

Démonstration. (ii) $\Rightarrow$ (i) est trivial. Nous allons prouver (i) $\Rightarrow$ (ii). Puisque $Ax(T) \cup S$ et S' sont équivalents, $T \models S \Rightarrow c = c'$ si et seulement si $S' \models c = c'$. Ce qui veut dire aussi que $S' \cup \{c \neq c'\}$ est insatisfiable. Par la complétude réfutationnelle du Calcul de Superposition, on peut dériver la clause vide à partir de $S' \cup \{c \neq c'\}$ en utilisant le système d'inférence $\mathcal{SP}$. Or S' est déjà saturé et ne contient pas de clause vide, les inférences qui permettent de dériver la clause vide doivent avoir comme prémisses des clauses dans S' et $c \neq c'$ ou des clauses dérivées à partir de S' et $c \neq c'$. Considérons maintenant les types d'inférences avec de telles prémisses. Soit C une clause dans S'. S'il y a une inférence utilisant C et $c \neq c'$, alors C doit être une égalité entre constantes ou variables. Si C contient une variable, alors C est une égalité de la forme $X = u$, où u est une constante, et donc $X = u$ est variable-active, ce qui est en contradiction avec l'hypothèse du lemme car S' ne contient pas de clause variable-active. Donc C doit être une égalité entre constantes. Par conséquent la conclusion d'une telle inférence sera une diségalité entre constantes. La preuve se termine comme dans le cas de la théorie de l'égalité. □

Le Lemme 5.4 nous permet de conclure le résultat suivant, qui généralise les résultats de complétude vis-à-vis de la déduction pour la théorie de l'égalité et la théorie des listes.

Théorème 5.4. *Soit T une théorie telle que*
- *T est axiomatisée par ensemble fini $Ax(T)$ de clauses de Horn, et*
- *T a la propriété de saturation finie, et*
- *T a la propriété de variable-inactivité.*

Alors $\mathcal{SP}$ est une procédure de satisfiabilité complète vis-à-vis de la déduction pour T.

Les Théorèmes 5.1, 5.2 et 5.4 nous donnent directement une méthode automatique par méta-saturation pour déterminer la complétude vis-à-vis de la déduction pour les théories de Horn qui ont la propriété de saturation finie.

Corollaire 5.2. *Soit T une théorie telle que*
- *T est axiomatisée par un ensemble fini $Ax(T)$ de clauses de Horn, qui est saturé par rapport à $\mathcal{SP}$, et*
- *T a la propriété de méta-saturation finie, et*

– la saturation de $Ax(T) \cup G_0^T$ par $m\mathcal{SP}$ ne contient pas de clause variable-active. Alors $\mathcal{SP}$ est une procédure de satisfiabilité complète vis-à-vis de la déduction pour T.

Exemple 5.7. *La théorie de l'égalité (respectivement la théorie des listes) est une théorie de Horn et $G_\infty^{\mathcal{E}}$ (respectivement $G_\infty^{\mathcal{L}}$) ne contient pas de clause variable-active. Par le Corollaire 5.2, $\mathcal{SP}$ est une procédure de satisfiabilité complète vis-à-vis de la déduction pour la théorie de l'égalité (respectivement la théorie des listes).*

5.5 Coopération de la clôture de congruence et de la saturation

Nous avons proposé précédemment une coopération de la clôture de congruence et de la saturation qui nous permet de construire des procédures de satisfiabilité capables de traiter efficacement les nouvelles égalités élémentaire sans avoir à refaire la saturation. Nous avons montré de façon "ad hoc" pour la théorie des listes et la théorie des tableaux que notre méthode est correcte. Dans cette section, nous définissons des critères permettant d'identifer une classe plus large de théories pour laquelle notre méthode s'applique. Nous développons également une méthode automatique, toujours par méta-saturation, afin de vérifier si une théorie axiomatisée par un ensemble fini de clauses satisfait ces critères.

Définition 5.9 (Clause constante-active)**.** *Une clause est constante-active si un de ses littéraux maximaux contient au moins une constante. Une clause est constante-inactive si elle n'est pas constante-active. Une clause contrainte est constante-active si une de ses instances contraintes est constante-active.*

Proposition 5.3. *Soit $C \parallel \phi$ une clause contrainte. Alors, les conditions suivantes sont équivalentes :*

(i) une des instances contraintes de $C \parallel \phi$ est constante-active ;

(ii) toute instance contrainte de $C \parallel \phi$ est constante-active.

Lemme 5.5. *Soient A, B deux clauses constante-inactives. Alors pour toute paire de constantes c, c' :*

(i) si une inférence de $\mathcal{SP}$ avec prémisses $A[c \to c']$ et $B[c \to c']$ et conclusion $C[c \to c']$ a lieu, alors nous pouvons dériver C par une inférence de $\mathcal{SP}$ avec prémisses A, B, et

(ii) si une inférence de $\mathcal{SP}$ avec prémisse $A[c \to c']$ et conclusion $C[c \to c']$ a lieu, alors nous pouvons dériver C par une inférence de $\mathcal{SP}$ avec prémisse A.

Démonstration. Nous prouvons (i) pour seulement la règle $\mathsf{Superposition - droite}$, car la preuve pour les règles $\mathsf{Superposition - gauche}$ et $\mathsf{Simplification}$ est très similaire.

Supposons que $A[c \to c']$ est la prémisse gauche de la règle $\mathsf{Superposition - droite}$, i.e. $A \equiv \Gamma \Rightarrow \Delta, l[u'] = r$ et $B \equiv \Pi \Rightarrow \Sigma, u = t$. Puisque A n'est pas constante-active non close, aucune constante n'apparaît dans $l[u'] = r$ et nous avons donc $A[c \to c'] \equiv (\Gamma \Rightarrow \Delta)[c \to c'], l[u'] = r$. De façon similaire $B[c \to c'] \equiv (\Pi \Rightarrow \Sigma)[c \to c'], u = t$. Une superposition entre $A[c \to c']$ et $B[c \to c']$ donne $\sigma((\Gamma, \Pi \Rightarrow \Delta, \Sigma)[c \to c'], l[t] = r)$, où σ est l'unificateur principal de u et u'. Or une superposition entre $\Gamma \Rightarrow \Delta, l[u'] = r$

et $\Pi \Rightarrow \Sigma, u = t$ donne $\sigma(\Gamma, \Pi \Rightarrow \Delta, \Sigma, l[t] = r)$. Il est facile de voir que $\sigma((\Gamma, \Pi \Rightarrow \Delta, \Sigma)[c \to c'], l[t] = r)$ est équivalent à $\sigma(\Gamma, \Pi \Rightarrow \Delta, \Sigma, l[t] = r)[c \to c']$.

Pour (ii), considérons les règles Reflection et Factorisation.

- Pour Reflection, soit $A \equiv \Gamma, u' = u \Rightarrow \Delta$. Puisque A n'est pas constante-active non close, aucune constante apparaît dans $u' = u$. Par conséquent, $A[c \to c'] \equiv \Gamma[c \to c'], u' = u \Rightarrow \Delta[c \to c']$ et une réflection avec prémisse $\Gamma[c \to c'], u' = u \Rightarrow \Delta[c \to c']$ donne $\sigma((\Gamma \Rightarrow \Delta)[c \to c'])$ qui est équivalent à $\sigma((\Gamma \Rightarrow \Delta))[c \to c']$. Ce qui veut dire qu'une réflection avec prémisse $\Gamma, u' = u \Rightarrow \Delta$ donne $\Gamma \Rightarrow \Delta$.
- Pour Factorisation, soit $A \equiv \Gamma \Rightarrow \Delta, u = t, u' = t'$. Puisque A n'est pas constante-active non close, aucune constante n'apparaît dans u et t et donc $A[c \to c'] \equiv \Gamma[c \to c'] \Rightarrow \Delta[c \to c'], u = t, u' = t'[c \to c']$. Une factorisation avec prémisse $\Gamma[c \to c'] \Rightarrow \Delta[c \to c'], u = t, u' = t'[c \to c']$ donne $\sigma(\Gamma[c \to c'], t = (t'[c \to c']) \Rightarrow \Delta[c \to c'], u = (t'[c \to c']))$, où σ est l'unificateur principal de u et $(u'[c \to c'])$. Considérons les cas suivants.
  - Si u', t' ne contiennent pas de constante alors $\sigma(\Gamma[c \to c'], t = (t'[c \to c']) \Rightarrow \Delta[c \to c'], u = (t'[c \to c']))$ est équivalent à $\sigma(\Gamma, t = t' \Rightarrow \Delta, u = t')[c \to c']$. Mais une factorisation de $\Gamma \Rightarrow \Delta, u = t, u' = t'$ donne $\sigma(\Gamma, t = t' \Rightarrow \Delta, u = t')$.
  - Si t' contient une constante alors A est constante-active non close car $t = t'$ sera maximal et nous avons donc une contradiction avec l'hypothèse du lemme. t' ne doit donc pas contenir de constante.
  - Maintenant considérons le cas où u' contient une constante c. Puisque u ne contient pas de constante, si σ est l'unificateur principal de u et $(u'[c \to c'])$ alors $\sigma' \equiv \sigma[c' \to c]$ est l'unificateur principal de u et u'. Donc une factorisation sur $\Gamma \Rightarrow \Delta, u = t, u' = t'$ donne $\sigma'(\Gamma, t = t' \Rightarrow \Delta, u = t')$. De plus $\sigma'(\Gamma, t = t' \Rightarrow \Delta, u = t')[c \to c']$ est équivalent à $\sigma(\Gamma[c \to c'], t = (t'[c \to c']) \Rightarrow \Delta[c \to c'], u = (t'[c \to c']))$.

□

Lemme 5.6. *Soit T une théorie axiomatisée par un ensemble fini $Ax(T)$ de clauses. Supposons que pour tout ensemble de littéraux plats et clos, une saturation de $Ax(T) \cup S$ par $\mathcal{SP}$ ne génère pas de clause constante-active non close. Alors pour toute égalité élémentaire $c = c'$, les propriétés suivantes sont équivalentes :*

(i) $T \cup S \cup \{c = c'\}$ est satisfiable,

(ii) $S_g \cup \{c = c'\}$ est satisfiable, où S_g est le sous-ensemble de la saturation de $Ax(T) \cup S$ par $\mathcal{SP}$, qui contient tous les clauses closes.

Démonstration. (i) ⇒ (ii) est trivial. Nous allons prouver (ii) ⇒ (i) par contradiction. Soit S' la saturation de $Ax(T) \cup S$ par $\mathcal{SP}$. Puisque $Ax(T) \cup S$ et S' sont équivalents, $T \cup S \cup \{c = c'\}$ est insatisfiable si et seulement si $S' \cup \{c = c'\}$ l'est aussi. Par la complétude réfutationnelle du Calcul de Superposition, on peut dériver la clause vide à partir de $S' \cup \{c = c'\}$ en utilisant le système d'inférence $\mathcal{SP}$. Or Orientation est prioritaire. Ce qui signifie que l'on peut dériver la clause vide à partir de $S'[c \to c']$ utilisant le système d'inférence $\mathcal{SP}$. Puisque la clause vide est toujours obtenue par une dérivation se terminant par une application de Reflection qui doit utiliser une diségalités $(s \neq t)[c \to c']$ pour dériver la clause vide. Considérons les différents cas possibles.

- Si $(s \neq t)[c \to c']$ est dérivée en utilisant seulement des clauses closes, alors cette dernière doit être close. Ce qui signifie que la clause vide est dérivée en utilisant les clauses closes et le lemme est prouvé.
- Si $(s \neq t)[c \to c']$ est dérivé en utilisant une clause non close. Puisque toute saturation de $Ax(T) \cup S$ par $\mathcal{SP}$ ne génère pas de clause constante-active non close, par le Lemme 5.5, on peut dériver $s \neq t$ à partir de S'. En conséquence $s \neq t$ et $c = c'$ permet de dériver la clause vide. Considérons les sous-cas suivants.
  - Si $s \neq t$ est clos, alors le lemme est prouvé car $s \neq t$ et $c = c'$ sont clos.
  - Dans le cas où $s \neq t$ est non clos, si une Orientation sur $s \neq t$ utilise $c = c'$, alors $s \neq t$ est constante-active non close. Nous avons donc une contradiction avec l'hypothèse du lemme. Si Orientation ne s'applique pas sur $s \neq t$ en utilisant $c = c'$, alors seule $s \neq t$ permet de dériver la clause vide. Ce qui contredit l'hypothèse de satisfiabilité de S'.

□

Le Lemme 5.6 nous permet de développer une méthode automatique afin d'identifier les théories pour lesquelles la correction de la coopération entre la clôture de congruence et la saturation est assurée.

Théorème 5.5. *Soit T une théorie telle que*
- *T est axiomatisée par un ensemble fini $Ax(T)$ de clauses, qui est saturé par rapport à $\mathcal{SP}$, et*
- *la saturation de $Ax(T) \cup G_0^T$ par $m\mathcal{SP}$ ne contient pas de clause constante-active non close.*

Soient S un ensemble T-satisfiable de littéraux plats et clos et S' une saturation de $Ax(T) \cup S$ par $\mathcal{SP}$. Si $c = c'$ est une égalité élémentaire, alors les propriétés suivantes sont équivalentes :

(i) $T \cup S \cup \{c = c'\}$ *est satisfiable,*

(ii) $S_g \cup \{c = c'\}$ *est satisfiable, où S_g est le sous-ensemble de S', qui contient tous les clauses closes.*

Démonstration. Par le Théorème 5.1, si la saturation de $Ax(T) \cup G_0^T$ par $m\mathcal{SP}$ ne contient pas de clause constante-active non close alors toute saturation de $Ax(T) \cup S$ par $\mathcal{SP}$ ne contient pas de clause constante-active non close. Par le Lemme 5.6 nous pouvons déduire le résultat. □

Une version multi-sortée du théorème peut être facilement obtenue en généralisant le concept de clause constante-active au cadre multi-sorté et ainsi nous pouvons généraliser le résultat obtenu pour la théorie des tableaux.

5.6 Propriété de saturation finie pour l'union de théories

Dans cette partie, nous étudions la dérivation de procédures de satisfiabilité pour l'union de théories par saturation. Pour cela, nous considérons la modularité de la propriété de saturation finie. Nous allons d'abord étudier les conditions pour lesquelles l'union $T_1 \cup T_2$ des théories admet une procédure de satisfiabilité basée sur la saturation sachant que les théories T_1 et T_2 admettent de telles procédures. Dans cette optique,

nous considérons la terminaison d'une saturation de l'union $Ax(T_1) \cup Ax(T_2)$ des axiomes des théories avec un ensemble arbitraire S de littéraux plats et clos. Il est évident que si durant une saturation de $Ax(T_1) \cup Ax(T_2) \cup S$, les inférences croisées entre les théories (inférences utilisant des clauses provenant de différentes théories) ne génèrent qu'un nombre fini de clauses, alors cette saturation ne génère qu'un nombre fini de clauses car toute saturation de $Ax(T_i) \cup S$ est finie pour chaque ensemble arbitraire S de littéraux plats et clos. Puisque la signature des théories ne partage pas de symboles de fonction, les inférences croisées entre les théories ne se produisent qu'à travers les variables ou constantes. Il est facile de voir que les inférences aux constantes ne génèrent qu'un nombre fini de clauses. Il en résulte que si on arrive à exclure les inférences croisées entre les théories aux positions de variables, alors la saturation de l'union des axiomes des théories avec un ensemble arbitraire de littéraux plats et clos sera finie, comme souhaité.

Lemme 5.7. *Soit T_i une théorie telle que*
- *T_i est axiomatisée par un ensemble fini $Ax(T_i)$ de clauses, et*
- *T_i a la propriété de saturation finie, et*
- *T_i a la propriété de variable-inactivité,*

pour $i = 1, 2$. Si les signatures de T_1 et T_2 sont disjointes, alors $T_1 \cup T_2$ a la propriété de saturation finie.

Démonstration. Nous considérons toutes les inférences croisées entre les théories et montrons qu'elles ne génèrent qu'un nombre fini de clauses pendant une saturation de $Ax(T_1) \cup Ax(T_2) \cup S$ pour tout ensemble S de littéraux plats et clos. Puisque les signatures des théories sont disjointes, les inférences croisées entre les théories ne se produisent qu'à travers les constantes, les variables ou les symboles de fonction non interprétés. Nous considérons cas par cas ces trois types d'inférences :

- Inférences avec variables : supposons qu'il y a une inférence avec comme prémisses les deux clauses $X = t \vee D$ et $u[s] = v \vee D'$ et comme conclusion $\lambda(u[t] \vee D \vee D')$, où $\lambda = \{X \to s\}$. Si $X \in Var(t)$ alors $\lambda(t) \succ \lambda(X)$ pour toute substitution λ par propriété de sous-terme et donc il n'y aurait pas cette inférence. Si X apparaît comme un sous-terme strict dans D, alors $\lambda(X)$ n'est pas maximal dans $\lambda(X = t \vee D)$ et l'inférence n'a pas lieu. Par conséquent, l'inférence est possible seulement si X n'apparaît pas comme un sous-terme strict dans $X = t \vee D$. Autrement dit, $X = t \vee D$ est variable-active. Ce qui contredit l'hypothèse du lemme.
- Inférences avec constantes : supposons qu'il y a une inférence avec prémisses les deux clauses $c = c' \vee D$ et $u[c] = v \vee D'$ et la conclusion $u[c'] \vee D \vee D'$. Mais il existe un nombre fini de constantes dans $Ax(T_1) \cup Ax(T_2) \cup S$ et donc les inférences de ce type ne génèrent qu'un nombre fini de clauses.

D'après l'hypothèse du lemme, les inférences au sein d'une théorie ne génèrent qu'un nombre fini de clauses. Nous pouvons alors conclure que toute saturation de $Ax(T_1) \cup Ax(T_2) \cup S$ est finie pour tout ensemble S de littéraux plats et clos. □

Lemme 5.8. *Soit T_i une théorie axiomatisée par un ensemble fini $Ax(T_i)$ de clauses, et qui satisfait la propriété de variable-inactivité, pour $i = 1, 2$. Alors $T_1 \cup T_2$ a la propriété de variable-inactivité.*

Démonstration. En procédant de façon similaire au Lemme 5.7, nous pouvons montrer que les inférences croisées entre théories ne génèrent pas de clause variable-active. □

Les Lemmes 5.7 et 5.8 nous permettent de conclure le résultat de modularité suivant.

Théorème 5.6. *La classe des théories axiomatisées par un ensemble fini de clauses, satisfaisant la propriété de saturation finie et la propriété de variable-inactivité est stable par union disjointe.*

Nous avons étudié la terminaison de la saturation de l'union des ensembles des axiomes de deux théories T_1, T_2 avec un ensemble arbitraire de littéraux plats et clos sachant que la saturation de l'ensemble des axiomes de chacune des théories avec un ensemble arbitraire de littéraux plats et clos termine toujours. En introduisant la propriété de variable-inactivité, nous avons défini un critère permettant d'identifier une classe de théories pour laquelle la propriété de saturation finie est modulaire. Or le Théorème 5.2 nous dit qu'il est possible de déterminer par méta-saturation si une théorie a la propriété de variable-inactivité. De ce fait, nous avons une méthode automatique pour déterminer la modularité de la propriété de saturation finie pour l'union de théories par méta-saturation.

Corollaire 5.3. *Soit T_i une théorie telle que*
- *T_i est axiomatisée par un ensemble fini $Ax(T_i)$ de clauses, qui est saturé par rapport à $\mathcal{SP}$, et*
- *T_i a la propriété de méta-saturation finie, et*
- *la saturation de $Ax(T_i) \cup G_0^{T_i}$ par $m\mathcal{SP}$ ne contient pas de clause variable-active,*

pour $i = 1, 2$. Si les signatures de T_1 et T_2 sont disjointes, alors $\mathcal{SP}$ est une procédure de satisfiabilité pour $T_1 \cup T_2$.

Il est intéressant de noter que si une théorie de Horn a la propriété de variable-inactivité et la propriété de saturation finie alors elle a la propriété de complétude vis-à-vis de la déduction. En conséquence, nous pouvons conclure le résultat suivant

Corollaire 5.4. *Soit T_i une théorie telle que*
- *T_i est axiomatisée par un ensemble fini $Ax(T_i)$ de clauses de Horn, qui est saturé par rapport à $\mathcal{SP}$, et*
- *T_i a la propriété de méta-saturation finie, et*
- *la saturation de $Ax(T_i) \cup G_0^{T_i}$ par $m\mathcal{SP}$ ne contient pas de clause variable-active,*

pour $i = 1, 2$. Si les signatures de T_1 et T_2 sont disjointes, alors $\mathcal{SP}$ est une procédure de satisfiabilité complète vis-à-vis de la déduction pour $T_1 \cup T_2$.

5.7 Étendre la procédure à la saturation avec fragmentation

Pour étudier les propriétés des théories non Horn, nous devons considérer, comme dans le chapitre précédent, la propriété de saturation finie pour $\mathcal{SP}'$, i.e. la terminaison d'une saturation par $\mathcal{SP}'$ d'un ensemble fini de littéraux plats et clos avec les axiomes d'une théorie. Pour cela, nous définissons un système d'inférence $m\mathcal{SP}'$, qui est $m\mathcal{SP}$ enrichi par les deux règles de clôture suivantes.

Fragmentation

$$\frac{S' \cup \{A \vee B \parallel \phi\}}{S' \cup \{A \vee p \parallel \phi, \neg p \vee B \parallel \phi\}} \text{ si } Var(A) \cap Var(B) = \emptyset$$

Resolution

$$\frac{A \vee p \parallel \phi \quad \neg p \vee B \parallel \varphi}{A \vee B \parallel \phi \wedge \varphi} \text{ si } \begin{cases} \forall L \in A : p \not\preceq L \\ \forall L \in B : \neg p \not\preceq L \end{cases}$$

5.7.1 Propriété de méta-saturation finie

La propriété de méta-saturation finie s'étend sans le moindre problème de $m\mathcal{SP}$ à $m\mathcal{SP}'$ et on dit qu'une théorie a la propriété de méta-saturation finie par rapport à $m\mathcal{SP}'$.

Lemme 5.9. *Soit T une théorie axiomatisée par un ensemble fini $Ax(T)$ de clauses, qui est saturé par rapport à $\mathcal{SP}'$. Soit G^T_∞ une saturation de $Ax(T) \cup G^T_0$ par $m\mathcal{SP}'$. Alors, pour tout ensemble S de Σ_T-littéraux plats et clos, chaque clause dans la saturation de $Ax(T) \cup S$ par $\mathcal{SP}'$ a l'une des formes suivantes :*

(A) une clause de la forme $C \vee [\neg]p_1 \vee \ldots \vee [\neg]p_n$ tel que C est une instance contrainte d'une clause C' dans G^T_∞ ;

(B) $[\neg]p_1 \vee \ldots \vee [\neg]p_n$;

où $c, c', c_1, \ldots, c_m$ sont des constantes, $p_1, \ldots, p_n$ sont des constantes propositionnelles, $n \geq 0$, $m \geq 1$, $f \notin \Sigma_T$ et $[\neg]$ signifie que $\neg$ est optionnel.

Démonstration. La preuve est faite par induction sur la longueur d'une dérivation par $\mathcal{SP}'$. Le cas de base est trivial car $Ax(T) \cup S$ contient bien des clauses de formes listées. Pour le cas d'induction, nous devons montrer que :

1. Toute clause ajoutée à la saturation de $Ax(T) \cup S$ par une règle de clôture de $\mathcal{SP}'$ est une instance contrainte de l'une des formes listées ci-dessus.
2. Aucune instance contrainte d'une clause supprimée de la saturation de $Ax(T) \cup G^T_0$ par une règle de simplification de $m\mathcal{SP}'$ ne se trouve dans la saturation de $Ax(T) \cup S$ par $\mathcal{SP}'$.

Pour le Cas 1, nous considérons seulement la règle Superposition − droite car la preuve pour les autres règles se fait de façon similaire. Nous considérons les cas correspondant aux inférences entre clauses de mêmes ou différentes formes :

- Superposition − droite entre clauses de forme (A) : supposons que les prémisses sont $l[u'] = r \vee C_1 \vee [\neg]p_1 \vee \ldots \vee [\neg]p_n$ et $u = t \vee C_2 \vee [\neg]p'_1 \vee \ldots \vee [\neg]p'_m$, où $l[u'] = r \vee C_1$ (respectivement $u = t \vee C_2$) est une instance contrainte d'une clause C'_1 (respectivement C'_2) dans la saturation de $Ax(T) \cup G^T_0$ par $m\mathcal{SP}'$. Alors il existe une substitution λ telle que $\lambda(C'_1) \equiv l[u'] = r \vee C_1$ et $\lambda(C'_2) \equiv u = t \vee C_2$ et il existe une Superposition − droite de $m\mathcal{SP}'$ entre C'_1 et C'_2 pour dériver une clause C'_3 telle que $\lambda(C'_3) \equiv l[t] = r \vee C_1 \vee C_2$.
- Superposition − droite entre clauses de forme (B) : génère une clause de la forme (B).

Pour le Cas 2, nous prouvons la propriété pour la règle Subsomption et la preuve pour les autres se fait de façon similaire. Supposons qu'il existe une clause $D \vee [\neg]p_1 \vee \ldots \vee [\neg]p_n$ dans la saturation de $Ax(T) \cup S$ par $\mathcal{SP}'$ telle que D est une instance contrainte de la clause supprimée C'. Nous considérons les cas correspondant aux conditions d'application de la règle :

- $C \in Ax(T)$ et il existe une substitution θ telle que $\theta(C) \subseteq C'$: puisque C est dans la saturation de $Ax(T) \cup S$ par $\mathcal{SP}'$ alors il existe une substitution λ' telle que $\lambda'(C) \subseteq D$. Par conséquent D sera supprimée de la saturation de $Ax(T) \cup S$ par $\mathcal{SP}'$ en utilisant Subsomption.
- C, C' sont des renommages l'une de l'autre : cette inférence a pour but de supprimer les duplications.

□

Corollaire 5.5. *Si T est une théorie ayant la propriété de méta-saturation finie par rapport à $m\mathcal{SP}'$ alors T a la propriété de saturation finie par rapport à $\mathcal{SP}'$.*

5.7.2 Complétude vis-à-vis de la déduction

Comme nous avons étendu la signature de départ avec les constantes propositionnelles, il est nécessaire d'adapter la notion de clause variable-active pour prendre en compte cette extension.

Définition 5.10 (Clause quasi-variable-active)**.** *Une clause est quasi-variable-active si elle a la forme $C \vee [\neg]p_1 \vee \ldots \vee [\neg]p_n$, où C est une clause variable-active, $p_1, \ldots, p_n$ sont des constantes propositionnelles, $n \geq 0$ et $[\neg]$ signifie que $\neg$ est optionnel. Une clause contrainte est quasi-variable-active si une de ses instances contraintes est quasi-variable-active.*

Proposition 5.4. *Soit $C \parallel \phi$ une clause contrainte. Alors, les conditions suivantes sont équivalentes :*

(i) une des instances contraintes de $C \parallel \phi$ est quasi-variable-active ;

(ii) toute instance contrainte de $C \parallel \phi$ est quasi-variable-active.

Définition 5.11 (Propriété de quasi-variable-inactivité)**.** *Soit T une théorie axiomatisée par un ensemble fini $Ax(T)$ de clauses. On dit que T a la propriété de quasi-variable-inactivité si pour tout ensemble S de littéraux plats et clos, toute saturation par $\mathcal{SP}'$ de $Ax(T) \cup S$ ne contient pas de clause quasi-variable-active.*

Lemme 5.10. *Soit T une théorie telle que*
- *T est axiomatisée par un ensemble fini $Ax(T)$ de clauses, et*
- *T a la propriété de quasi-variable-inactivité.*

Soient S un ensemble T-satisfiable de littéraux plats et clos et S_{qe} le sous-ensemble de la saturation de $Ax(T) \cup S$ par $\mathcal{SP}'$, contenant toutes les clauses quasi-élémentaires. Alors, pour toute clause élémentaire C nous avons $Ax(T) \cup S \models C$ si et seulement si $S_{qe} \models C$.

Démonstration. Soit S' la saturation de $Ax(T) \cup S$ par $\mathcal{SP}'$. Soit $C \equiv c_1 = c'_1 \vee \ldots c_n = c'_n$ une clause élémentaire. Raisonnons par réfutation, $Ax(\mathcal{A}) \cup S \models c_1 = c'_1 \vee \ldots c_n = c'_n$ si et seulement si $Ax(T) \cup S \cup \{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$ est insatisfiable. Il est donc possible de dériver la clause vide à partir de $Ax(T) \cup S \cup \{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$ par $\mathcal{SP}'$. Il résulte de la Proposition 4.6 qu'il est aussi possible de dériver la clause vide à partir de $S' \cup \{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$ par $\mathcal{SP}'$. Puisque S est T-satisfiable, la clause vide ne peut être dérivée qu'en commençant par une inférence utilisant une clause C' dans S' et une clause dans $\{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$. Analysons une telle inférence, par exemple entre C' et $c_k \neq c'_k$. Si une telle inférence a lieu, C' doit être nécessairement

une clause contenant soit des égalités entre constantes ou variables soit des (négations de) constantes propositionnelles soit les deux en même temps, autrement l'inférence n'est pas possible. C'est-à-dire C' a l'une des formes suivantes :

(a) une clause quasi-élémentaire, ou

(b) une clause quasi-variable-active.

Si une clause de la forme (b) est dans S', alors on est en contradiction avec l'hypothèse du lemme. Par conséquent, C' est une clause quasi-élémentaire et la conclusion de l'inférence entre C' et une clause dans $\{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$ est de la forme $u \neq u' \vee [\neg]p_1 \vee \ldots \vee [\neg]p_m$, où $m \geq 0$. Encore une fois, s'il y a une inférence entre ce type de clause avec une autre clause, alors cette dernière doit être une clause quasi-élémentaire. Cela signifie que le sous-ensemble S_{qe} de S', qui contient toutes les clauses quasi-élémentaires, et $\{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$ suffisent pour dériver la clause vide. Ou de façon équivalente, $S_{qe} \models c_1 = c'_1 \vee \ldots c_n = c'_n$. □

Les Lemmes 4.1 et 5.10 nous permettent de conclure de la complétude vis-à-vis de la déduction d'une théorie (axiomatisée par un ensemble fini de clauses) ayant la propriété de quasi-variable-inactivité.

Théorème 5.7. *Soit T une théorie telle que*
- *T est axiomatisée par un ensemble fini de clauses, et*
- *T a la propriété de saturation finie par rapport à $m\mathcal{SP}'$, et*
- *T a la propriété de quasi-variable-inactivité.*

Alors $Defrag(\mathcal{SP}'())$ est une procédure de satisfiabilité complète vis-à-vis de la déduction pour T.

Nous sommes maintenant prêt à donner une méthode automatique qui détermine la complétude vis-à-vis de la déduction pour les théories non Horn.

Corollaire 5.6. *Soit T une théorie telle que*
- *T est axiomatisée par un ensemble fini de clauses, et*
- *T a la propriété de méta-saturation finie par rapport à $\mathcal{SP}'$, et*
- *la saturation de $Ax(T) \cup G_0^T$ par $m\mathcal{SP}'$ ne contient pas de clause quasi-variable-active.*

Alors $Defrag(\mathcal{SP}'())$ est une procédure de satisfiabilité complète vis-à-vis de la déduction pour T.

5.8 Discussion

Dans ce chapitre, nous avons montré que la méta-saturation peut être utilisée non seulement pour déterminer la terminaison et la complexité des procédures de satisfiabilité dérivées par saturation comme établi dans [LM02] mais aussi fournir une méthode automatique permettant de vérifier : la stable infinité, la propriété de complétude vis-à-vis de la déduction, et la propriété de saturation finie pour l'union de théories.

L'étude de l'intégration de procédures de satisfiabilité dans la combinaison à la Nelson-Oppen nous a conduit à développer une méthode automatique pour déterminer la stable infinité des théories axiomatisées par un ensemble fini de clauses. En utilisant la méta-saturation et le concept de clause variable-active nous avons pu montrer

que si pour une théorie T la méta-saturation termine et ne dérive pas de clause variable-active alors T est stablement infinie. Ce résultat complète celui de [BGN$^+$06], dans lequel les auteurs ont montré que sous certaines hypothèses la saturation génère nécessairement des clauses d'une forme particulière qui contraint la cardinalité de ses modèles. Ces clauses sont un cas particulier de clauses variable-actives.

Pour avoir une intégration efficace de la combinaison à la Nelson-Oppen, il est important que les procédures de satisfiabilité composantes soient complètes vis-à-vis de la déduction. Nous avons également montré dans ce chapitre que le concept de clause variable-active nous permet de développer une méthode automatique pour déterminer si les procédures de satisfiabilité dérivées par saturation sont complètes vis-à-vis de la déduction. Plus précisément, si pour une théorie de Horn T, la méta-saturation termine et ne dérive pas de clause variable-active alors le Calcul de Superposition est une procédure de satisfiabilité complète vis-à-vis de la déduction pour T. Afin d'obtenir la complétude vis-à-vis de la déduction pour les théories non Horn, nous pouvons utiliser une variante du Calcul de Superposition, qui est le Calcul de Superposition augmenté avec une règle d'analyse de cas, pour activer toutes les inférences possibles et ainsi dériver toutes les clauses élémentaires à propager. Il est aussi important que les procédures de satisfiabilité utilisées dans la combinaison aient la capacité de traiter efficacement de nouvelles égalités. La coopération de la clôture de congruence et de la saturation est ainsi proposée pour répondre à ce problème. Nous avons développé dans ce chapitre une méthode automatique, toujours par méta-saturation, afin de vérifier pour une théorie donnée si la correction de la coopération de la clôture de congruence et de la saturation est garantie.

En ce qui concerne la propriété de saturation finie pour l'union de théories, notre travail étend les résultats de [ABRS05, ABRS06]. En effet dans ces travaux, les auteurs ont défini une classe de théories dites variable-inactives de telle sorte que toute saturation des axiomes avec un ensemble arbitraire de littéraux plats et clos ne permet pas d'inférences croisées entre théories autres que les inférences aux constantes ou termes non interprétés. Ainsi nous obtenons la modularité de la terminaison de la saturation. Dans ce travail, le concept de clause variable-active nous permet d'établir un critère syntaxique permettant de montrer que si la méta-saturation d'une théorie termine et ne dérive pas de clause variable-active alors la propriété de saturation finie pour l'union de théories est garantie.

Nous avons vu que le concept de clause variable-active nous permet définir la classe des théories variable-inactives pour laquelle la stable infinité est garantie. Cependant il existe des théories variable-actives qui sont stablement infinies. En effet, prenons l'exemple suivant.

Exemple 5.8. *T est une théorie axiomatisée par l'ensemble*

$$Ax(T) = \{X = f(Y) \vee Y = g(X)\}.$$

Considérons l'ordre $\succ$ défini de sorte que $f \succ g \succ b \succ a$ et les deux substitutions $\lambda_1 = \{X \to a, Y \to g(b)\}$ et $\lambda_2 = \{X \to f(b), Y \to a\}$.

Nous avons $\lambda_1(X = f(Y)) \succ \lambda_1(Y = g(X))$ et $\lambda_2(Y = g(X)) \succ \lambda_2(X = f(Y))$. De ce fait $X = f(Y)$ et $Y = g(X)$ sont tous les deux maximaux. La définition de clause variable-active est vérifiée. Or la théorie est stablement infinie.

Des exemples similaires peuvent être facilement trouvés pour la complétude vis-à-vis de la déduction. Par conséquent la propriété de variable-inactivité n'est pas une

condition nécessaire de la stable infinité et de la complétude vis-à-vis de la déduction. Nous pensons qu'il est possible de remplacer la condition de variable-inactivité par une condition plus faible afin d'obtenir un résultat plus général pour la stable infinité et la complétude vis-à-vis de la déduction.

Chapitre 6

Nelson-Oppen, Shostak vs. Extended canonizer

Préface

Depuis quelques années, certains papiers ont clarifié les subtilités de la combinaison des théories de Shostak en étudiant leurs relations avec les théories de Nelson-Oppen [CLS96, RS01, BDS02, Kap02, Gan02, CK03b, RS02, CK03a, MZ03]. Malheureusement, ces papiers manquent d'uniformité dans la présentation des résultats ce qui peut troubler les non-experts. Par exemple, certains papiers utilisent du pseudo-code pour décrire les algorithmes de combinaison alors que d'autres adoptent un présentation plus abstraite. Les deux approches ont leurs avantages (et désavantages) respectifs : le pseudo-code permet de démarrer plus rapidement une implantation alors qu'un système d'inférence facilite les preuves de correction. La première contribution de ce chapitre est de fournir une synthèse des approches de Nelson-Oppen et Shostak pour la combinaison disjointe en utilisant une approche à base de règles dans laquelle beaucoup de résultats récents sont reformulés et prouvés corrects de façon uniforme, rigoureuse et simple.

Dans notre reconstruction raisonnée, nous procédons comme suit. Tout d'abord nous introduisons les classes de théories pertinentes pour la combinaison, à savoir les théories SH (pour Shostak) et les théories NO (pour Nelson-Oppen), et utilisons le fait que les théories SH sont dans la classe des théories NO. Cette classification nous amène à étudier trois scénarios lorsque deux théories sont combinées : (a) les deux sont des théories NO, (b) les deux sont des théories SH, (c) l'une est une théorie SH et l'autre une théorie NO. Nous formalisons le schéma de combinaison pour chacun des scénarios comme un système d'inférence. Les conditions d'application des règles d'inférence sont dérivées des propriétés des théories composantes. Dans la lignée de certains papiers récents, le schéma de combinaison pour (b) est vu comme un raffinement de celui pour (a). Le système d'inférence formalisant le schéma de combinaison pour (c) peut être obtenu en réutilisant ceux pour (a) et (b) de façcon modulaire et naturelle. Comme remarque finale, on peut mentionner la possibilité de raffiner les systèmes d'inférence présentés ici avec des stratégies, de manière à obtenir une description plus fine, qui puisse parfaitement imiter la procédure de Shostak. Cette possibilité n'est pas approfondie ici, car nous nous intéressons plus à la modularité qu'à l'efficacité.

Notre synthèse des schémas de combinaison sert plusieurs objectifs. Premièrement, même si les résultats ne sont pas nouveaux, nous pensons que les présenter dans un cadre uniforme peut fournir une référence de valeur pour les personnes intéressées par les problèmes de combinaison et plus particulièrement pour les non-experts du domaine. Deuxièmement, cela peut servir comme point de départ pour des recherches supplémentaires. Comme exemple, l'un des problèmes importants en combinant des theories SH est le manque de modularité des solveurs [CK03b] : il n'existe pas de méthode générale pour produire un solveur pour l'union de théories SH à partir des solveurs connus pour les théories composantes. Ce manque de modularité, plus l'observation que la théorie de l'égalité n'est pas une théorie SH, peut nous laisser penser que n'importe quel schéma de combinaison *ad hoc* pour le scénario (c) constitue un compromis raisonnable entre efficacité et généralité : solveurs et canoniseurs pour les théories SH dérivent efficacement de nouvelles égalités et coopèrent à la Nelson-Oppen. Cette solution peut être facilement spécifiée dans le cadre proposé. Toutefois cette solution laisse ouverte la question d'un concept permettant d'avoir un résultat modulaire tout en gardant l'efficacité apportée par les solveurs et les canoniseurs.

En réponse à cette question, nous proposons le concept de *canoniseur étendu* qui constitue la deuxième contribution de ce chapitre.

Intuitivement, un canoniseur étendu nous permet de canoniser des termes par rapport à une théorie et un ensemble donné d'équations T-satisfiable, de telle sorte que le problème du mot uniforme, i.e. $T \models \Gamma \Rightarrow s = t$, se réduit au problème de vérifier l'identité $ecan(\Gamma)(s) = ecan(\Gamma)(t)$, où $ecan(\Gamma)(s)$ et $ecan(\Gamma)(t)$ sont respectivement les "formes canoniques étendues" de s et t. Un concept similaire introduit dans [RS01] pour la théorie de l'égalité et sa combinaison avec une théorie de Shostak est aussi décrit dans une version rigoureuse du schéma de Shostak. Dans [RS02], un tel schéma est généralisé pour considérer la combinaison de la théorie de l'égalité avec un nombre arbitraire de théories de Shostak grâce à une généralisation intéressante du schéma de Shostak ayant seulement besoin de la construction d'un canoniseur pour l'union des théories et des solveurs pour les théories composantes. La différence principale avec notre travail est que le concept de canoniseur étendu introduit dans ce chapitre est *modulaire*, i.e. il existe une procédure qui, étant donnés deux canoniseurs étendus pour deux théories composantes, construit un canoniseur étendu pour leur union. Une autre caractéristique intéressante des canoniseurs étendus est qu'ils peuvent être construits efficacement en réutilisant une panoplie de techniques existantes telles que canoniseurs et solveurs pour les théories SH, ou techniques de réécriture pour les théories n'admettant pas de solveur. Pour résumer, le concept de canoniseur étendu offre un bon compromis entre modularité et possibilité de réutiliser des techniques différentes pour résoudre le problème du mot uniforme dans une interface commune.

6.1 Introduction

Recently, a series of papers [CLS96, RS01, BDS02, Kap02, Gan02, CK03b, RS02, CK03a, MZ03] have clarified the subtle issues of combining *SH* theories by studying their relationships with *NO* theories. Unfortunately, these papers lack uniformity and non-experts may be confused. For example, some works [CLS96, RS01, BDS02, RS02] use pseudo-code to describe the combination algorithms while others [Gan02, CK03b, CK03a, MZ03] adopt a more abstract (rule-based) presentation. There are advantages (and disadvantages) in both approaches : the pseudo-code offers a better starting point for implementation while inference systems make correctness proofs easier. The first contribution of this chapter is to provide a synthesis of Nelson-Oppen and Shostak approaches to disjoint combination by using a rule-based approach in which many recent results are recast and proved correct in a uniform, rigorous, and simple way.

Our rational reconstruction proceeds as follows. First, we recall that *SH* theories are contained in the class of (convex) *NO* theories. According to this abstract classification, three possible scenarios are to be considered when combining two theories : (a) both are *NO* theories (Section 6.3.1), (b) both are *SH* theories (Section 6.3.2), and (c) one is a *SH* and the other is a *NO* theory (Section 6.3.3). We formalize the combination schema for each scenario as an inference system. The applicability conditions of the inference rules are derived from the properties of the theories being combined. Along the lines of [Gan02, MZ03, CK03a], the combination schema for (b) is obtained as a refinement of that for (a). The inference system formalizing the combination schema for (c), already considered in [BDS02], is obtained by modularly reusing those for (a) and (b) in a natural and straightforward way. As a final remark, we mention the possibility of refining the abstract inference systems presented here with strategies as done in [CK03a], so to get a more fine-grained rule-based implementation which mimics a Shostak procedure as described in [RS02]. We do not do this here, since we are interested in modularity rather than efficiency.

Our synthesis of combination schemas serves two purposes. First, although the results are not new, we believe that presenting them in a uniform framework could provide a valuable reference for people interested in combination problems, especially for non-experts of the field. Second, it can serve as the starting point for further investigations. As an example, a problem of great importance when combining *SH* theories is the lack of modularity for solvers [CK03b] : no general method exists to produce a solver for the union of *SH* theories from the solvers of the component theories. This lack of modularity together with the observation that the theory of equality (ubiquitous in virtually any application where combinations of decision procedures are needed) is not a *SH* theory seem to suggest that any *ad hoc* combination schema for scenario (c) constitute a reasonable trade-off between efficiency and generality : solvers and canonizers for *SH* theories efficiently derive new equalities and cooperate in a Nelson-Oppen way. This solution (adopted, for example, in ICS [FORS01]) can be easily specified in the framework proposed in this chapter. In fact, the schema of Section 6.3.1 can be applied to construct a satisfiability procedure for the union of many *NO* theories which can then be used as the component *NO* theory in a simple generalization of the schema in Section 6.3.3 to accommodate several solvers and canonizers. However, this solution leaves open the question about the existence of a suitable concept that would allow us to obtain a modularity result and retain some of the efficiency of the canonizers and solvers.

By investigating this question in our framework, we propose the concept of *extended canonizer* which constitutes the second contribution of this chapter. Intuitively, an extended canonizer allows us to canonize terms with respect to a given theory T and a given T-satisfiable set of equations Γ, so that the uniform word problem for T, i.e. $T \models \Gamma \Rightarrow s = t$, reduces to the problem of checking the identity $ecan(\Gamma)(s) = ecan(\Gamma)(t)$, where $ecan(\Gamma)(s)$ and $ecan(\Gamma)(t)$ are the "extended canonical forms" of s and t, respectively (Section 6.4.1). A similar concept was introduced in [RS01] for the theory of equality and its combination with one Shostak theory is also described by a rigorous version of Shostak schema. In [RS02], such a schema is generalized to consider the combination of the theory of equality with an arbitrary number of *SH* theories by an interesting generalization of Shostak schema requiring only the construction of a canonizer for the union of the theories and invoking the solvers for the constituent theories. The main difference with our work is that the concept of extended canonizer is *modular*, i.e. there exists a procedure that, given two extended canonizers for two component theories, yields an extended canonizer for their union (Section 6.4.2). Another interesting feature of extended canonizers is that they can be *efficiently built* by reusing a wealth of existing techniques such as canonizers and solvers for *SH* theories and rewriting techniques (as advocated in [Kap97, BTV03, ARR03, Mar96]) for theories which do not admit a solver (Section 6.5). To summarize, the concept of extended canonizer offers an interesting trade-off between modularity and the possibility to reuse disparate techniques to solve the uniform word problem under a common interface. As a final remark, we notice that our definition of extended canonizer is orthogonal to the line of research (advocated in [CK03b]) which suggests that modular solvers may exist in modified settings such as multi-sorted logic.

6.2 Background

We assume the usual first-order syntactic notions of signature, term, position, and substitution. Let Σ be a first-order signature containing only function symbols with their arity and $\mathcal{X}$ a set of variables. A 0-ary function symbol is called a *constant*. A Σ-*term* is a first-order term built out of the symbols in Σ and the variables in $\mathcal{X}$. We use the standard notion of substitution. We write substitution applications in postfix notation, e.g. $t\sigma$ for a term t and a substitution σ. The set of variables occurring in a term t is denoted by $Var(t)$. If l and r are two Σ-terms, then $l = r$ is a Σ-*equality* and $\neg(l = r)$ (also written as $l \neq r$) is a Σ-*disequality*. A Σ-literal is either a Σ-equality or a Σ-disequality. A Σ-*formula* is built in the usual way out of the universal and existential quantifiers, Boolean connectives, and symbols in Σ. If φ is a formula, then $Var(\varphi)$ denotes the set of free variables in φ. We call a formula *ground* if it has no variable, and a *sentence* if it has no free variables. Substitution applications are extended to arbitrary first-order formulas, and are written in postfix notation, e.g. $\varphi\sigma$ for a formula φ and a substitution σ.

We also assume the usual first-order notions of interpretation, satisfiability, validity, logical consequence, and theory, as given, e.g., in [End72]. A *first-order theory* is a set of first-order sentences. A Σ-*theory* is a theory all of whose sentences have signature Σ. All the theories we consider are first-order theories *with equality*, which means that the equality symbol $=$ is always interpreted as the identity relation. The

theory of equality is denoted by $\mathcal{E}$. A Σ-structure $\mathcal{A}$ is a model of a Σ-theory T if $\mathcal{A}$ satisfies every sentence in T. A Σ-formula is *satisfiable in* T if it is satisfiable in a model of T. Two Σ-formulas φ and ψ are *equisatisfiable in* T if for every model $\mathcal{A}$ of T, φ is satisfiable in $\mathcal{A}$ iff ψ is satisfiable in $\mathcal{A}$. The *satisfiability problem* for a theory T amounts to establishing whether any given finite quantifier-free conjunction of literals (or equivalently, any given finite set of literals) is T-satisfiable or not. A *satisfiability procedure* for T is any algorithm that solves the satisfiability problem for T.[14] Note that we can use free constants instead of variables to equivalently redefine the satisfiability problem for T as the problem of establishing the consistency of $T \cup S$ for a finite set S of ground literals. The *uniform word problem* for a theory T amounts to establishing whether $T \models \Gamma \Rightarrow e$, where Γ is a conjunction of Σ-equalities, e is a Σ-equality, and all the variables in $\Gamma \Rightarrow e$ are (implicitly) universally quantified.

Given an inference system R composed of inference rules (written as $P \vdash C$), the binary relation $\vdash_{\mathsf{R}}$ is defined on formulas as follows : $\Phi \vdash_{\mathsf{R}} \Phi'$ if Φ' can be derived from Φ by applying a rule in R. The reflexive and transitive closure of $\vdash_{\mathsf{R}}$, denoted by $\vdash_{\mathsf{R}}^*$, is called the *derivation relation* of R. Also, a *derivation* in R is a sequence $\Phi \vdash_{\mathsf{R}} \Phi' \vdash_{\mathsf{R}} \Phi'' \vdash_{\mathsf{R}} \cdots$. A formula Φ is in *normal form w.r.t.* $\vdash_{\mathsf{R}}$ if there is no derivation in R starting from Φ. The relation $\vdash_{\mathsf{R}}^*$ is *terminating* if there is no infinite derivation. We will write the *configuration* Γ, Δ to denote a formula $\Gamma \wedge \Delta$, where Γ is a conjunction of equalities and Δ is a conjunction of disequalities.

6.2.1 Combination of theories

In the sequel, let Σ_1 and Σ_2 be two disjoint signatures (i.e. $\Sigma_1 \cap \Sigma_2 = \emptyset$) and T_i be a Σ_i-theory for $i = 1, 2$. A $\Sigma_1 \cup \Sigma_2$-term t is an *i-term* if it is a variable or it has the form $f(t_1, ..., t_n)$, where f is in Σ_i (for $i = 1, 2$ and $n \geq 0$). Notice that a variable is both a 1-term and a 2-term. A non-variable subterm s of an i-term is *alien* if s is a j-term, and all superterms of s are i-terms, where $i, j \in \{1, 2\}$ and $i \neq j$. An i-term is *i-pure* if it does not contain alien subterms. A literal is i-pure if it contains only i-pure terms. A formula is said to be *pure* if there exists $i \in \{1, 2\}$ such that every term occurring in the formula is i-pure. We will write the *configuration* $\Phi_1; \Phi_2$ to denote a formula $\Phi_1 \wedge \Phi_2$, where Φ_i is a conjunction of i-pure literals ($i = 1, 2$).

In this paper, we shall consider the problem of solving the satisfiability problem for $T_1 \cup T_2$ (i.e. the problem of checking the $T_1 \cup T_2$-satisfiability of conjunctions of $\Sigma_1 \cup \Sigma_2$-literals) by using the satisfiability procedures for T_1 and T_2. For certain theories, more basic algorithms exist which can be used to build satisfiability procedures, e.g. canonizers and solvers for the class of Shostak theories (see below for a formal definition). When such algorithms exist for either T_1, T_2, or both, we are interested in using them to solve the satisfiability problem for $T_1 \cup T_2$. In order to know which basic algorithms are available for T_1 and T_2 and what are the assumptions on T_1 and T_2, the following notions and results are useful.

Remember that a conjunction Γ of Σ-literals is convex in a Σ-theory T iff for any disjunction $\bigvee_{i=1}^{n} x_i = y_i$ (where x_i, y_i are variables and $i = 1, ..., n$) we have that $T \cup \Gamma \models \bigvee_{i=1}^{n} x_i = y_i$ iff $T \cup \Gamma \models x_i = y_i$, for some $i \in \{1, ..., n\}$. A Σ-theory

14. The satisfiability of any quantifier-free formula can be reduced to the satisfiability of sets of literals by converting to disjunctive normal form and then splitting on disjunctions, e.g., checking whether $S_1 \vee S_2$ (where S_1 and S_2 are conjunction of literals) is T-satisfiable reduces to checking the T-satisfiability of both S_1 and S_2.

T is *convex* iff all the conjunctions of Σ-literals are convex. A Σ-theory T is *stably-infinite* iff for any T-satisfiable Σ-formula φ, there exists a model of T whose domain is infinite and which satisfies φ. A *Nelson-Oppen* theory (NO-theory, for short) is a stably-infinite theory which admits a satisfiability algorithm. A NO-convex-theory is a convex NO-theory. The class of NO-theories (resp. NO-convex-theories) is denoted by NO (resp. NO-convex).

Recall that a *solver* (denoted by *solve*) for a Σ-theory T is a function which takes as input a Σ-equality $s = t$ and such that (a) $solve(s = t)$ returns $false$, if $T \models s \neq t$, or (b) $solve(s = t)$ returns a substitution $\lambda = \{x_1 \rightarrow t_1, ..., x_n \rightarrow t_n\}$ such that (b.1) x_i is a variable occurring in s or t for $i = 1, ..., n$, (b.2) x_i does not occur in any t_j for $i, j = 1, ..., n$, and (b.3) $T \models s = t \Leftrightarrow \exists y_1, ..., y_m. \bigwedge_{i=1}^{n} x_i = t_i$, where $y_1, ..., y_m$ ($m \geq 0$) are "fresh" variables s.t. y_k does not occur in s or t, for all $k = 1, ..., m$. A conjunction of Σ-equalities is in *solved form* iff it has the form $\bigwedge_{i=1}^{n} x_i = t_i$, which will be denoted by $\hat{\lambda}$, where $\lambda = \{x_1 \rightarrow t_1, ..., x_n \rightarrow t_n\}$ is the substitution returned by *solve*. A *canonizer canon* for a Σ-theory T is an idempotent function from Σ-terms to Σ-terms such that $T \models a = b$ iff $\models canon(a) = canon(b)$. A *Shostak* theory is a convex theory which admits a solver and a canonizer. A SH-theory is a stably-infinite Shostak theory. The class of SH-theories is denoted by SH.

We assume SH-theories to be stably-infinite since this is necessary to combine them with other theories as suggested by many recent papers (see e.g. [MZ03]). This is not too restrictive since, as shown in [BDS02], any convex theory with no trivial models is stably-infinite.

Proposition 6.1. *$SH \subseteq NO - convex \subseteq NO$.*

6.3 Rational reconstruction of combination schemas

Let T_i be a Σ_i-theory ($i = 1, 2$) such that $\Sigma_1 \cap \Sigma_2 = \emptyset$. We consider the problem of building a satisfiability procedure for $T_1 \cup T_2$. As a preliminary step, we consider a *purification process* converting any conjunction Φ of $\Sigma_1 \cup \Sigma_2$-literals into a conjunction of pure literals. Such a process is achieved by replacing each alien subterm t by a new variable x and adding the equality $x = t$ to Φ. This mechanism, called *variable abstraction*, is repeatedly applied to Φ until no more alien subterms t can be abstracted away. Obviously, the purification process always terminates yielding $\Phi_1 \wedge \Phi_2$, where Φ_i is a conjunction of Σ_i-literals ($i = 1, 2$) such that $\Phi_1 \wedge \Phi_2$ is equisatisfiable to Φ in $T_1 \cup T_2$. In the sequel, without loss of generality, we consider the satisfiability of formulae of the form $\Phi_1 \wedge \Phi_2$ (or, equivalently, of configurations $\Phi_1; \Phi_2$), where Φ_i is a conjunction of i-pure literals.

Our combination schemas are specified by inference systems. To prove that an inference system R yields a satisfiability procedure, we follow a three steps methodology. First, we show that the derivation relation $\vdash_{\mathsf{R}}$ induced by R is terminating. Second, we prove that $\vdash_{\mathsf{R}}$ preserves (un-)satisfiability. Finally, we check that the normal forms defined by $\vdash_{\mathsf{R}}$ (i.e. configurations to which no rule in R can be applied) distinct from $false$ must be satisfiable. The proof of the last step proceeds by contradiction showing that a normal form distinct from $false$ cannot be unsatisfiable by using the following (technical) lemma from which the proof of correctness of Nelson-Oppen schema in [RT03] essentially depends.

Lemma 6.1. *[RT03] If T_1 and T_2 are two signature-disjoint stably-infinite theories, then any conjunction $\Phi_1 \wedge \Phi_2$ of pure quantifier-free formulas is $T_1 \cup T_2$-satisfiable if and only if there exists some equivalence relation E over shared variables in $V = Var(\Phi_1) \cap Var(\Phi_2)$ such that $\Phi_i \cup arr(V, E)$ is T_i-satisfiable for $i = 1, 2$, where*

$$arr(V, E) = \{x = y \mid (x, y) \in E\} \cup \{x \neq y \mid (x, y) \in (V \times V) \setminus E\}.$$

In the remainder, we need Lemma 6.1 (cf. Section 6.3) and the following Lemmas.

Lemma 6.2. *Let $\sigma = \{x_1 \to t_1, ..., x_n \to t_n\}$ be a substitution such that $x_i \notin Var(t_j)$ for all i, j. For every theory T and every conjunction Γ of literals we have :*

$$T \cup \Gamma \cup \{x_1 = t_1, ..., x_n = t_n\} \models x = y \text{ iff } T \cup \Gamma\sigma \models x\sigma = y\sigma$$

Proof. ($\Rightarrow$) Assume that $T \cup \Gamma \cup \{x_1 = t_1, ..., x_n = t_n\} \models x = y$ and $\mathcal{A}$ is a structure satisfying $T \cup \Gamma\sigma$ with a valuation α. By the fact that x_i does not appear in $T \cup \Gamma\sigma$, one can redefine α such that $\alpha(x_i) = \alpha(t_i)$, for all i. Is easy to see that $(\mathcal{A}, \alpha)$ satisfies $T \cup \Gamma \cup \{x_1 = t_1, ..., x_n = t_n\}$. It implies $(\mathcal{A}, \alpha)$ satisfies $x = y$ too, thus $x\sigma = y\sigma$ evaluates to true in $(\mathcal{A}, \alpha)$.

($\Leftarrow$) Assume that $T \cup \Gamma\sigma \models x\sigma = y\sigma$ and $\mathcal{A}$ is a structure satisfying $T \cup \Gamma \cup \{x_1 = t_1, ..., x_n = t_n\}$ with a valuation α. It is straightforward that $(\mathcal{A}, \alpha)$ satisfies $T \cup \Gamma\sigma$. Hence $(\mathcal{A}, \alpha)$ satisfies $x\sigma = y\sigma$. In addition, $\alpha(x_i) = \alpha(t_i)$, for all i, which implies $x = y$ maps to true in $(\mathcal{A}, \alpha)$. □

Lemma 6.3. *Let T be a convex theory, Γ a set of equalities, and Δ a set of disequalities. If $\Gamma \wedge \Delta$ is T-satisfiable, then*

$$T \models (\Gamma \wedge \Delta) \Rightarrow x = y \text{ iff } T \models \Gamma \Rightarrow x = y$$

Proof. ($\Leftarrow$) Trivial. ($\Rightarrow$) Assume that $T \cup \Gamma \cup \Delta \models x = y$. Then $T \cup \Gamma \cup \Delta \cup \{x \neq y\}$ is unsatisfiable. By convexity of T, either $T \cup \Gamma \cup \{x \neq y\}$ is unsatisfiable or there exists a disequality $s \neq t \in \Delta$ such that $T \cup \Gamma \cup \{s \neq t\}$ is unsatisfiable. In the first case, the Lemma is proved. In the second one, by convexity $\Gamma \wedge \Delta$ is T-unsatisfiable, we obtain a contradiction. □

6.3.1 Combining theories in NO-convex

We assume that T_1 and T_2 are in NO-convex. This implies the availability of two satisfiability procedures for T_1 and T_2. We consider the inference system NO obtained as the union of NO_1 presented in Figure 6.1 and NO_2 obtained from NO_1 by symmetry.[15] NO takes configurations of the form $\Phi_1; \Phi_2$ where Φ_i is a set of Σ_i-literals ($i = 1, 2$). Rule $\mathsf{Contradiction}_1$ reports the T_1-unsatisfiability of Φ_1 (and hence of $\Phi_1 \wedge \Phi_2$), detected by the available satisfiability procedure. Rule $\mathsf{Deduction}_1$ propagates equalities between shared variables known to the procedure for T_1 to that for T_2 (if they are not already known to the latter). The problem of checking whether the equality $x = y$ is a logical consequence of $T_1 \cup \Phi_1$ is transformed into the problem of checking the T_1-unsatisfiability of $\Phi_1 \cup \{x \neq y\}$ so to be able to exploit the available satisfiability procedure.

15. A symmetric rule for T_2 is obtained from a rule for T_1 by swapping indexes 1 and 2. A symmetric inference system for T_2 is the set of symmetric rules for T_2 obtained from the rules for T_1.

$\mathsf{Contradiction}_1$

$$\frac{\Phi_1;\Phi_2}{false} \ \textbf{if } \Phi_1;\Phi_2 \text{ is } T_1\text{-unsatisfiable}$$

$\mathsf{Deduction}_1$

$$\frac{\Phi_1;\Phi_2}{\Phi_1;\Phi_2 \cup \{x=y\}} \ \textbf{if} \begin{cases} \Phi_1 \textit{ is } T_1\text{-satisfiable}, \\ \Phi_1 \wedge x \neq y \text{ is } T_1\text{-unsatisfiable}, \\ \Phi_2 \wedge x \neq y \text{ is } T_2\text{-satisfiable, and} \\ x, y \in Var(\Phi_1) \cap Var(\Phi_2) \end{cases}$$

FIGURE 6.1 – The Inference System NO_1

Theorem 6.1. *Let T_1, T_2 be two signature-disjoint NO-convex-theories. Let NO be the inference system defined as the union $\mathsf{NO}_1 \cup \mathsf{NO}_2$, where NO_1 is depicted in Figure 6.1 and NO_2 is obtained from NO_1 by symmetry. The relation $\vdash^*_{\mathsf{NO}}$ is terminating and $\Phi_1;\Phi_2 \vdash^*_{\mathsf{NO}} false$ iff $\Phi_1 \wedge \Phi_2$ is $T_1 \cup T_2$-unsatisfiable.*

Proof. Direct consequence of the three following Lemmas. □

Lemma 6.4 (Termination)**.** *The relation $\vdash^*_{\mathsf{NO}}$ is terminating.*

Proof. If the rule Contradiction applies, the procedure terminates. The rule Deduction decreases strictly the number of equivalence classes of shared variables. □

Lemma 6.5 (Soundness)**.** *The relation $\vdash_{\mathsf{NO}}$ preserves equisatisfiability in $T_1 \cup T_2$.*

Proof. The soundness of the rule Contradiction is straightforward . For Deduction : ($\Rightarrow$) Assume that $\models^{\alpha}_{\mathcal{M}} (\Phi_1 \cup \Phi_2)$. The application conditions guarantees that $T_1 \cup \Phi_1 \models x = y$, implying $\alpha(x) = \alpha(y)$. Thus $\models^{\alpha}_{\mathcal{M}} (\Phi_1 \cup \Phi_2 \cup \{x = y\})$. ($\Leftarrow$) Trivial. □

Lemma 6.6 (Completeness)**.** *If $\Phi_1;\Phi_2$ is a normal form wrt. $\vdash_{\mathsf{NO}}$ different from $false$, then $\Phi_1 \wedge \Phi_2$ is $T_1 \cup T_2$-satisfiable.*

Proof. If the procedure terminates without reporting $false$, the final configuration must be of the form $\Phi_1;\Phi_2$ such that :

- Φ_i is T_i-satisfiable for $i = 1, 2$ (otherwise Contradiction applies),
- $\forall x, y \in Var(\Phi_1) \cap Var(\Phi_2),\ T_1 \models \Phi_1 \Rightarrow x = y$ iff $T_2 \models \Phi_2 \Rightarrow x = y$ (otherwise Deduction applies).

Assume $\Phi_1 \wedge \Phi_2$ is $T_1 \cup T_2$-unsatisfiable. Consider the equivalence relation ξ over variables in $Var(\Phi_1) \cap Var(\Phi_2)$ such that $(x, y) \in \xi$ iff $T_1 \models \Phi_1 \Rightarrow x = y$ and $T_2 \models \Phi_2 \Rightarrow x = y$.

By Lemma 6.1, $\Phi_1 \wedge \Phi_2$ $T_1 \cup T_2$-unsatisfiable implies that there exists $i \in \{1, 2\}$ such that $\Phi_i \cup arr(V, \xi)$ is T_i-unsatisfiable. Then two cases must be distinguished :

- $arr(V, \xi)$ contains at least a disquality. By convexity there exist some i and some disequality $x_k \neq y_k$ such that $\Phi_i \cup \xi \cup \{x_k \neq y_k\}$ is T_i-unsatisfiable, or equivalently $T - i \cup \Phi_i \cup \xi \models x_k = y_k$. There by $x_k \neq y_k$ should not be in $arr(V, \xi)$, a contradiction.

$\mathsf{Solve-fail_1}$

$$\frac{\Gamma_1, \Delta_1; \Gamma_2, \Delta_2}{false} \quad \textbf{if } solve_1(\Gamma_1) = false$$

$\mathsf{Solve-success_1}$

$$\frac{\Gamma_1, \Delta_1; \Gamma_2, \Delta_2}{\widehat{\gamma_1}, \Delta_1; \Gamma_2, \Delta_2} \quad \textbf{if } \begin{cases} \Gamma_1 \text{ is not in solved form,} \\ \gamma_1 = solve_1(\Gamma_1) \neq false \end{cases}$$

$\mathsf{Contradiction_1}$

$$\frac{\widehat{\gamma_1}, \Delta_1 \cup \{s \neq t\}; \Gamma_2, \Delta_2}{false} \quad \textbf{if } canon_1(s\gamma_1) = canon_1(t\gamma_1)$$

$\mathsf{Deduction_1}$

$$\frac{\widehat{\gamma_1}, \Delta_1; \widehat{\gamma_2}, \Delta_2}{\widehat{\gamma_1}, \Delta_1; \widehat{\gamma_2} \cup \{x = y\}, \Delta_2}$$

$$\textbf{if } \begin{cases} canon_1(x\gamma_1) = canon_1(y\gamma_1), \\ canon_2(x\gamma_2) \neq canon_2(y\gamma_2), \\ x, y \in Var(\gamma_1) \cap Var(\gamma_2) \end{cases}$$

FIGURE 6.2 – The Inference System $\mathsf{SH_1}$

– $arr(V, \xi)$ contains no disqualities. Then $\Phi_i \cup arr(V, \xi)$ is equivalent to $\Phi_i \cup \xi$ which is T_i-equivalent to Φ_i. Thus Φ_i is T_i-unsatisfiable, a contradiction again. □

Indeed, NO specifies only the essence of the Nelson-Oppen schema. Such a schema can be refined to increase efficiency. For example, the satisfiability procedures of some theories, such as Linear Arithmetic, can be extended so to derive entailed equalities while checking for satisfiability (see, e.g. [KN94, vHG92]) thereby avoiding the guessing done when applying $\mathsf{Deduction_1}$. In this chapter, we will not consider this kind of amelioration (the interested reader is referred to [DNS03] for a comprehensive guideline to the efficient implementation of the Nelson-Oppen schema). In the following, we will consider refinements of NO which allow us to incorporate solvers and canonizers for theories in SH.

6.3.2 Combining theories in SH

We assume that T_1 and T_2 are in SH. This implies the availability of a canonizer $canon_i$ and a solver $solve_i$ for each theory T_i $(i = 1, 2)$.

Preliminary to the combination schema, we extend solvers to handle sets of equalities as follows : $solve(\emptyset)$ returns the identity substitution ϵ ; $solve(\Gamma \cup \{s = t\}) = false$, if $solve(s = t) = false$; and $solve(\Gamma \cup \{s = t\}) = \sigma \circ solve(\Gamma\sigma)$, if $solve(s = t) = \sigma$, where $\circ$ denotes composition of substitutions.

We consider the inference system SH obtained as the union of $\mathsf{SH_1}$ presented in Figure 6.2 and $\mathsf{SH_2}$ obtained from $\mathsf{SH_1}$ by symmetry. SH takes configurations of the form $\Gamma_1, \Delta_1; \Gamma_2, \Delta_2$, where Γ_i is a set of Σ_i-equalities and Δ_i is a set of Σ_i-

disequalities for $i = 1, 2$. Rule $\mathsf{Solve} - \mathsf{fail}_1$ reports the T_1-unsatisfiability of Γ_1 (and hence of $\Gamma_1 \wedge \Delta_1 \wedge \Gamma_2 \wedge \Delta_2$) detected by $solve_1$. Rule $\mathsf{Solve} - \mathsf{success}_1$ replaces the Σ_1-equalities Γ_1 with their solved form which is obtained again by using $solve_1$. This is important for the next two rules. Dealing with solved forms allows us to simply determine entailed equalities (possibly between shared variables, see $\mathsf{Deduction}_1$) using canonizers. Hence, it is possible to lazily report unsatisfiability as soon as we find a disequality whose corresponding equality is entailed (see $\mathsf{Contradiction}_1$). Indeed, convexity allows us to handle disequalities one by one.

Theorem 6.2. *Let T_1, T_2 be two signature-disjoint SH-theories. Let SH be the inference system defined as the union $\mathsf{SH}_1 \cup \mathsf{SH}_2$, where SH_1 is depicted in Figure 6.2 and SH_2 is obtained by symmetry. The relation $\vdash^*_{\mathsf{SH}}$ is terminating and $\Gamma_1, \Delta_1; \Gamma_2, \Delta_2 \vdash^*_{\mathsf{SH}} false$ iff $\Gamma_1 \wedge \Delta_1 \wedge \Gamma_2 \wedge \Delta_2$ is $T_1 \cup T_2$-unsatisfiable.*

Proof. Direct consequence of the three following Lemmas. □

Lemma 6.7 (Termination)**.** *The relation $\vdash^*_{\mathsf{SH}}$ is terminating.*

Proof. If rules Solve-fail and Contradiction apply, the procedure terminates. For other rules, we must show that rules Solve-success and Deduction can not be applied infinitely. Indeed, Solve-success can only be applied if Γ_1 is not in solved form. Only Deduction may likely modify a solved form into a non-solved form by integrating an equality between variables. We only need to prove that Deduction can not be applied infinitely. The set of shared variables is finite and Deduction integrates equalities detected in T_1 to T_2 only if they have not been detected in T_2 ($canon_1(x\lambda_1) = canon_1(y\lambda_1)$ and $canon_2(x\lambda_2) \neq canon_2(y\lambda_2)$). This guarantees that Deduction can only be applied finitely many times. □

Lemma 6.8 (Soundness)**.** *The relation $\vdash_{\mathsf{SH}}$ preserves equisatisfiability in $T_1 \cup T_2$.*

Proof. The soundness is straightforward for rules Solve-fail and Contradiction.

- For Solve-success :
  - ($\Rightarrow$) Assume that $\models^{\alpha}_{\mathcal{M}} \Gamma_1 \wedge \Delta_1 \wedge \Gamma_2 \wedge \Delta_2$. Let $\lambda_1 = \{x_1 \to t_1, ..., x_n \to t_n\} = solve(\Gamma_1)$. By definition of $solve$, the equivalence

$$\Gamma_1 \Leftrightarrow \exists \tilde{y}[\bigwedge_{i=1}^{n} x_i = t_i]$$

    is $T_1 \cup T_2$-valid. We can extend α to $\tilde{y}$ such that $\alpha(x_i) = \alpha(t_i)$. It is easy to see that $\models^{\alpha}_{\mathcal{M}} \widehat{\lambda_1} \wedge \Delta_1 \wedge \Gamma_2 \wedge \Delta_2$.
  - ($\Leftarrow$) Assume that $\models^{\alpha}_{\mathcal{M}} \widehat{\lambda_1} \wedge \Delta_1 \wedge \Gamma_2 \wedge \Delta_2$, where $\lambda_1 = solve(\Gamma_1)$. Like above, one can restrict α to $Dom(\alpha) - \tilde{y}$. Clearly, we have $\models^{\alpha}_{\mathcal{M}} \Gamma_1 \wedge \Delta_1 \wedge \Gamma_2 \wedge \Delta_2$.
- For Deduction :
  - ($\Rightarrow$) Assume that $\models^{\alpha}_{\mathcal{M}} \widehat{\lambda_1} \wedge \Delta_1 \wedge \widehat{\lambda_2} \wedge \Delta_2$. We have $\alpha(x) = \alpha(x\lambda_1)$ and $\alpha(y) = \alpha(y\lambda_1)$. By Lemma 6.2, $T_1 \cup \widehat{\lambda_1} \models x = y$ iff $T_1 \models x\lambda_1 = y\lambda_1$. But then $T_1 \models x\lambda_1 = y\lambda_1$ implies $T_1 \cup T_2 \models x\lambda_1 = y\lambda_1$, thus $\alpha(x\lambda_1) = \alpha(y\lambda_1)$. Hence $\alpha(x) = \alpha(y)$ and $\models^{\alpha}_{\mathcal{M}} \widehat{\lambda_1} \wedge \Delta_1 \wedge \widehat{\lambda_2} \wedge x = y \wedge \Delta_2$.
  - ($\Leftarrow$) Trivial. □

Lemma 6.9 (Completeness)**.** *$\Gamma_1, \Delta_1; \Gamma_2, \Delta_2$ is a normal form wrt. $\vdash_{\mathsf{SH}}$ different from $false$, then $\Gamma_1 \wedge \Delta_1 \wedge \Gamma_2 \wedge \Delta_2$ is $T_1 \cup T_2$-satisfiable.*

Proof. If the procedure terminates without reporting $false$, the final configuration must be of the form $(\hat{\lambda}_1, \Delta_1); (\hat{\lambda}_2, \Delta_2)$ such that :

- $\hat{\lambda}_i \wedge \Delta_i$ is T_i-satisfiable for $i = 1, 2$ (otherwise Contradiction applies),
- $\forall x, y \in Var(\Phi_1) \cap Var(\Phi_2)$, $canon_1(x\lambda_1) = canon_1(y\lambda_1)$ iff $canon_2(x\lambda_2) = canon_2(y\lambda_2)$ (otherwise Deduction applies).

Since $\hat{\lambda}_i \wedge \Delta_i$ is T_i-satisfiable we have

$$T_i \models (\hat{\lambda}_i \wedge \Delta_i) \Rightarrow x = y \text{ iff } canon_i(x\lambda_i) = canon_i(y\lambda_i)$$

thanks to Lemma 6.3, Lemma 6.2, and the definition of *canon*. Then the proof can be continued by contradiction in a way similar to the proof of Lemma 6.6. □

It is easy to see that a strategy applying rules $\mathsf{Solve-fail}_1$, $\mathsf{Solve-success}_1$, and $\mathsf{Contradiction}_1$ in SH to a configuration $\Gamma_1, \Delta_1; \Gamma_2, \Delta_2$ yields the same result as that of applying rule $\mathsf{Contradiction}_1$ in NO to $\Gamma_1 \cup \Delta_1; \Gamma_2 \cup \Delta_2$. Similarly, the application of rules $\mathsf{Solve-success}_1$ and $\mathsf{Deduction}_1$ in SH simulates the application of $\mathsf{Deduction}_1$ in NO ; showing that equalities between shared variables can be derived by invoking a solver (and a canonizer) rather than resorting to guessing as for NO when applying the rule $\mathsf{Deduction}_i$ $(i = 1, 2)$. This is one of the key insights underlying Shostak schema.

Furthermore, similarly to [Gan02], the abstract schema presented here seems to emphasize the importance of the solver w.r.t. the canonizer. In fact, if the solved form returned by the solver is also canonical, the canonizer can be trivially implemented as the identity function. Nonetheless, we believe that the concept of canonizer is quite important for two crucial reasons. First, it offers the entry point to refinements of the proposed schema to increase efficiency. In fact, solving a set of equalities in "one-shot", as done when applying rule $\mathsf{Solve-success}_1$, may not be as efficient as solving equalities incrementally, as done e.g. in [RS01, Kap02]. This can be incorporated in our schema by refining the inference system SH along the lines described in [CK03a] so that the solver is applied to only one equality at a time and the canonizer needs to return a canonical form for arbitrary terms. The second reason is that a generalization of the concept of canonizer will be the basis for a new combination schema as we will see in Section 6.4.

6.3.3 Combining a theory in NO-convex with one in SH

Without loss of generality, let us assume that T_1 is in NO-convex and that T_2 is in SH. This situation frequently arises in practical verification problem, e.g. the union of a theory in SH and $\mathcal{E}$ (which is *not* in SH). We consider the inference system NS obtained as the union of NO_1 in Figure 6.1 and SH_2, the symmetric of SH_1 in Figure 6.2. NS takes configurations of the form $\Phi_1; \Gamma_2, \Delta_2$ where Φ_1 is a set of Σ_1-literals, Γ_2 is a set of Σ_2-equalities, and Δ_2 is a set of Σ_2-disequalities. We furtherly assume that when a rule of NO is applied, $\Phi_1; \Gamma_2, \Delta_2$ stands for $\Phi_1; \Gamma_2 \cup \Delta_2$ and when a rule of SH is applied, $\Phi_1; \Gamma_2, \Delta_2$ is considered as $\Gamma_1, \Delta_1; \Gamma_2 \cup \Delta_2$, where $\Phi_1 = \Gamma_1 \cup \Delta_1$ and Γ_1 (Δ_1) is a set of Σ_1-equalities (-disequalities, respectively). NS can be seen as an abstract version of the one proposed in [BDS02].

Theorem 6.3. *Let T_1, T_2 be two signature-disjoint theories such that T_1 is in NO-convex and T_2 is in SH. Let* NS *be the inference system defined as the union $\mathsf{NO}_1 \cup \mathsf{SH}_2$, where NO_1 is in Figure 6.1 and SH_2 is obtained from SH_1 in Figure 6.2 by symmetry. The relation $\vdash^*_{\mathsf{NS}}$ is terminating and $\Phi_1; \Gamma_2, \Delta_2 \vdash^*_{\mathsf{NS}} false$ iff $\Phi_1 \wedge \Gamma_2 \wedge \Delta_2$ is $T_1 \cup T_2$-unsatisfiable.*

Let $T_1, ..., T_k$ and $T_{k+1}, ..., T_{k+n}$ be k theories in NO-convex and n theories in SH, respectively, and such that $\Sigma_i \cap \Sigma_j \neq \emptyset$ for $i, j = 1, \ldots, k+n$, $i \neq j$, and $n, k \geq 1$. It is possible to modularly build a satisfiability procedure for $T = \bigcup_{j=1}^{k+n} T_j$ as follows. Repeatedly use NO to obtain a satisfiability procedure for $U_0 = \bigcup_{j=1}^{k} T_j$, then repeatedly use NS to build satisfiability procedures for $U_1 = U_0 \cup T_{k+1}, \ldots, U_n = U_{n-1} \cup T_{k+n}$, where U_n is T. An alternative would be to repeatedly use SH to construct satisfiability procedures for unions of two theories in SH, followed by a repeated use of NO on the resulting theories. The particular case of combining $\mathcal{E}$ with one or more theories in SH (i.e. $k = 1$) has been extensively studied by many researchers [CLS96, RS01, Kap02, BDS02, Gan02, CK03b, RS02, CK03a], Shostak [Sho84] being the first. The correctness of the combination schemas outlined above immediately follows from the correctness of NO, SH, NS, the fact that the union of two theories in NO-convex is also in NO-convex , and that SH is contained in NO-convex . Similar results are given in [MZ03].

Finally, let us mention still another possibility to combine k theories in NO-convex and n theories in SH. It would be possible to slightly modify our inference rules to take into account $k + n$ theories ; configurations would be of the form

$$\Phi_1; \ldots; \Phi_k; \Gamma_{k+1}, \Delta_{k+1}; \ldots; \Gamma_{k+n}, \Delta_{k+n}$$

and the rule Deduction would propagate an equality between shared variables, deduced in one theory, to the other $(k+n)-1$ theories. At this point, it would not be difficult to modify the proof of correctness for NS to show that the resulting rules (taken from $\mathsf{NO}_1, \ldots, \mathsf{NO}_k, \mathsf{SH}_{k+1}, \ldots, \mathsf{SH}_{k+n}$) yield a satisfiability procedure for T. The resulting proof would be a bit more involved because of the more complex notation.

6.4 Combining ECAN-convex-theories

Although the combination schemas of Section 6.3 are already sufficient to combine several theories either in NO-convex, SH, or both, we investigate how to find a generic combination schema which features the modularity of NO and retains some of the efficiency of SH. To this end, we introduce a new basic building block which generalizes the concept of canonizer for SH-theories and can be modularly combined either to (1) build a satisfiability procedure for the union of theories (admitting extended canonizers) by a schema which allows to efficiently propagate entailed equalities as in SH but does not require to solve equalities, or to (2) obtain an extended canonizer out of two extended canonizers in a modular way, thereby showing that the class of theories for which an extended canonizer exists is closed under disjoint union.

6.4.1 Extended canonizers and ECAN-convex-theories

Definition 6.1 (Extended canonizer)**.** *Let T be a Σ-theory, and let Γ be a conjunction of Σ-equalities. Given any T-satisfiable Γ, an* extended canonizer *for T is a function $ecan(\Gamma) : T(\Sigma, X) \to T(\Sigma \cup K(\Gamma), X)$, such that, for any terms s, t, we have $T \models \Gamma \Rightarrow s = t$ iff $ecan(\Gamma)(s) = ecan(\Gamma)(t)$, where $K(\Gamma)$ is a finite set of fresh constant symbols such that $\Sigma \cap K(\Gamma) = \emptyset$.*

$ECAN-convex$ denotes the class of convex and stably infinite theories admitting an extended canonizer.

The concept of extended canonizer presents many similarities with the function $can(\Gamma)$ in [RS01]. [16] An important difference is that our extended canonizers can be modularly combined to yield satisfiability procedures for union of disjoint theories (see Section 6.4.2 below). However, [RS02] describes a solution to the problem of combining $\mathcal{E}$ with several theories in SH by means of an interesting generalization of Shostak algorithm which only requires to build a canonizer for the union of the theories (which is always possible for convex theories [CK03b]) and invokes the solvers for the component theories.

If a theory T admitting an extended canonizer *ecan* is also convex, then it is always possible to build a satisfiability procedure for T by recalling that $\Gamma \wedge \neg e_1 \wedge \cdots \wedge \neg e_n$ is T-unsatisfiable if and only if there exists some $i \in \{1, \ldots, n\}$ such that $\Gamma \wedge \neg e_i$ is T-unsatisfiable, or equivalently $T \models \Gamma \Rightarrow e_i$. This immediately implies the following proposition.

Proposition 6.2. *$ECAN - convex \subseteq NO - convex$.*

Although we can always decide the uniform word problem for a convex theory T by invoking a satisfiability procedure, it is not clear whether an extended canonizer always exists for T in NO-convex. Recall that in the Definition 6.1 of extended canonizer, we require it to return terms over $T(\Sigma \cup K(\Gamma), X)$ where $K(\Gamma)$ must be a *finite* set of fresh constant symbols. The intuition is that a constant in $K(\Gamma)$ is the representative of an equivalence class induced by $T \cup \Gamma$. Since $K(\Gamma)$ is finite, extended canonizers can only be built for a theory T such that, for each finite set Γ of ground equalities, the equivalence relation induced by $T \cup \Gamma$ has a finite number of equivalence classes. So, the problem of determining that the inclusion in Proposition 6.2 is strict amounts to proving the existence of a theory T in NO-convex such that for some set Γ of ground equalities, $T \cup \Gamma$ induces an equivalence relation with an infinite number of equivalence classes. We conjecture that such a theory exists and hence conclude the inclusion in Proposition 6.2 is strict.

6.4.2 Extended canonizers and combination of ECAN-convex theories

For technical reasons (that will become clear in a moment), we introduce the concept of *equational simplifier*, which is a partial function *eqs* taking conjunctions of equalities and returning a function whose input is an equality and which returns either $true$ or $false$ such that for any conjunction of equalities Γ and equality e, (a) *eqs* is defined for Γ and e iff Γ is T-satisfiable, and (b) $eqs(\Gamma)(e)$ is $true$ if $T \models \Gamma \Rightarrow e$, and $false$ otherwise. Indeed, for T in ECAN-convex, clause (b) can be restated as follows : for any T-satisfiable Γ and any equality $s = t$, $eqs(\Gamma)(s = t) = true$ iff $ecan(\Gamma)(s) = ecan(\Gamma)(t)$.

In the rest of this section, we assume that equational simplifiers are defined in terms of extended canonizers and we study the problem of building satisfiability procedures for unions of ECAN-convex theories. There are two possibilities.

– First, adapt NO to combine satisfiability procedures built out of equational simplifiers. (To see this, observe that equational simplifiers decides uniform word

16. $can(\Gamma)(t)$ returns a canonical form of the term t in which any (uninterpreted) subterm that is congruent to a known left hand side in an equation of Γ is replaced by the associated right hand side.

problems since these can be transformed to satisfiability problems as described in Section 6.4.1.) This gives a straightforward reformulation of the inference rules in NO where side conditions are expressed in terms of *eqs*. Since this is easy, the details are left to the reader.

– Second, build an equational simplifier for the union of theories and then derive the corresponding satisfiability procedure.

In the following, we develop the second possibility. Let T_i be a Σ_i-theory in ECAN-convex and $ecan_i$ its extended canonizer for $i = 1, 2$ and $\Sigma_1 \cap \Sigma_2 = \emptyset$. First, we show how to obtain an equational simplifier $eqs_{1,2}$ for $T_1 \cup T_2$ by using a variant of NO restricted to equalities. Then, we show how to build an extended canonizer $ecan_{1,2}$ for $T_1 \cup T_2$ out of $ecan_1, ecan_2$ and $eqs_{1,2}$. The reader may ask why we need to build the equational simplifier for $T_1 \cup T_2$ to be able to build an extended canonizer. The answer is in the definition of extended canonizer which requires Γ to be satisfiable for $ecan(\Gamma)$ to be defined. So, we need to check the $T_1 \cup T_2$-satisfiability of conjunctions of $\Sigma_1 \cup \Sigma_2$-equalities to decide whether $ecan_{1,2}$ is defined.

Lemma 6.10. *Let eqs be the function constructed in Figure 6.3. For any conjunction Γ of $\Sigma_1 \cup \Sigma_2$-equalities, we have $eqs(\Gamma)$ is defined iff Γ is $T_1 \cup T_2$-satisfiable.*

Proof. Consider the notations introduced in Figure 6.3. The lemma is a direct consequence of the fact that Γ and $\Gamma_1 \wedge \Gamma_2$ are equisatisfiable in $T_1 \cup T_2$ and the definition of EEC as $\mathsf{EEC}_1 \cup \mathsf{EEC}_2$, where EEC_1 is depicted in Figure 6.4 and EEC_2 is obtained from EEC_1 by symmetry. The key observation here is that EEC is a variant of NO restricted to equalities. □

Lemma 6.11. *Let eqs be the function constructed in Figure 6.3. For any $T_1 \cup T_2$-satisfiable conjunction Γ of $\Sigma_1 \cup \Sigma_2$-equalities, and any $\Sigma_1 \cup \Sigma_2$-equality $s = t$, we have $eqs(\Gamma)(s = t) = true$ iff $T_1 \cup T_2 \models \Gamma \Rightarrow s = t$.*

Proof. Consider the conjunction $\Gamma_1''' \wedge \Gamma_2'''$ introduced in Figure 6.3. By definition we have $x = s$ and $y = t$, and we know that $\Gamma \wedge s \neq t$ and $\Gamma_1''' \wedge \Gamma_2''' \wedge x \neq y$ are equisatisfiable in $T_1 \cup T_2$. If $\Gamma \wedge s \neq t$ is $T_1 \cup T_2$-unsatisfiable, $\Gamma_1''' \wedge \Gamma_2''' \wedge x \neq y$ is $T_1 \cup T_2$-unsatisfiable too, and we can use the inference system NO to derive $false$ from a configuration where $x \neq y$ can be viewed as either a 1-disequality or a 2-disequality. In the following, let us assume $x \neq y$ is a 1-disequality :

$$\Gamma_1''' \cup \{x \neq y\}; \Gamma_2''' \vdash^*_{\mathsf{NO}} false$$

To analyze how $false$ is obtained, several cases must distinguished :

1. If $x \in Var(\Gamma_1''')$ and $y \in Var(\Gamma_2''')$, then y is a shared variable, but x is not. The unique possibility to derive $false$ is to apply $\mathsf{Deduction}_2$ followed by $\mathsf{Contradiction}_1$. Hence, there is a shared variable z such that $T_2 \models \Gamma_2''' \Rightarrow y = z$ for $\mathsf{Deduction}_2$ to be applicable, and $T_1 \models \Gamma_1''' \cup \{y = z\} \Rightarrow x = y$ for $\mathsf{Contradiction}_1$ to be applicable. Hence, $T_1 \cup \Gamma_1''' \cup \{y = z\} \models x = z$, and so $T_1 \cup \Gamma_1''' \models x = z$, by Lemma 6.2 since y does not occur in Γ_1'''. Equivalently, we can say that $eqs_1(\Gamma_1''')(x = z) = true$. To conclude, there exists a shared variable z such that $eqs_1(\Gamma_1''')(x = z) = eqs_2(\Gamma_2''')(y = z) = true$.
2. If $x, y \in Var(\Gamma_1''')$, then x, y occur only in one side, $\mathsf{Contradiction}_1$ applies necessarily, and $eqs_1(\Gamma_1''')(x = y) = true$.

Given a set Γ of equalities and an equality $s = t$, the following procedure shows how to construct eqs for $(\Gamma, s = t)$, when defined. Let EEC be the inference system defined as the union $\mathsf{EEC}_1 \cup \mathsf{EEC}_2$, where EEC_1 is depicted in Figure 6.4 and EEC_2 is obtained by symmetry.

1. Purify Γ into $\Gamma_1; \Gamma_2$.
2. If $\Gamma_1; \Gamma_2 \vdash^*_{\mathsf{EEC}} false$, then eqs is undefined for $(\Gamma, \_)$.
3. Otherwise, let $\Gamma'_1; \Gamma'_2$ be the normal form w.r.t. $\vdash_{\mathsf{EEC}}$ such that $\Gamma_1; \Gamma_2 \vdash^*_{\mathsf{EEC}} \Gamma'_1; \Gamma'_2$. Furthermore, consider $x = s \wedge y = t$, where x, y are new variables not occurring in $Var(\Gamma'_1 \wedge \Gamma'_2)$. Let $\Gamma''_1; \Gamma''_2$ be the result of purifying $\Gamma'_1 \wedge \Gamma'_2 \wedge x = s \wedge y = t$.
4. Let $\Gamma'''_1; \Gamma'''_2$ be the normal form w.r.t. $\vdash_{\mathsf{EEC}}$ such that $\Gamma''_1; \Gamma''_2 \vdash^*_{\mathsf{EEC}} \Gamma'''_1; \Gamma'''_2$. This normal form is necessarily different from $false$ since $\Gamma_1 \wedge \Gamma_2$ is $T_1 \cup T_2$-satisfiable and x, y are distinct new variables.
5. If there exists $i \in \{1, 2\}$ such that $x, y \in Var(\Gamma''_i)$, then $eqs(\Gamma)(s = t)$ is defined and it is equal to $eqs_i(\Gamma'''_i)(x = y)$.
6. Otherwise ($x \in Var(\Gamma''_i), y \in Var(\Gamma''_j)$, for $i \neq j$), $eqs(\Gamma)(s = t)$ is defined, and it is equal to $true$ if there exists $z \in Var(\Gamma''_1) \cap Var(\Gamma''_2)$ such that $eqs_i(\Gamma'''_i)(x = z) = eqs_j(\Gamma'''_j)(y = z) = true$, otherwise it is defined as $false$.

FIGURE 6.3 – Equational Simplifier for the Union of Theories

3. If $x, y \in Var(\Gamma'''_2)$, then it is a symmetric case to the previous one, since $\Gamma'''_1; \Gamma'''_2 \cup \{x \neq y\} \vdash^*_{\mathsf{NO}} false$.
4. If $x \in Var(\Gamma'''_2)$ and $y \in Var(\Gamma'''_1)$, then it is a symmetric case to the first one, since $\Gamma'''_1; \Gamma'''_2 \cup \{x \neq y\} \vdash^*_{\mathsf{NO}} false$. □

Lemma 6.12. *Let T_1 and T_2 be two signature-disjoint convex and stably infinite theories. If an equational simplifier eqs_i is known for T_i (for $i = 1, 2$), then it is possible to construct an equational simplifier eqs for $T_1 \cup T_2$ using the combination method described in Figure 6.3.*

Notice that the result above can be repeatedly applied to build an equational simplifier for the union of n signature-disjoint, convex, and stably-infinite theories $T_1, \ldots, T_n$. So, a satisfiability procedure for $T_1 \cup \cdots \cup T_n$ can be immediately obtained. However, this still does not answer the question : does there exist an extended canonizer $ecan_{1,2}$ for $T_1 \cup T_2$ given two extended canonizers $ecan_1$ and $ecan_2$ for T_1 and T_2, respectively, and an equational simplifier $eqs_{1,2}$ for their union ? To answer this question (constructively), we analyze the equational simplifier for $eqs_{1,2}$ for $T_1 \cup T_2$ given in Figure 6.3 and we show how an extended canonizer can be obtained. The key technique underlying the analysis consists of unfolding the fresh variables (abstracting alien subterms) introduced by purification so to get terms back in the right signature. This unfolding must be done with care since we must take into account the equivalence relation on fresh variables induced by the propagation of equalities between shared variables.

Theorem 6.4. *ECAN-convex is closed under disjoint union.*

$\mathsf{Contradiction}_1$

$$\frac{\Gamma_1;\Gamma_2}{false} \quad \textbf{if } eqs_1(\Gamma_1) \text{ is undefined}$$

$\mathsf{Deduction}_1$

$$\frac{\Gamma_1;\Gamma_2}{\Gamma_1;\Gamma_2 \cup \{x=y\}} \quad \textbf{if} \begin{cases} eqs_1(\Gamma_1) \text{ is defined,} \\ eqs_2(\Gamma_2) \text{ is defined,} \\ eqs_1(\Gamma_1)(x=y) = true, \\ eqs_2(\Gamma_2)(x=y) = false, \\ x, y \in Var(\Gamma_1) \cap Var(\Gamma_2) \end{cases}$$

FIGURE 6.4 – The Inference System EEC_1

Proof. Consider the algorithm in Figure 6.3. We first introduce some additional notations. Let AV be the set of fresh variables introduced by purification occurring in $Var(\Gamma_1''' \wedge \Gamma_2''')$. By definition, each variable $v \in AV$ is associated to a i-pure term and we call T_i the *theory of* v, denoted by $th(v)$. We define an equivalence relation $\equiv$ on AV as follows : $v \equiv v'$ if $eqs_i(\Gamma_i''')(v = v') = true$, where i can possibly be 1 or 2. We say that the equivalence class $[v]_\equiv$ is *pure* if $\forall v',\ v' \equiv v \Rightarrow th(v') = th(v)$, otherwise $[v]_\equiv$ is said to be *impure*. Now, for each impure equivalence class $[v]_\equiv$, there exists necessarily at least one variable z in $Var(\Gamma_1) \cap Var(\Gamma_2)$ such that $z \equiv v$, and we take one of these variables as the representative of the class, denoted by $\chi_{[v]_\equiv}$. We are now ready to inductively define a mapping from AV to a set of terms expressed in the union of signatures :

- $\rho(v) = t\{v' \mapsto \rho(v')\}_{v' \in AV \cap Var(t)}$, where $t = ecan_i(\Gamma_i''')(v)$ and $i = th(v)$, if $[v]_\equiv$ is pure ; and
- $\rho(v) = \chi_{[v]_\equiv}$, if $[v]_\equiv$ is impure.

According to this definition, the range of ρ is included in $T(\Sigma_1 \cup \Sigma_2, X \cup (AV \cap Var(\Gamma_1) \cap Var(\Gamma_2)))$. Now, we are almost done since we can use ρ to define the heterogeneous "normal forms" associated to terms s and t which are respectively abstracted by x and y (see Figure 6.3). If we define $ecan(\Gamma)(s)$ (resp. $ecan(\Gamma)(t)$) as equal to $\rho(x)$ (resp. $\rho(y)$), we get a mapping satisfying the expected property since $eqs(\Gamma)(s = t) = true$ iff $\rho(x) = \rho(y)$. Therefore, we have shown how to construct an extended canonizer for $T_1 \cup T_2$ when extended canonizers are known for T_1 and T_2. □

6.5 Extended canonizers, solvers, canonizers, and satisfiability procedures

We describe the relationships between theories in ECAN-convex, those in SH, and some in NO-convex which are not in SH.

Extended canonizer for Shostak theories by solvers and canonizers

Proposition 6.3. *$SH \subseteq ECAN - convex$, i.e. theories in SH admits an extended canonizer.*

Proof. Let T be an SH-theory and *solve* and *canon* its solver and canonizer, respectively. We define an extended canonizer *ecan* for T, as follows. If $solve(\Gamma) = false$, then *ecan* is undefined. If $solve(\Gamma)$ returns a substitution λ, then $ecan(\Gamma)(s)$ returns $canon(s\lambda)$. □

The proof of the Proposition above suggests how to reduce the uniform word problem $T \models \Gamma \Rightarrow s = t$ for a theory T in SH to the word problem $T \models s\lambda = t\lambda$, where λ is the substitution obtained by iteratively applying *solve* to Γ. In turn, $T \models s\lambda = t\lambda$ reduces to checking whether $canon(s\lambda)$ is syntactically equal to $canon(t\lambda)$ (a similar observation is done in [Gan02]). The key observation here is that substituting equalities in Γ with their solved form $\hat{\lambda}$ can be done without backtracking, thanks to the properties of *solve*. This is not possible for some theories whose uniform word problem can be decided by using an extended canonizer. For example, $\mathcal{E}$ admits efficient algorithms to solve its uniform word problem (see, e.g. [DST80]) but it is easy to show that it does not admit a solver (see e.g. [MZ03]) ; so $\mathcal{E}$ is not in SH.

Extended canonizer for Horn theories by saturation

Now we show that extended canonizers can be effectively computed for the class of theories axiomatized by a finite set of Horn clauses, which enjoys the finite saturation property (cf. Definition 4.4 of Chapter 4). Recall that a theory has the finite saturation property if any saturation of an arbitrary set of ground flat literals together with the theory axioms is finite.

To decide the uniform word problem we can adopt a refutational approach, i.e. to decide $T \models \Gamma \Rightarrow s = t$ we can resort to a T-satisfiability procedure invoked on $\Gamma \cup \{s \neq t\}$. In the context of building rewriting-based satisfiability procedures, the set $\Gamma \cup \{s \neq t\}$ is assumed to be flat, i.e. s, t are constants and Γ only contains ground flat equalities. In other word, each input ground term can be represented by a constant. This allows us to refine, without loss of generality, the Definition 6.1 so that extended canonizers put constants, instead of terms, in normal form. This refinement makes sense when combining decision procedures using the Nelson-Oppen method because only equalities between constants (free variables) must be propagted. In addition, it allows us to avoid confusion between free variables and universally quantified variables in clauses when dealing with saturation.

For these reasons, we refine the notion of extended canonizer as follows.

Definition 6.2. *Let T be a Σ-theory, and let Γ be a conjunction of ground flat Σ-equalities. Given any T-satisfiable Γ, an* extended canonizer *for T is a function $ecan(\Gamma) : T(\Sigma, X) \to \Sigma^C \cup K(\Gamma)$, such that, for any constants $c, c' \in \Sigma^C \cup K(\Gamma)$, we have $T \models \Gamma \Rightarrow c = c'$ iff $ecan(\Gamma)(c) = ecan(\Gamma)(c')$, where $K(\Gamma)$ is a finite set of fresh constant symbols such that $\Sigma^C \cap K(\Gamma) = \emptyset$.*

Clearly, a theory admits an extended canonizer as defined in the Definition 6.2 if and only if it admits an extended canonizer as defined in the Definition 6.1.

The following result is closyly related to the deduction-completeness of rewriting-based satisfiability procedures (cf. Lemma 5.4 of Chapter 5).

Lemma 6.13. *Let T be a theory axiomatized by a finite set $Ax(T)$ of Horn clauses, which is saturated with respect to $\mathcal{SP}$. Assume that for every set S of ground flat literals, any saturation S' of $Ax(T) \cup S$ by $\mathcal{SP}$ is finite and contains no equality of the form $X = t$, where $X \notin Var(t)$* [17]*. Then, for each equality between constants $c = c'$ such that $T \cup S \models c = c'$, we have that the subset containing all equalities between constants in S' entails $c = c'$.*

Proof. Assume that there is some equality $c = c'$ between constants such that $T \cup S \models c = c'$. Reasoning by refutation, $T \cup S \models c = c'$ iff $S \wedge c \neq c'$ is not T-satisfiable. Hence, it must be possible to derive the empty clause by applying $\mathcal{SP}$ to the set $S' \cup \{c \neq c'\}$. Since S' is T-satisfiable and saturated, only inferences involving both clauses from (or inferred from) S' and $c \neq c'$ can yield the empty clause. Let us analyze such inferences, i.e. inferences between $c \neq c'$ and some clause C' in S'. If there is an inference between $c \neq c'$ and C', then C' must be an equality between constants or variables. This is because the ordering used in the Superposition Calculus is defined such that a disequality is always bigger than an equality and as consequence an equality is maximal in a clause only if the latter contains no disequalities. If C' contains a variable, then C' must have the form $X = t$, where $X \notin Var(t)$. That would contradict the assumption of the lemma. If C' only contains constants, then C' is an equality between constants and the clause inferred from $c \neq c'$ and C' must be a disequality between constants. This means that an inference between $c \neq c'$ and a clause in S' is possible only if the latter is an equality between constants and only derives a disequality between constants. Therefore, the subset containing all equalities between constants in S' together with $c \neq c'$ suffice to infer the empty clause. Or equivalently the subset containing all equalities between constants in S' entails $c = c'$. □

The above lemma allows us to state that the finite saturation property is suffcient for a Horn theory to have an extended canonizer.

Lemma 6.14. *Let T be a theory axiomatized by a finite set $Ax(T)$ of Horn clauses. Assume that for every set S of ground flat literals, any saturation of $Ax(T) \cup S$ by $\mathcal{SP}$ is finite. Then T admits an extended canonizer.*

Proof. Given a set Γ of ground flat equalities, the following method allows us to compute $ecan(\Gamma)$.

1. Saturate $Ax(T) \cup \Gamma$ with respect to $\mathcal{SP}$ to obtain the set S'.
2. If S' contains an equality of the form $X = t$, where $X \notin Var(t)$, then $T \cup \Gamma$, or equivalently S', has only trivial model. Therefore we can fix an arbitrary constant c_0 and define $ecan(\Gamma)(c) = c_0$ for all constants c.
3. Otherwise, let S_e be the subset of S' containing all equalities between constants. By using abstract congruence closure technique described in [BTV03], we can build a canonical rewrite system R_{S_e} out of the set S_e of equalities between constants. Then define $ecan(\Gamma)(c)$ to be the normal form of c with respect to R_{S_e}.

By construction, for every pair of constants c, c' we have that $T \models \Gamma \Rightarrow c = c'$ if and only if $ecan(\Gamma)(c) = ecan(\Gamma)(c')$. □

The following theorem follows directly from Lemma 6.14.

17. This condition also implies that T is stably infinite (cf. Section 5.3 of Chapter 5). Therefore we can apply the Nelson-Oppen based combination schemas proposed in this chapter.

Theorem 6.5. *Let T be a theory axiomatized by a finite set $Ax(T)$ of Horn clauses, which is saturated with respect to $\mathcal{SP}$. Assume that for every set S of ground flat literals, any saturation of $Ax(T) \cup S$ by $\mathcal{SP}$ is finite and contains no equality of the form $X = t$, where $X \notin Var(t)$. Then $T \in ECAN - convex$.*

It is interesting to notice that the superposition calculus provide us with an uniform method to derive satisfiability procedures for not only a theory T with the finite saturation property, but also the union of T with the theory of equality. As consequence, the extended canonizer built by superposition is actually an extended canonizer for the union of T and the theory of equality. In this respect, variants of the superposition calculus (e.g. superposition modulo Shostak theories [GHW03], superposition modulo associativity and commutativity [RV95, Vig94]) give us a Shostak-like method, which is different from the Nelson and Oppen-like one in the proof of Theorem 6.4, to build in an uniform way an extended canonizer for the union of some theory and the theory of equality.

6.6 Discussion

We have presented combination schemas for disjoint unions of (a) two theories in NO-convex, (b) two theories in SH, and (c) one theory in NO-convex with one in SH. We have shown how such schemas are related to Nelson-Oppen and Shostak approaches to combination as well as with many of the refinements available in the literature. Our formalization highlights the key ideas underlying each combination and allows us to derive proofs of correctness which are easy to grasp. We believe this is a valuable synthesis for further investigations. To justify this claim, we have introduced the concept of extended canonizer which abstracts algorithms for deciding the uniform word problem of a theory and it is modular, i.e. an extended canonizer can be built out of the extended canonizers for the component theories. This is in contrast to the lack of modularity of solvers for Shostak combination schema. Another advantage is the fact that it can be easily implemented in terms of solvers and canonizers for Shostak theories or by rewriting techniques as suggested e.g. in [ARR03].

There are several main lines for future work. First, we want to derive a more precise characterization of the theories admitting an extended extended canonizer. In this respect, a promising line of research would be to study for which theories the uniform word problem can be reduced to a word problem. Second, we want to study the complexity of extended canonizers in the union of theories. We believe it would be interesting to apply our combination results to polynomial time decidable uniform word problems as described in [BG96, Gan01]. Third, we intend to empirically evaluate the efficiency of extended canonizers by conducting some experiments in haRVey [DR03]. The interest here is to obtain an efficient combination between extended canonizers and propositional solvers. This requires to equip extended canonizers with the capability of generating useful theory-specific facts which, once projected into the propositional domain, allow to reduce the search space. Finally, we plan to study how extended canonizers can be used when non-convex theories are combined.

Chapitre 7

Mélanges disjoints de théories convexes et de théories universelles

Nous avons vu dans le Chapitre 6 que la correction des procédures de combinaison (pour des théories convexes, des théories de Shostak et des théories admettant un canoniseur étendu) est garantie sous l'hypothèse de stable infinité. Nous travaillons dans le but d'étendre les résultats obtenus aux théories convexes mais pas nécessairement stablement infinies. Dans [BDS02], les auteurs ont montré que toute théorie convexe qui n'a pas de modèle trivial est stablement infinie et donc peut être combinée en utilisant la méthode de Nelson et Oppen. Nous allons plus loin en montrant dans ce travail que les procédures de satisfiabilité des théories convexes peuvent être combinées même si ces théories ne sont pas stablement infinies, à condition de savoir si elles ont un modèle trivial. En pratique, cette condition n'est pas vue comme une restriction d'applicabilité de notre résultat car être capable de décider si une théorie a un modèle trivial ou non est une moindre condition par rapport au problème de satisfiabilité dans cette théorie. Notre résultat s'applique bien aux théories finiment axiomatisées puisque pour décider si ces théories ont un modèle trivial il suffit de remplacer tous les termes se trouvant dans les axiomes par la même constante et puis de tester la satisfiabilité de l'ensemble de formules closes obtenu [18].

Dans le même esprit, nous montrons également qu'on peut avoir une construction modulaire de canoniseur étendu pour l'union des théories convexes (pas nécessairement stablement infinies) admettant un canoniseur étendu. De plus nous n'avons pas besoin de savoir si les théories composantes ont ou non un modèle trivial. Ce résultat généralise le résultat de modularité de canoniseur étendu pour les théories convexes et stablement infinies (cf. Théorème 6.4) énoncé dans le Chapitre 6.

Nous considérons par la suite la combinaison des théories universelles non convexes. En s'inspirant du résultat de [BGN+06], nous montrons que les théories universelles non convexes peuvent être combinées, à condition que pour ces théories le problème de satisfiabilité dans un modèle infini de ces théories soit décidable. Finalement, nous conjecturons que les théories universelles finiment axiomatisées peuvent être combinées sans aucune autre restriction.

18. Pour vérifier la satisfiabilité d'un ensemble de formules closes, on le transforme en un ensemble de clauses closes et ensuite on applique la saturation à ce dernier (cf. Chapitre 4)

Par simplicité, nous nous limitons dans ce travail à la combinaison des théories dont la signature ne contient pas de symbole de relation.

7.1 Satisfiabilité dans des mélanges de théories convexes

Le résultat suivant sera utilisé ultérieurement dans la preuve de correction de la procédure de combinaison des théories convexes.

Proposition 7.1. *Soit T une théorie convexe. Si T a un modèle non trivial, alors T a un modèle infini.*

Démonstration. Par contradiction, supposons que T n'a pas de modèle infini, alors T implique nécessairement une disjonction d'égalités entre variables, i.e $T \models \bigvee_{1 \leq j \neq k \leq n} x_j = x_k$. Puisque T est convexe, T implique alors une égalité entre variables, i.e. $T \models x_i = x_j$. T n'a donc que des modèles triviaux, ce qui contredit l'hypothèse de la proposition. □

7.1.1 Satisfiabilité

Dans ce qui suit, pour décider si une formule ϕ est satisfiable dans un modèle infini d'une théorie convexe T, nous vérifions simplement si $\phi \cup \{x \neq y\}$ est T-satisfiable, où x, y sont deux nouvelles variables (existentiellement quantifiées).

Cette même technique a été aussi utilisée pour vérifier si une théorie équationnelle est stablement infinie : pour toute théorie équationnelle T, T est stablement infinie si et seulement si $T \cup \{\exists X, Y(X \neq Y)\}$ est consistante (voir e.g [BT97]). Cependant elle ne permet pas de vérifier la stable infinité des théories convexes. Par exemple, la théorie T dont l'ensemble des axiomes est $Ax(T) = \{\forall X, Y(X = Y \vee f(a) \neq g(b))\}$ n'est pas stablement infinie mais $T \cup \{\exists X, Y(X \neq Y)\}$ est consistante.

Proposition 7.2. *Soit T_i une Σ_i-théorie convexe telle que nous savons si T_i a un modèle trivial, et le problème de T_i-satisfiabilité est décidable, pour $i = 1, 2$. Supposons que les signatures de T_1 et T_2 sont finies et disjointes. Alors le problème de satisfiabilité est décidable dans $T_1 \cup T_2$.*

Démonstration. Soit Φ un ensemble de $\Sigma_1 \cup \Sigma_2$-littéraux sans quantificateur. Par abstraction de variable, Φ est $T_1 \cup T_2$-équisatisfiable à un ensemble $\Phi_1 \cup \Phi_2$ tel que Φ_i ne contient que des Σ_i-littéraux sans quantificateur. Soit $V = Var(\Phi_1) \cap Var(\Phi_1)$. Soit Φ_0 un arrangement sur V, i.e. un ensemble tel que pour $x, y \in V$ soit $x = y \in \Phi_0$ soit $x \neq y \in \Phi_0$. Il est évident que $\Phi_1 \cup \Phi_2$ est $T_1 \cup T_2$-satisfiable si et seulement si $\Phi_1 \cup \Phi_2 \cup \Phi_0$ est $T_1 \cup T_2$-satisfiable pour un certain arrangement Φ_0. Or $\Phi_1 \cup \Phi_2 \cup \Phi_0$ est $T_1 \cup T_2$-satisfiable si et seulement si $\Phi_1 \cup \Phi_0$ est T_1-satisfiable dans un modèle $\mathcal{M}_1$ et $\Phi_2 \cup \Phi_0$ est T_2-satisfiable dans un modèle $\mathcal{M}_2$ tels que les domaines de $\mathcal{M}_1$ et $\mathcal{M}_2$ ont la même cardinalité. Nous allons montrer que de tels modèles peuvent être exhibés quand on considère les théories convexes pour lesquelles nous savons si elles admettent un modèle trivial. Nous avons les cas suivants :

- $\Phi_1 \cup \Phi_0$ est T_1-satisfiable dans un modèle $\mathcal{M}_1$ et $\Phi_2 \cup \Phi_0$ est T_2-satisfiable dans un modèle $\mathcal{M}_2$ tels que $\mathcal{M}_1$ et $\mathcal{M}_2$ sont tous les deux non triviaux. Ceci est déterminé en vérifiant que pour $i = 1, 2$, $\Phi_i \cup \Phi_0 \cup \{x \neq y\}$ est T_i-satisfiable, où x, y sont de nouvelles variables. Par la Proposition 7.1, nous savons qu'il existe

un modèle $\mathcal{M}'_1$ de T_1 satisfaisant $\Phi_1 \cup \Phi_0$ et un modèle $\mathcal{M}'_2$ de T_2 satisfaisant $\Phi_2 \cup \Phi_0$ tels que $\mathcal{M}'_1$ et $\mathcal{M}'_2$ ont la même cardinalité infinie. En conséquence, $\Phi_1 \cup \Phi_2$ est $T_1 \cup T_2$-satisfiable.
- $\Phi_i \cup \Phi_0$ est T_i-satisfiable mais seulement dans un modèle trivial, pour $i = 1, 2$. Pour ce cas, il suffit de vérifier que pour $i = 1, 2$, $\Phi_i \cup \Phi_0 \cup \{x \neq y\}$ est T_i-insatisfiable, où x, y sont de nouvelles variables. Donc $\Phi_1 \cup \Phi_2$ est $T_1 \cup T_2$-satisfiable dans un modèle trivial.
- $\Phi_1 \cup \Phi_0$ est T_1-satisfiable seulement dans un modèle trivial et $\Phi_2 \cup \Phi_0$ est T_2-satisfiable dans un modèle non trivial. Ceci est vérifiable car il suffit d'assurer $\Phi_1 \cup \Phi_0 \cup \{x \neq y\}$ est T_1-insatisfiable, et $\Phi_2 \cup \Phi_0 \cup \{x \neq y\}$ est T_2-satisfiable, où x, y sont de nouvelles variables. Maintenant, il suffit de vérifier si $\Phi_2 \cup \Phi_0$ est satisfiable dans un modèle trivial de T_2. Pour cela, il suffit de regarder si $\Phi_2 \cup \Phi_0$ contient une diségalité. En effet, $\Phi_2 \cup \Phi_0$ est satisfiable dans un modèle trivial de T_2 si et seulement si $\Phi_2 \cup \Phi_0$ ne contient pas de diségalités et T_2 a un modèle trivial.

□

Si nous considérons les théories convexes finiment axiomatisées, la procédure de test d'existence de modèle trivial est donc effectivement réalisable. Comme expliqué précédemment, lorsque la théorie est finiment axiomatisée, pour décider si la théorie a un modèle trivial il suffit de remplacer tous les termes qui se trouvent dans les axiomes par la même constante et puis de tester la satisfiabilité de l'ensemble des formules closes obtenu. Par conséquent, nous avons le résultat suivant.

Corollaire 7.1. *Soit T_i une Σ_i-théorie convexe finiment axiomatisée telle que le problème de T_i-satisfiabilité est décidable, pour $i = 1, 2$. Supposons que les signatures de T_1 et T_2 sont finies et disjointes. Alors le problème de satisfiabilité est décidable dans $T_1 \cup T_2$.*

7.1.2 Procédure de combinaison déterministe

La Proposition 7.2 permet de dériver une procédure de combinaison *déterministe*. L'entrée de la procédure est une conjonction de littéraux mixtes. Elle est purifiée afin d'obtenir deux conjonctions de littéraux purs. L'étape de *test déterministe* consiste en deux sous-étapes : (1) appliquer le système d'inférence NO (cf. Sous-section 6.3.1 du Chapitre 6) aux deux conjonctions de littéraux purs pour déduire les égalités entre variables partagées et pour détecter l'insatisfiabilité ; (2) déterminer si les conjonctions pures sont satisfiables dans un modèle trivial ou un modèle infini et retourner le résultat `satisfiable` ou `insatisfiable`. La Figure 7.1 nous donne les détails de la procédure de combinaison déterministe.

7.2 Construction modulaire de canoniseur étendu

Nous considérons la modularité du problème du mot uniforme pour les théories convexes pour lesquelles le problème du mot uniforme est décidable. Nous étudions ensuite une construction modulaire de canoniseur étendu pour l'union des théories convexes qui admettent un canoniseur étendu. Les théories considérées ne sont pas nécessairement stablement infinies et nous n'avons pas besoin de savoir si elles ont un modèle trivial, contrairement à la modularité du problème de satisfiabilité.

Purification Soit Φ une conjonction de littéraux. Cette étape calcule une conjonction $\Phi_1 \cup \Phi_2$ satisfaisant les propriétés suivantes :

(a) chaque littéral dans Φ_i est un Σ_i-littéral, pour $i = 1, 2$;

(b) $\Phi_1 \cup \Phi_2$ est $T_1 \cup T_2$-satisfiable si et seulement si Φ est $T_1 \cup T_2$-satisfiable.

Test déterministe Soit $\Phi_1 \cup \Phi_2$ une conjonction de littéraux obtenue par l'étape de purification. Cette étape consiste en deux sous-étapes suivantes.

1. Appliquer le système d'inférence NO, qui est l'union de NO_1 et NO_2 (NO_1 est décrit dans la Figure 6.1 et NO_2 est obtenu par symétrie), à $\Phi_1; \Phi_2$. Si $\Phi_1; \Phi_2 \vdash^*_{\mathsf{NO}} \mathit{false}$ alors retourner `insatisfiable` sinon aller à 2.
2. Sans perte de généralité, soit $\Phi_1; \Phi_2$ la configuration finale obtenue à la fin de l'étape 1. Soit x, y deux nouvelles variables. Appliquer NO à $\Phi_1 \cup \{x \neq y\}; \Phi_2$ et à $\Phi_1; \Phi_2 \cup \{x \neq y\}$.
   - Si $\Phi_1 \cup \{x \neq y\}; \Phi_2 \vdash^*_{\mathsf{NO}} \mathit{false}$ et $\Phi_1; \Phi_2 \cup \{x \neq y\} \vdash^*_{\mathsf{NO}} \mathit{false}$ alors retourner `satisfiable`.
   - Si $\Phi_1 \cup \{x \neq y\}; \Phi_2 \not\vdash^*_{\mathsf{NO}} \mathit{false}$ et $\Phi_1; \Phi_2 \cup \{x \neq y\} \not\vdash^*_{\mathsf{NO}} \mathit{false}$ alors retourner `satisfiable`.
   - Si $\Phi_1 \cup \{x \neq y\}; \Phi_2 \vdash^*_{\mathsf{NO}} \mathit{false}$ et $\Phi_1; \Phi_2 \cup \{x \neq y\} \not\vdash^*_{\mathsf{NO}} \mathit{false}$ alors
     - si Φ_2 contient une diségalité, retourner `insatisfiable`.
     - sinon si T_2 a un modèle trivial, retourner `satisfiable`.
     - sinon retourner `insatisfiable`.
   - Si $\Phi_1 \cup \{x \neq y\}; \Phi_2 \not\vdash^*_{\mathsf{NO}} \mathit{false}$ et $\Phi_1; \Phi_2 \cup \{x \neq y\} \vdash^*_{\mathsf{NO}} \mathit{false}$ alors
     - si Φ_1 contient une diségalité, retourner `insatisfiable`.
     - sinon si T_1 a un modèle trivial, retourner `satisfiable`.
     - sinon retourner `insatisfiable`.

FIGURE 7.1 – Combinaison déterministe de théories convexes

Le problème du mot uniforme modulo une théorie convexe peut se ramener au problème de satisfiabilité modulo cette théorie, i.e. $T \models s_1 = t_1, \ldots, s_n = t_n \Rightarrow s = t$ si et seulement si $\{s_1 = t_1, \ldots, s_n = t_n, s \neq t\}$ est T-insatisfiable. Or nous savons par la Proposition 7.1 que si l'ensemble de littéraux $\{s_1 = t_1, \ldots, s_n = t_n, s \neq t\}$ est T-satisfiable alors il est satisfiable dans un T-modèle infini. De plus, $\{s_1 = t_1, \ldots, s_n = t_n, s \neq t\}$ n'est pas satisfiable dans un T-modèle trivial. Il n'est donc plus nécessaire de savoir si la théorie a un modèle trivial. De ce fait, nous avons le résultat suivant sur la modularité du problème du mot uniforme pour les théories convexes.

Proposition 7.3. *Soit T_i une Σ_i-théorie convexe telle que le problème de mot uniforme modulo T_i est décidable, pour $i = 1, 2$. Supposons que les signatures de T_1 et T_2 sont finies et disjointes. Alors le problème du mot uniforme modulo $T_1 \cup T_2$ est décidable.*

Nous allons considérer une construction modulaire de canoniseur étendu. Nous avons défini un canoniseur étendu comme une fonction définie seulement pour des conjonctions d'égalités satisfiables (cf. Définition 6.1 du Chapitre 6). Cependant pour décider si une conjonction d'égalités est satisfiable, on a besoin de faire appel à une procédure de satisfiabilité. Ceci nous amène à raffiner la Définition 6.1 de telle sorte

qu'un canoniseur étendu devient une fonction définie pour toutes les conjonctions d'égalités.

Définition 7.1. *Soit T une Σ-théorie telle que le problème de T-satisfiabilité est décidable. Pour toute conjonction Γ d'égalités, un* canoniseur étendu *pour T est une fonction $ecan(\Gamma) : T(\Sigma, X) \to T(\Sigma \cup K(\Gamma), X)$ telle que pour tout terme s, t, $T \models \Gamma \Rightarrow s = t$ si et seulement si $ecan(\Gamma)(s) = ecan(\Gamma)(t)$, où $K(\Gamma)$ est un ensemble fini de constantes tel que $\Sigma \cap K(\Gamma) = \emptyset$.*

Dans la nouvelle définition, Γ n'est plus nécessairement T-satisfiable. Lorsque Γ est T-insatisfiable, il suffit de choisir arbitrairement une constante c dans $K(\Gamma)$ et de définir $ecan(\Gamma)(t) = c$ pour tout terme t. Cette nouvelle définition s'applique toujours aux théories de Shostak ainsi qu'aux théories axiomatisées par un ensemble fini de clauses de Horn ayant la propriété de saturation finie. En effet, un solveur et un canoniseur nous permet d'avoir non seulement un canoniseur étendu mais également une procédure de satisfiabilité pour les théories de Shostak. De même, la saturation fournit une procédure de satisfiabilité et aussi un canoniseur étendu pour des théories axiomatisées par un ensemble fini de clauses de Horn ayant la propriété de saturation finie.

Avant de donner le résultat de modularité de canoniseur étendu, nous rappelons la notion de simplificateur équationnel (equational simplifier, cf. Sous-section 6.4.2 du Chapitre 6). Un simplificateur équationnel est une fonction partielle *eqs* qui prend comme argument des conjonctions d'égalités et qui retourne une fonction, dont l'entrée est une égalité, et qui retourne *true* ou $false$ telle que pour toute conjonction d"égalités Γ et pour toute égalité e (a) *eqs* est définie pour Γ et e si et seulement si Γ est T-satisfiable, et (b) $eqs(\Gamma)(e)$ est *true* si $T \models \Gamma \Rightarrow e$, et $false$ sinon. Il est facile de voir que si T est une théorie convexe admettant un canoniseur étendu *ecan*, et telle que le problème de T-satisfiabilité est décidable, alors la clause (b) peut être reformulée comme suit : pour toute conjonction T-satisfiable Γ d'égalités et pour toute égalité $s = t$, $eqs(\Gamma)(s = t) = true$ si et seulement si $ecan(\Gamma)(s) = ecan(\Gamma)(t)$.

Avec la reformulation ci-dessus du concept de canoniseur étendu, nous avons le résultat de modularité suivant.

Théorème 7.1. *Soit T_i une Σ_i-théorie convexe qui admet un canoniseur étendu, et telle que le problème de T_i-satisfiabilité est décidable, pour $i = 1, 2$. Supposons que les signatures de T_1 et T_2 sont finies et disjointes. Alors $T_1 \cup T_2$ admet un canoniseur étendu.*

Démonstration. Considérons un ensemble Γ d'égalités, on purifie Γ en $\Gamma_1; \Gamma_2$. L'argument de preuve continue de façon très similaire à celle du Théorème 6.4. Seulement les points suivants sont différents :

(i) Si $\Gamma_1; \Gamma_2 \not\vdash^*_{\mathsf{EEC}} false$, cela ne garantit pas que $\Gamma_1 \cup \Gamma_2$ est $T_1 \cup T_2$-satisfiable. Il nous faut faire une vérification de cardinalité des modèles composants. Soit $\Gamma'_1; \Gamma'_2$ la forme normale de $\Gamma_1; \Gamma_2$ par rapport à $\vdash^*_{\mathsf{EEC}}$. Nous devons considérer les cas suivants
   - Γ'_i est T_i-satisfiable dans un modèle non trivial, pour $i = 1, 2$: alors pour tout terme t, $ecan(\Gamma, t)$ est définie comme dans le cas des théories convexes et stablement infinies (cf. preuve du Théorème 6.4).

– Γ'_i (ou Γ'_2) est satisfiable seulement dans un modèle trivial de T_1 (ou T_2) : alors pour tout terme t, $ecan(\Gamma, t) = c$, où c une constante dans $K(\Gamma)$ arbitrairement fixée au départ.

(ii) Quand un simplificateur équationnel (equational simplifier) n'est pas défini pour Γ—i.e. Γ est $T_1 \cup T_2$-insatisfiable—alors pour tout terme t, $ecan(\Gamma, t) = c$, où c une constante dans $K(\Gamma)$ arbitrairement fixée au départ.

□

7.3 Théories universelles

Nous avons vu que la convexité nous permet de réduire la satisfiabilité d'une formule à la satisfiabilité de celle-ci dans un modèle trivial ou dans un modèle infini. Ceci nous permet d'avoir une procédure terminante pour exhiber un modèle quand la formule considérée est insatisfiable. Nous allons généraliser le résultat de combinaison de théories convexes aux théories non convexes en imposant des restrictions sur les théories composantes de telle sorte que la procédure de recherche de modèle soit toujours terminante.

Définition 7.2 (Théorie universelle)**.** *Une théorie universelle est un ensemble de sentences universelles.*

Les résultats suivants sont classiques et peuvent être trouvés dans n'importe quelle référence sur la théorie de modèle (voir e.g. [Hod94]).

Proposition 7.4. *Soit $\varphi(\tilde{x})$ une Σ-formule sans quantificateurs et $\mathcal{A}$, $\mathcal{B}$ deux Σ-structures telles que*
– *$\mathcal{B} \models \varphi(\tilde{b})$ pour une certaine séquence $\tilde{b}$ dans B, et*
– *il existe un plongement h de $\mathcal{A}$ dans $\mathcal{B}$, dont l'image contient $\tilde{b}$.*
Alors $\varphi(\tilde{x})$ est satisfiable dans $\mathcal{A}$.

Proposition 7.5. *Soient T une Σ-théorie universelle et $\mathcal{A}$ une Σ-structure. Alors $\mathcal{A}$ peut être plongée dans un modèle de T si et seulement si $\mathcal{A}$ est un modèle de T.*

Nous allons définir la notion de diagramme d'une interprétation. Intuitivement, le diagramme d'une interprétation donne une spécification de celle-ci. A noter que dans les références sur la théorie des modèles, e.g. [Hod94], le diagramme d'une $\Sigma(\tilde{c})$-structure $\mathcal{A}$ est l'ensemble de tous les $\Sigma(\tilde{c})$-littéraux qui sont satisfiables dans $\mathcal{A}$. Ceci est dû au fait que les variables existentiellement quantifiées sont considérées comme des constantes enrichissant la signature, de ce fait on parle de $\Sigma(\tilde{c})$-structure. Dans ce travail, nous préférons garder les variables et utilisons les interprétations au lieu des $\Sigma(\tilde{c})$-structures. Nous adoptons donc la définition standard du diagramme comme suit.

Définition 7.3 (Diagramme)**.** *Soient $\mathcal{A}$ une Σ-structure et α une valuation d'un ensemble de variable X dans A. Le diagramme de l'interprétation $(\mathcal{A}, \alpha)$, noté $Diag((\mathcal{A}, \alpha))$, est l'ensemble construit par la procédure suivante :*

1. *Au départ, $Diag((\mathcal{A}, \alpha)) = \emptyset$.*
2. *Pour chaque élément a dans A, générer une nouvelle variable x_a. Pour chaque pair d'éléments distincts $a, b \in A$, ajouter $x_a \neq x_b$ à $Diag((\mathcal{A}, \alpha))$.*

3. *Pour chaque variable x dans X, si $\alpha(x) = a$ alors ajouter $x = x_a$ à $Diag((\mathcal{A}, \alpha))$.*
4. *Pour chaque constante c dans Σ, si $c^{\mathcal{A}} = a$ alors ajouter $c = x_a$ à $Diag((\mathcal{A}, \alpha))$.*
5. *Pour chaque symbole de fonction f d'arité n dans Σ, si $f^{\mathcal{A}}(a_1, \ldots, a_n) = a$ alors ajouter $f^{\mathcal{A}}(x_{a_1}, \ldots, x_{a_n}) = x_a$ à $Diag((\mathcal{A}, \alpha))$.*
6. *Pour chaque symbole de relation P d'arité n dans Σ, si $(a_1, \ldots, a_n) \in P^{\mathcal{A}}$ alors ajouter $P^{\mathcal{A}}(x_{a_1}, \ldots, x_{a_n})$ à $Diag((\mathcal{A}, \alpha))$, sinon ajouter $\neg P^{\mathcal{A}}(x_{a_1}, \ldots, x_{a_n})$ à $Diag((\mathcal{A}, \alpha))$.*

Proposition 7.6. *Soient $\mathcal{A}$, $\mathcal{B}$ deux Σ-structures et α une valuation d'un ensemble de variables X dans A. Si $Diag((\mathcal{A}, \alpha))$ est satisfiable dans $\mathcal{B}$, alors $\mathcal{A}$ peut être plongée dans $\mathcal{B}$.*

Démonstration. On définit l'application h de telle sorte que :

$$h(a) = x_a^{\mathcal{B}}\,, \text{ pour chaque } a \in A\,,$$

Il est facile de voir que h est un plongement de $\mathcal{A}$ dans $\mathcal{B}$. □

Les Propositions 7.4, 7.5 et 7.6 nous permettent de conclure le résultat suivant.

Proposition 7.7. *Soient T une Σ-théorie universelle et Φ une Σ-formule sans quantificateurs. Soient $\mathcal{A}$ une Σ-structure et α une certaine valuation de $Var(\Phi)$ dans A. Si $\Phi \cup Diag((\mathcal{A}, \alpha))$ est T-satisfiable alors $\mathcal{A}$ est un modèle de T et de plus $\mathcal{A}$ satisfait Φ.*

Rappelons que $\exists^{\geq n}$ désigne la formule $\exists x_1 \ldots x_n . \bigwedge_{j \neq k}(x_j \neq x_k)$, et $\exists^{\leq n}$ désigne la formule $\forall x_0 \ldots x_n . \bigvee_{j \neq k}(x_j = x_k)$. Et on note par $\exists^{\infty}$ la formule $\{\exists^{\geq n} | n \geq 2\}$. Il est facile de voir que $\exists^{\geq n}$, $\exists^{\leq n}$, et $\exists^{\infty}$ impose la cardinalité de chacun de leur modèles à être respectivement au moins n, au plus n et infini.

Nous considérons les théories universelles et montrons que si le problème de satisfiabilité modulo une théorie universelle dans un modèle infini est décidable dans chaque théorie composante alors le problème de satisfiabilité modulo l'union des théories est décidable.

Rappelons que le problème de satisfiabilité dans un modèle de cardinalité κ est décidable dans une théorie T si pour toute formule sans quantificateur Φ nous savons si Φ est T-satisfiable dans un modèle de cardinalité κ.

Proposition 7.8. *Soit T_i une Σ_i-théorie universelle telle que le problème de satisfiabilité et le problème de satisfiabilité dans un modèle de cardinalité infinie sont décidables, pour $i = 1, 2$. Supposons que les signatures de T_1 et T_2 sont finies et disjointes. Alors le problème de satisfiabilité et le problème de satisfiabilité dans un modèle de de cardinalité infinie sont décidables dans $T_1 \cup T_2$.*

Démonstration. De façon similaire à la Proposition 7.2, soit Φ un ensemble de $\Sigma_1 \cup \Sigma_2$-littéraux sans quantificateur. Par abstraction de variable, Φ est $T_1 \cup T_2$-équisatisfiable à un ensemble $\Phi_1 \cup \Phi_2$ tel que Φ_i ne contient que des Σ_i-littéraux sans quantificateur. Soit $V = Var(\Phi_1) \cap Var(\Phi_1)$. Soit Φ_0 un arrangement sur V, i.e. un ensemble tel que pour $x, y \in V$ soit $x = y \in \Phi_0$ soit $x \neq y \in \Phi_0$. Il est évident que $\Phi_1 \cup \Phi_2$ est $T_1 \cup T_2$-satisfiable si et seulement si $\Phi_1 \cup \Phi_2 \cup \Phi_0$ est $T_1 \cup T_2$-satisfiable pour un certain

arrangement Φ_0. Or $\Phi_1 \cup \Phi_2 \cup \Phi_0$ est $T_1 \cup T_2$-satisfiable si et seulement si $\Phi_1 \cup \Phi_0$ est T_1-satisfiable dans un modèle $\mathcal{M}_1$ et $\Phi_2 \cup \Phi_0$ est T_2-satisfiable dans un modèle $\mathcal{M}_2$ tels que les domaines de $\mathcal{M}_1$ et $\mathcal{M}_2$ ont la même cardinalité. Nous allons montrer que de tels modèles peuvent être exhibés quand on considère les théories satisfaisant les hypothèses de la proposition.

Puisque le problème de satisfiabilité de cardinalité infinie est décidable dans T_i, pour $i = 1, 2$, nous avons un algorithme pour faire l'analyse des cas suivants :

- $\Phi_i \cup \Phi_0$ est T_i-satisfiable dans un modèle infini, pour $i = 1, 2$. Donc Φ est $T_1 \cup T_2$-satisfiable dans un modèle infini.
- $\Phi_1 \cup \Phi_0$ est T_1-satisfiable mais n'est pas T_1-satisfiable dans un modèle infini. Cela implique que Φ n'est pas $T_1 \cup T_2$-satisfiable dans un modèle infini. Par compacité, nous savons que la cardinalité de tous les modèles de $T_1 \cup T_2$ satisfaisant Φ est bornée par un entier positif. Maintenant, il suffit de trouver un modèle fini $\mathcal{M}_1$ de T_1 satisfaisant $\Phi_1 \cup \Phi_0$ et un modèle fini $\mathcal{M}_2$ de T_2 satisfaisant $\Phi_2 \cup \Phi_0$ tels que $\mathcal{M}_1$ et $\mathcal{M}_2$ ont la même cardinalité. Nous procédons de façon suivante :

  1. Calculer le nombre N tel que la cardinalité de tous les modèles de $T_1 \cup T_2$ satisfaisant Φ est bornée par N. En effet, N est effectivement calculable car il suffit de calculer le plus petit i tel que $\Phi \cup \Phi_0 \cup \exists^{\geq i}$ est T_1-insatisfiable : pour $i = 1, \ldots$, on appelle la procédure de T_1-satisfiabilité sur $\Phi \cup \Phi_0 \cup \exists^{\geq i}$, si $\Phi \cup \Phi_0 \cup \exists^{\geq i}$ est T_1-insatisfiable alors on retourne i.
  2. Pour $n = 1, 2, \ldots, N$: nous choisissons une $\Sigma_1 \cup \Sigma_2$-structure $\mathcal{M}$ de cardinalité n et une valuation α de $Var(\Phi_1) \cup Var(\Phi_2)$ dans M et vérifions si $\Phi_i \cup \Phi_0 \cup Diag((\mathcal{M}^{\Sigma_i}, \alpha))$ est T_i-satisfiable, pour $i = 1, 2$. Si oui, par la Proposition 7.7, nous savons que $\mathcal{M}^{\Sigma_i}$ est un modèle de T_i et de plus $\mathcal{M}^{\Sigma_i}$ satisfait $\Phi_i \cup \Phi_0$. Ceci implique que Φ est satisfiable dans un modèle fini de $T_1 \cup T_2$. Sinon, nous essayons une autre $\Sigma_1 \cup \Sigma_2$-structure de cardinalité n.

  La procédure est terminante car il existe un nombre fini de structures dont la cardinalité est bornée par un entier positif, modulo isomorphisme.

□

De façon similaire au cas convexe, nous pouvons dériver une procédure de combinaison (voir Figure 7.2) à partir de la Proposition 7.8, dans laquelle nous avons la même étape *purification*. Seule l'étape *test* est différente de la procédure non déterministe du cas convexe.

7.4 Discussion

Dans un premier temps, nous avons étendu aux théories convexes non stablement infinies les résultats de modularité du Chapitre 6 concernant le problème de satisfiabilité, le problème du mot uniforme et le canoniseur étendu. Nous sommes partis de l'observation que si une formule est satisfiable dans un modèle non trivial d'une théorie convexe alors elle est satisfiable dans un modèle infini de cette théorie. Notre méthode de combinaison est basée sur la propagation d'égalités entre variables partagées et une diségalité qui sert à vérifier l'existence de modèle trivial ou de modèle infini. En supposant que nous savons si les théories composantes ont un modèle trivial ou non, on peut se passer de l'hypothèse de stable infinité. Une conséquence directe de

Purification Identique au cas convexe.

Test Soit $\Phi_1 \cup \Phi_2$ une conjonction de littéraux obtenue par l'étape de purification. Soit $V = Var(\Phi_1) \cap Var(\Phi_2)$.

1. Pour chaque relation d'équivalence E sur les variables dans V, nous construisons un *arrangement* sur E, défini par

$$arr(V, E) = \{x = y \mid (x, y) \in E\} \cup \{x \neq y \mid (x, y) \in (V \times V) \setminus E\},$$

   Si $\Phi_i \cup arr(V, E)$ est T_i-satisfiable pour $i = 1, 2$, alors

   a. si $\Phi_i \cup arr(V, E)$ est T_i-satisfiable dans un modèle infini pour $i = 1, 2$, aller à l'étape 3.

   b. sinon, pour $k = 1, 2, \ldots$, si $T_i \cup \Phi_i \cup arr(V, E) \cup \{\bigwedge_{1 \leq j \neq k \leq k}(x_i \neq x_j)\}$ est insatisfiable alors soit $N = k$. Aller à l'étape suivante c.

   c. pour $\kappa = 1, 2, \ldots N$, et pour chaque $\Sigma_1 \cup \Sigma_2$-structure $\mathcal{M}$ de cardinalité κ et pour chaque valuation α de $Var(\Phi_1) \cup Var(\Phi_2)$ dans M, si $T_i \cup \Phi_i \cup \Phi_0 \cup Diag((\mathcal{M}^{\Sigma_i}, \alpha))$ est satisfiable, pour $i = 1, 2$, aller à l'étape 3.

2. Retourner `insatisfiable`.

3. Retourner `satisfiable`.

FIGURE 7.2 – Combinaison de théories universelles

notre résultat est que les théories convexes finiment axiomatisées peuvent être combinées car nous savons toujours si ces théories ont un modèle trivial ou non. Nous avons ensuite proposé deux procédures de combinaison : une procédure non déterministe et une autre déterministe.

Dans un second temps, nous avons généralisé notre méthode combinaison aux théories universelles en imposant que le problème de satisfiabilité dans des modèles infinies de ces théories soit décidable. Le point de départ de cette étude est l'observation suivante : si un ensemble de formules n'as pas de modèle infini alors la cardinalité de tout modèle de celui-ci est bornée par un entier positif. Cette observation nous ouvre la possibilité de faire un test de type force brute si un modèle fini de la théorie satisfait une formule. En exploitant l'universalité des théories composantes et la notion de diagramme, nous avons donné une procédure de satisfiabilité pour l'union des théories.

Récemment, une série de travaux ont eu pour but de relâcher l'hypothèse de stable infinité dans la méthode de combinaison de Nelson-Oppen. Dans un style de combinaison asymétrique, i.e. les hypothèses faites sur les théories composantes ne sont pas les mêmes, les travaux de Tinelli et Zarba dans [TZ05] permettent de montrer la décidabilité de l'union de la théorie de l'égalité et d'une théorie décidable arbitraire (montré également par Ganzinger dans [Gan02]). Ce résultat a été également généralisé dans [TZ05] à la combinaison d'une théorie dite "shiny" avec une théorie décidable arbitraire. Dans le même esprit de combinaison asymétrique, les travaux [RRZ05, FRZ05, FG04, Fon04] proposent la combinaison de théories des structures de données avec une théorie des éléments arbitraire. Dans le cadre de la combinaison

symétrique, i.e d'un côté il n'y a pas d'hypothèse sur une des théories composantes autre que la décidabilité du problème de satisfiabilité mais on suppose une hypothèse beaucoup plus forte que la stable infinité sur l'autre théorie, [BGN$^+$06] classifie les théories composantes par l'existence de modèles finis ou/et infinis. Toujours dans ce cadre, [Zar04] s'intéresse à la combinaison des théories universelles mais la méthode de combinaison proposée ne fournit qu'une semi-procédure de décision car celle-ci ne termine pas en général.

Nos résultats s'apparentent à la méthode de combinaison proposé dans [BGN$^+$06], qui consiste à classifier les théories par la décidabilité du problème de satisfiabilité dans des modèles infinis et finis. La différence entre notre méthode de combinaison et celle de [BGN$^+$06] est que notre hypothèse de convexité permet de se passer de l'hypothèse de décidabilité de la satisfiabilité dans un modèle infini et fini tandis que l'hypothèse d'universalité implique que la satisfiabilité dans un modèle fini est décidable.

Un problème intéressant est de déterminer les situations où on peut se passer de l'hypothèse de satisfiabilité dans des modèles infinis et arbitrairement finis. On peut imaginer que les théories axiomatisées par un ensemble fini de clauses puissent être dans ce cas. Nous avons observé que si C est une clause satisfiable, qui n'a pas de modèle infini, alors la cardinalité de tout modèle de C est bornée par le nombre d'occurrences de variables dans C. Si cette propriété est encore vraie pour un ensemble fini de clauses, on peut utiliser cette méthode suivante pour décider la satisfiabilité dans un modèle infini. Soient T une théorie axiomatisée par un ensemble fini $Ax(T)$ de clauses et S un ensemble quelconque de littéraux clos. Considérons la borne supérieure n correspondant au nombre d'occurrences de variables dans $Ax(T)$. Alors S est satisfiable dans un modèle infini de T si et seulement si $S \cup \exists^{\geq n+1}$ est T-satisfiable. A partir de ces observations, nous conjecturons le résultat suivant.

Conjecture 7.1. *Soit T une théorie axiomatisée par un ensemble fini $Ax(T)$ de clauses. Si le problème de satisfiabilité est décidable dans T, alors le problème de satisfiabilité de cardinalité infinie est décidable dans T.*

Chapitre 8

Integration of satisfiability procedures in SMT solvers

Préface

Le problème de décider la satisfiabilité d'une formule sans quantificateur par rapport à une théorie en arrière-plan, appelé Satisfiabilité Modulo Théories (SMT), est reconnu comme essentiel dans le domaine de la vérification. La théorie de l'égalité, l'arithmétique linéaire et leur combinaison sont des théories particulièrement pertinentes. Des capacités de représentation allant au delà de la logique propositionnelle permettent une modélisation naturelle de nombreux problèmes réels, notamment pour les obligations de preuve impliquant la présence de bogues dans les systèmes logiciels. En pratique, c'est très souvent un mélange de plusieurs théories qui permet de résoudre un problème donné. Par exemple, le *model-checking* borné, appliqué à des systèmes *hardware*, profite avantageusement de l'utilisation du mélange de la théorie de l'égalité et d'un fragment de l'arithmétique linéaire pour modéliser des structures de données standards comme les files. Etre capable de décider de telles combinaisons automatiquement permet d'éviter de passer manuellement à des problèmes de vérification plus élémentaires, ce qui permet d'accroître la capacité des outils de vérification et d'avoir une meilleure productivité dans la conception.

Une approche proéminente pour SMT, adoptée dans plusieurs outils de vérification (comme MathSat [BBC+06], DPLL(T) [GHN+04], ICS [FORS01], CVC-Lite [BB04], haRVey [DR03], et Zapato [BCLZ04] pour en nommer certains), est basée sur l'intégration d'un solveur Booléen (capable d'énumérer les assignations booléennes) et d'une procédure de satisfiabilité pour une théorie en arrière-plan. Une technique importante pour obtenir un outil SMT efficace consiste à engendrer des (ensembles représentatifs de) *conflits* ("conflict set" en anglais) comme effet de bord de la procédure de satisfiabilité car leur utilisation permet d'élaguer énormément l'espace de recherche géré par le solveur Booléen. Des travaux ont été réalisés pour étendre, par la production de conflits, des procédures de satisfiabilité pour certaines théories (dont la théorie de l'égalité) [Fon04, dMRS04, NO05, ST05]. Par contre, il n'existe pas (à notre connaissance) de travaux publiés sur la construction modulaire de conflits pour des mélanges de théories, même si des systèmes SMT doivent d'une façon ou d'une autre implanter cette fonctionnalité. Ainsi, une personnes souhaitant intégrer cette fonctionnalité dans son propre système SMT doit comprendre le code d'autres systèmes (qui n'est

pas toujours disponible) afin d'en extraire les idées essentielles et les mettre en oeuvre dans son architecture.

Dans ce chapitre, nous présentons comment étendre la méthode de combinaison de Nelson-Oppen de manière à construire une procédure de satisfiabilité produisant des conflits pour le mélange de théories (disjointes), lorsque sont connues des procédures appropriées pour les théories composantes. Dans ce but, nous introduisons le concept de *graphe d'explication*, qui encode de façon compacte les égalités élémentaires déduites. Nous montrons l'intérêt de ce concept pour construire une procédure de satisfiabilité pour la théorie de l'égalité augmentée par la capacité à produire des conflits. Nous montrons ensuite comment combiner des procédures de satisfiabilité, appelées *moteurs d'explication*, qui ont la capacité de construire des graphes d'explication. Ceci permet de construire une procédure de satisfiabilité avec la même capacité pour le mélange des théories. Pour finir, nous introduisons le concept de *quasi-conflit* ("quasi-conflict set" en anglais), qui nous permet de caractériser précisement une forme de minimalité satisfaite par les explications construites par notre méthode.

Notre méthode d'explication est appliquée à la clôture de congruence et à l'élimination de Gauss.

8.1 Introduction

Deciding the satisfiability of a quantifier-free formula with respect to a background theory T, called Satisfiability Modulo Theories (SMT), is being recognized as crucial for verification. Notable theories of interest are equality (or, equivalently, the theory of uninterpreted functions), Linear Arithmetic, and their combination. In fact, representation capabilities beyond propositional logic allow for a natural modeling of a number of real-world problems, e.g., verification of pipelines or discharging proof obligations which imply the presence of bugs in software systems. Particularly relevant is the combination of (at least) two theories T_1 and T_2 since it is often the case that heterogeneous information need to be taken into account to solve a certain problem. For instance, the bounded model checking of a large class of hardware designs greatly benefit from using the union of the theory of uninterpreted functions and some fragment of Linear Arithmetic to model commonly used data structures such as queues. Being able to decide such combinations automatically avoids the need for manual breakdown of verification problems into subparts, and results in augmented capacity of verification tools, and higher design productivity.

A prominent approach to SMT, adopted in several verification tools (e.g., MathSat [BBC+06], DPLL(T) [GHN+04], ICS [FORS01], CVC-Lite [BB04], haRVey [DR03], and Zapato [BCLZ04] to name but a few), is based on the integration of a Boolean solver (capable of enumerating boolean assignments) and a satisfiability procedure for the background theory. One of the key technique to obtain an efficient SMT tool is the generation of *conflict sets* by the satisfiability procedure and their use to dramatically prune the search space of the Boolean solver. While there has been some work on extending satisfiability procedures for some theories (mainly equality) [Fon04, dMRS04, NO05, ST05], there has been no published work (to the best of our knowledge) on the modular construction of conflict sets in unions of theories. Many of the SMT systems listed above have implemented this capability somehow, but no one has offered a high level description of how this is done. So, implementors desiring to build such a capability in their own SMT tool are required to understand the code of other systems (and in many cases the code is not even available), abstract away unimportant implementation details, and finally adapt the ideas to their architecture.

In this work, we provide an abstract account of how to extend the well-known combination method by Nelson and Oppen [NO79] to build a satisfiability procedure capable of producing conflict sets in the union of theories T_1 and T_2, whenever the satisfiability procedures for T_1 and T_2 provide certain interface capabilities. To this end, we first introduce the concept of *explanation graph* (Section 8.3), a data structure which compactly encodes the fact that a certain equality between variables (called elementary equality) is a logical consequence of a set of elementary equalities. We also provide evidence that such a data structure can be profitably used to build satisfiability procedures for the theory of equality with the capability of producing conflict sets in a very simple way. Then (Section 8.5), we show how to combine satisfiability procedures, called *explanation engines*, capable of building explanation graphs so to obtain a satisfiability procedure with the capability of producing conflict sets in the union of the component theories. In Section 8.6, we introduce the concept of *quasi-conflict set*, which allows us to precisely characterize a form of minimality satisfied by the explanations computed by our combination method.

8.2 Background

First-Order Logic. We assume the usual syntactic and semantic notions of term, formula, interpretation, satisfiability, etc. for first-order logic. Since we are concerned with satisfiability, we may consider all symbols with arity 0 as constants since a formula is equisatisfiable to its existential closure and existentially quantified variables can be replaced by Skolem constants. This explains why we may talk about constants instead of variables depending on the context. In this work, we will be concerned with the following theories :

- The theory of equality $\mathcal{E}$ whose signature contains a finite set of function and constant symbols, and such that the equality symbol $=$ is interpreted as the identity relation.
- The theory $\mathcal{E}^c$ consisting only of equalities or disequalities between constants. We will also refer to the literals of $\mathcal{E}^c$ as *elementary* equalities and disequalities and to $\mathcal{E}^c$ as the *theory of pure equality.*
- The quantifier-free fragment of Linear Rational Arithmetic consisting only of equalities or disequalities is denoted with $\mathcal{LA}$.

As a consequence, we will consider literals of the form $s = t$ and $\neg s = t$ (also abbreviated with $s \neq t$) only, for suitable terms s and t. A set of literals of the form $\Omega = \{x_1 = t_1, \ldots, x_n = t_n\}$ is said in *solved form* if $x_1, \ldots, x_n$ are distinct variables occurring only once in Ω. The solved form is *flat* if $t_1, \ldots, t_n$ are flat terms, where a *flat term* is a term $f(s_1, \ldots, s_m)$ such that f is a m-ary function symbol and s_i is a variable/constant for $i = 1, \ldots, m$. In the following, φ denotes an arbitrary set of literals, Γ denotes a set of equalities, Ω denotes a solved form, E denotes a set of elementary equalities, and Δ_V denotes a set of elementary disequalities. The set of variables occurring in φ is denoted by $Var(\varphi)$. The reflexive, symmetric and transitive closure of E is denoted by E^*.

Satisfiability procedures and conflict sets. The *satisfiability problem* for a theory T amounts to establishing whether any (finite) quantifier-free conjunction of literals (or equivalently, any finite set of ground literals) is T-satisfiable or not. A *satisfiability procedure* for T is any algorithm that solves the satisfiability problem for T. A *T-conflict set* is a T-unsatisfiable set of literals. A T-conflict set CS of literals is *minimal* if there is no $CS' \subset CS$ such that CS' is a T-conflict set. A *T-explanation* of an equality e is a T-satisfiable set Γ of equalities such that $T \cup \Gamma \models e$. A T-explanation of e is *minimal* if there is no $\Gamma' \subset \Gamma$ such that $T \cup \Gamma' \models e$. [19]

Proposition 8.1. *A T-satisfiable set of equalities Γ is a minimal T-explanation for an equality e iff $\Gamma \cup \{\neg e\}$ is a minimal T-conflict set.*

Nelson-Oppen combination schema. For simplicity, we will only consider convex and stably infinite theories in this work. A set φ of T-literals is *convex* iff for any disjunction $x_1 = y_1 \vee \cdots \vee x_n = y_n$ for $n > 1$, where x_i, y_i are constants $(i = 1, ..., n)$ we have that $T \cup \varphi \models x_1 = y_1 \vee \cdots \vee x_n = y_n$ iff $T \cup \varphi \models x_i = y_i$ for some $i \in \{1, ..., n\}$. A theory T is *convex* iff all sets of T-literals are convex. We say that T is *stably infinite* if for every T-satisfiable set of literals φ there exists a model of T satisfying φ such that its interpretation domain is infinite. For example, $\mathcal{E}^c$, $\mathcal{E}$ and $\mathcal{LA}$ are convex

19. We will omit the theory T when it is clear from the context.

and stably infinite theories. The Nelson-Oppen combination method assumes that component theories are signature-disjoint, convex, and stably infinite.

Proposition 8.2. *If T is a convex theory, then a minimal conflict set contains at most one disequality.*

For example, $\mathcal{E}^c$ and $\mathcal{E}$ are convex theories such that any set of equalities is satisfiable. [20] Notice that $\mathcal{LA}$ does not satisfy this property, for example $\{x-y=3, x-y=2\}$ is a (minimal) $\mathcal{LA}$-conflict set.

Graphs. We use some standard concepts about graphs such as undirected graph, acyclic graph, sub-graph, connected graph, tree, forest, path, simple path, elementary path, connected components.

In the sequel, we consider only acyclic undirected graphs, which will be often called graphs for the sake of simplicity. An undirected graph G is a pair (V, E) where V (also written as $Vertex(G)$) is a finite set and E (also written as $Edge(G)$) is a set of unordered pairs written as (v, w) or $v = w$ for v, w in V. Note that unordered pairs in E can be considered as elementary equalities, thus we use E for both sets of elementary equalities and sets of edges. $G^V_\emptyset$ denotes the graph whose vertices are in the set V which are connected by no edges, i.e. $G^V_\emptyset = (V, \emptyset)$. The "is a subgraph of" relation is denoted by $\subseteq$. The set $ElemPath(G, x, y)$ denotes the set of edges in an elementary path between x and y in a graph G, i.e. if $v_0, ..., v_n$ is an elementary path where v_0 is x and v_n is y, then $ElemPath(G, x, y)$ is the set of edges $v_{i-1} = v_i$, for $i = 1, ..., n$. Given two different vertices x and y, $ElemPath(G, x, y)$ is empty if and only if x and y are not in the same connected component of G. The set of *pairs of connected vertices in* G is $CP(G) = \{x = y \mid x, y \in V \text{ and } ElemPath(G, x, y) \neq \emptyset\}$. Given an acyclic undirected graph $G = (V, E)$, one can remark that $E^* = CP(G) \cup \{x = x \mid x \in V\}$. In the following, we consider that a graph G is equipped with a labelling mapping $\mathcal{L}_G$, which is a mapping from $Edge(G)$ to a set of sets of literals. If $G = (V, E)$ is a (labelled acyclic undirected) graph, x and y are two different vertices in V such that $x = y \notin CP(G)$, and L is a set of literals, then

$$Insert(G, x = y, L)$$

denotes the (labelled acyclic undirected) graph $G' = (V, E \cup \{x = y\})$ where $\mathcal{L}_{G'}$ is such that $\mathcal{L}_{G'}(x = y) = L$ and $\forall e \in Edge(G), \mathcal{L}_{G'}(e) = \mathcal{L}_G(e)$.

8.3 Explanation Graphs

We consider a graph data-structure encoding the entailment of elementary equalities. An elementary equality corresponds to a labelled edge whose label is a satisfiable set of literals implying the elementary equality. In the label, we may find literals occurring in the input formula as well as auxiliary elementary equalities whose entailment is given by the graph. To avoid redundancy, we only consider acyclic graphs.

Definition 8.1. *Let φ be a set of literals, $G = (V, E)$ be an acyclic undirected graph, and $<$ be an ordering relation on E. G is an* explanation graph *of φ w.r.t. $<$ if (i)*

20. This holds for theories axiomatized by a finite set of equalities.

V is the set of constants occurring in φ, (ii) there exists a labelling function $\mathcal{L}_G$ with domain E and codomain $\mathbf{2}^{\varphi \cup CP(G)}$, [21] *(iii) the following properties are satisfied for any $v_1 = v_2 \in E$:*

(iii.a) $\mathcal{L}_G(v_1 = v_2)$ is satisfiable and $\mathcal{L}_G(v_1 = v_2) \models v_1 = v_2$, [22]

(iii.b) for each $v_1' = v_2'$ in $\mathcal{L}_G(v_1 = v_2) \backslash \varphi$ we have that $e < (v_1 = v_2)$, for any e in $ElemPath(G, v_1', v_2')$.

The set of literals of φ in G *is* $Lit(G) = \varphi \cap (\bigcup_{e \in E} \mathcal{L}_G(e))$. *The* set of elementary equalities of G *is* $Eq(G) = \bigcup_{x=y \in E} \{x = y\}$.

An edge $v_1 = v_2 \in E$ is minimally explained *if $\mathcal{L}_G(v_1 = v_2)$ is a minimal explanation for $v_1 = v_2$. An explanation graph is said* edge-minimal *if all its edges are minimally explained.*

It is important to consider explanation graphs from which all possible entailed elementary equalities can be extracted. For convex theories, such "complete" explanation graphs allow us to process first equalities and then elementary disequalities.

Definition 8.2. *Let T be a theory and let φ be a T-satisfiable set of literals. A set of elementary equalities E is* complete *for φ (modulo T) if*

$$\forall x, y \in Var(\varphi),\ T \models \varphi \Rightarrow x = y \text{ iff } (x, y) \in E^*$$

An explanation graph G of a T-satisfiable set of literals φ is said complete *(modulo T) if $Eq(G)$ is complete for φ (modulo T).*

8.4 Explaining Satisfiability Procedures

Below, we discuss the problem of constructing a complete explanation graph. A naïve brute-force method consists in considering each possible elementary equality made of disconnected vertices of the graph one after the other. But there are also satisfiability procedures with the capability of generating complete edge-minimal explanation graphs without resorting to such a brute-force method and are thus more efficient.

8.4.1 Explaining Congruence Closure

For the theory of equality $\mathcal{E}$, a complete edge-minimal explanation graph can be built by a suitable extension of the Congruence Closure algorithm (see, e.g., [NO80] for details). The explanation graph is updated each time an equality is processed. When all equalities are processed, we get a complete edge-minimal explanation graph.

The corresponding Congruence Closure algorithm is presented in Figure 8.1 : given a set Ω in solved form and a set of elementary equalities E, $CC(\Omega, E)$ defined by

$$(\mathsf{Init}^* \cdot (\mathsf{Ins} \cdot \mathsf{Cong}^* + \mathsf{Skip})^*)(\ \Omega; E; G_\emptyset^{Var(\Omega \cup E)}\ )$$

returns a complete edge-minimal explanation graph of $\Omega \cup E$.

21. Let X be a set, $\mathbf{2}^X$ denotes the powerset of X.

22. For the sake of simplicity, we do not refer explicitly to a theory, but satisfiability and validity should be understood as modulo the theory to which the literals in φ belong.

Init

$$\frac{\Omega; E; G}{\Omega; E; Insert(G, z = z', \{z = t, z' = t\})}$$

$$\textbf{if} \begin{cases} z = t, z' = t \in \Omega \\ (z, z') \notin CP(G) \end{cases}$$

Ins

$$\frac{\Omega; E \cup \{x = x'\}; G}{\Omega; E; Insert(G, x = x', \{x = x'\})}$$

$$\textbf{if} \begin{cases} x \neq x' \\ (x, x') \notin CP(G) \end{cases}$$

Skip

$$\frac{\Omega; E \cup \{x = x'\}; G}{\Omega; E; G} \quad \textbf{if } (x, x') \in CP(G)$$

Cong

$$\frac{\Omega; E; G}{\Omega; E; Insert(G, z = z', \{z = f(y_1, \ldots, y_n), z' = f(y'_1, \ldots, y'_n)\} \cup \bigcup_{j \in J} \{y_j = y'_j\})}$$

$$\textbf{if} \begin{cases} z = f(y_1, \ldots, y_n), z' = f(y'_1, \ldots, y'_n) \in \Omega \\ z \neq z', (z, z') \notin CP(G) \\ I, J \text{ is a partition of } \{1, \ldots, n\} \text{ such that } J \neq \emptyset \text{ and} \\ (\forall i \in I : y_i = y'_i), (\forall j \in J : (y_j, y'_j) \in CP(G)) \end{cases}$$

FIGURE 8.1 – Union-Find and Congruence Closure with Explanation

For efficiency considerations, the interested reader is pointed to [NO05] where entailment of equalities is encoded by a proof forest. Such a notion is quite similar to that of explanation graph. There is just a slight difference in the labelling of edges. In a proof forest, an edge (x, y) is labelled by $\{x = y\}$ or by $\{x = f(x_1), y = f(y_1)\}$, where the equality $x_1 = y_1$ does not appear explicitly in the label. In an explanation graph, the label would be $\{x = f(x_1), y = f(y_1), x_1 = y_1\}$. In the simple case of the theory of pure equality $\mathcal{E}^c$, labels are restricted to elementary equalities and collecting elementary equalities in the labels of a path from x to y yields a minimal explanation of $x = y$. The explanation graph is built thanks to a Union-Find algorithm. Given a set of elementary equalities E, $UF(E)$ defined by

$$((\mathsf{Ins} + \mathsf{Skip})^*)(\ \emptyset; E; G_\emptyset^{Var(E)}\ )$$

returns a complete edge-minimal explanation graph of E.

8.4.2 Explaining Gauss Elimination

For the important case of Linear Arithmetic, Gauss elimination can also be extended to generate minimal conflict sets and minimal explanations for elementary equalities. It is important to notice that explanation graphs can be used to store the (explained) entailment of elementary equalities.

Our satisfiability procedure for $\mathcal{LA}$ is based on the Gauss elimination algorithm (see e.g. [Sch86]) and will be denoted with GA. It takes as input a set of linear equalities and disequalities and it is capable of (1) checking for their (un-)satisfiability, (2) returning a minimal conflict set, and (3) deriving elementary equalities which are entailed by the input set of literals. The algorithm resembles that in [BLH96] but it differs in three points. First, GA directly handles disequalities rather than resorting to the Simplex as in [BLH96]. Second, the technique to build minimal conflict sets is abstractly described in terms of linear combinations of equalities rather than using matrices as in [BLH96]. As already observed e.g. in [Bjø98], this makes it easier to use the algorithm in theorem provers where sparse systems of constraints are expected and a list-based representation of arithmetic constraints is usually adopted. Third, GA is capable of computing minimal explanations of elementary equalities. This is a crucial feature to construct a complete edge-minimal explanation graph as a post-processing of GA.

Satisfiability checking

The input to GA has the form $\Gamma|\Delta$ where Γ is a set of linear equalities and Δ is a set of disequalities. Observe that $l \neq r$ is equisatisfiable to $l - r = s \wedge s \neq 0$ where s is a fresh variable, also called *slack variable*. The first step of GA consists in replacing each disequality $l \neq r$ in Δ with the equality $l - r = s$, where s is a slack variable. From $\Gamma|\Delta$, we obtain a system $\Gamma|\Gamma'$ of equalities. The second step of GA is to perform Gauss elimination on $\Gamma|\Gamma'$ with the proviso that two equalities in Γ' should never be linearly combined [23] (in other words, only combinations of two equalities in Γ or an equality in Γ and one in Γ' are allowed). If we obtain an equality of the form $0 = c$ (where c is a non zero constant) from Γ, then GA returns the $\mathcal{LA}$-unsatisfiability of $\Gamma \cup \Delta$. Also, if we obtain an equality of the form $s = 0$ from Γ' where s is a slack variable, then GA returns the $\mathcal{LA}$-unsatisfiability of $\Gamma \cup \Delta$. Otherwise, GA returns the $\mathcal{LA}$-satisfiability of $\Gamma \cup \Delta$.

Generating minimal conflict sets

When considering a new equality e, GA labels it with a unique identifier ℓ producing $\ell : e$. If $\ell_1 : t_1 = 0$ and $\ell_2 : t_2 = 0$ are linearly combined so to obtain $t_1 + c * t_2 = 0$, then its label will be the expression $\ell_1 + c * \ell_2$. We also assume that expressions for labels are simplified according to the usual arithmetic rules. In this way, when an unsatisfiable equality $\ell : 0 = c$ (with $c \neq 0$) or $\ell : s = 0$ (with s a slack variable) is detected by GA, the identifiers occurring in the expression ℓ will yield a minimal conflict set. To illustrate the idea, consider the following example.

23. We say that $e + k * e'$ (for some number $k \neq 0$) is a linear combination of the equality e of the form $a_1 * x_1 + \cdots + a_n * x_n = b$ and the equality e' of the form $a'_1 * x_1 + \cdots + a'_n * x_n = b'$ if it is the result of simplifying the equality $a_1 * x_1 + \cdots + a_n * x_n + k * (a'_1 * x_1 + \cdots + a'_n * x_n) = b + k * b'$.

Exemple 8.1. *Let*

$$\begin{aligned}
&\ell_1 : x_4 = x_1\\
&\ell_2 : x_5 = x_2\\
&\ell_3 : x_6 = x_3\\
&\ell_4 : x_5 + x_4 = 4\\
&\ell_5 : x_7 = x_4\\
&\ell_6 : x_7 = x_5\\
&\ell_7 : x_6 = 2\\
&\ell_8 : x_1 \neq x_2
\end{aligned}$$

be the input to GA.

The result of internalizing the above input literals is

$$\begin{aligned}
&\ell_1 : x_4 = x_1\\
&\ell_2 : x_5 = x_2\\
&\ell_3 : x_6 = x_3\\
&\ell_4 : x_5 + x_4 = 4\\
&\ell_5 : x_7 = x_4\\
&\ell_6 : x_7 = x_5\\
&\ell_7 : x_6 = 2\\
&\ell_8 : x_1 - x_2 = s
\end{aligned}$$

Then, after performing the various linear combinations, we get

$$\begin{aligned}
\ell_1 &: x_4 = x_1\\
\ell_2 &: x_5 = x_2\\
\ell_3 &: x_6 = x_3\\
\ell_1 + \ell_2 - \ell_4 &: x_1 + x_2 = 4\\
\ell_5 - \ell_1 &: x_7 = x_1\\
0.5\ell_6 - 0.5\ell_5 + 0.5\ell_4 - \ell_1 &: x_1 = 2\\
\ell_7 &: x_6 = 2\\
\ell_8 + \ell_6 + \ell_2 - \ell_5 - \ell_1 &: s = 0
\end{aligned}$$

The unsatisfiable equation $s = 0$ (since s is a slack variable) has the expression $\ell_8 + \ell_6 + \ell_2 - \ell_5 - \ell_1$ as its label and so the minimal conflict is $\{\ell_1, \ell_2, \ell_5, \ell_6, \ell_8\}$.

Explaining elementary equalities

Observe that if an equality $x = y$ is entailed by a system of linear equalities $t_i = 0$ for $i = 1, \ldots, n$, then $x - y$ must be a linear combination of t_i, $i \in \{1, \ldots, n\}$, i.e. $x - y = \sum_{i \in \{1,\ldots,n\}} \lambda_i * t_i$ such that $\lambda_k \neq 0$ for (at least one) k in $\{1, \ldots, n\}$. When GA cannot perform linear combinations any more, it back-substitutes variables in order to compute the most general solution of the system. As a consequence, equalities of

the form $\ell'_i : x_i = t_i$ for $i = 1, ...n$ are obtained where x_i does not occur in t_i. Then, GA performs a guessing step, i.e. it picks two equalities $\ell'_i : x_i = t_i$ and $\ell'_j : x_j = t_j$ $(i \neq j)$ and checks the syntactical identity of t_i and t_j. If this is the case, then $x_i - x_j = 0$ is entailed by the set of literals and its label is $\ell'_i - \ell'_j$.

Exemple 8.2. *To illustrate, consider again Example 8.1 without the disequality $x_1 \neq x_2$. So, let*

$$\begin{aligned}
&\ell_1 : x_4 = x_1\\
&\ell_2 : x_5 = x_2\\
&\ell_3 : x_6 = x_3\\
&\ell_4 : x_5 + x_4 = 4\\
&\ell_5 : x_7 = x_4\\
&\ell_6 : x_7 = x_5\\
&\ell_7 : x_6 = 2
\end{aligned}$$

be the input to GA.

After performing all the possible linear combinations and back-substituting, we obtain

$$\begin{aligned}
\ell_1 &: x_4 = x_1\\
\ell_2 &: x_5 = x_2\\
\ell_3 &: x_6 = x_3\\
0.5\ell_5 + 0.5\ell_4 - \ell_2 - 0.5\ell_6 &: x_2 = 2\\
\ell_5 - \ell_1 &: x_7 = x_1\\
0.5\ell_6 - 0.5\ell_5 + 0.5\ell_4 - \ell_1 &: x_1 = 2\\
\ell_7 &: x_6 = 2
\end{aligned}$$

It is easy to see that $\ell_6 - \ell_5 - \ell_1 - \ell_2 : x_1 - x_2 = 0$ is entailed by $\ell_1, ..., \ell_7$ since the right hand sides of the labeled equalities, namely $0.5\ell_6 - 0.5\ell_5 + 0.5\ell_4 - \ell_1 : x_1 = 2$ and $0.5\ell_5 + 0.5\ell_4 - \ell_2 - 0.5\ell_6 : x_2 = 2$ are identical. So, the minimal explanation of $x_1 = x_2$ is $\{\ell_1, \ell_2, \ell_5, \ell_6\}$.

Correctness of GA

We now prove the correctness of the GA algorithm. Recall that the correctness of Gauss elimination is based on the basic properties of *Elementary Row Operations* which do not change the solution of the system [Sch86] : (i) *Interchange* : the order of equations can be interchanged ; (ii) *Scaling* : multiplying an equation by a constant ; (iii) *Replacement* : an equation can be replaced by the sum of that equation and a non zero multiple of any other equation.

Lemma 8.1. *The following properties are ensured by* GA *algorithm :*

- ***LA1** : Replacing any equality of Γ by a "linear combination" of all its equalities does not change the solution of the system.*
- ***LA2** : A redundant equality of Γ (i.e. an equality which can be expressed as a "linear combination" of the others) will be detected during Gauss elimination when obtaining $0 = 0$.*

- ***LA3*** *: Given* Γ *of* $t_i = 0$ *for* $i = 1, ..., n$*, after Gauss elimination, we obtain a system of the form* $t'_i = 0$ *for* $i = 1, \ldots, n$ *and each* t'_i *is a linear combination of* $t_1, ..., t_i$*, i.e. there exist* $\lambda_1, ..., \lambda_i$ *such that* $t'_i = \lambda_1 t_1 + \cdots + \lambda_i t_i$ *with* $\lambda_k \neq 0$ *for (at least one)* $k \in \{1, ..., i\}$*.*
- ***LA4*** *: The unsatisfiability of a set* Γ *of equalities will be detected during Gauss elimination when obtaining* $0 = c$ *with* $c \neq 0$ *(e.g. from* $x - y = 3, x - y = 2$*, you get* $x - y = 3, 0 = -1$*).*

The minimality of the conflict set generated by GA, whenever $\mathcal{LA}$-unsatisfiability is reported, is stated in the following result.

Lemma 8.2. *Given the unsatisfiable set of* k *equalities* $l_1 : e_1, l_2 : e_2, \ldots, l_k : e_k$*, let* $l'_1 : e'_1, l'_2 : e'_2, \ldots, l'_k : e'_k$ *be the set of equalities obtained after running* GA*. If the equality* $l'_h : e'_h$ *for* $h \in \{1, ..., k\}$ *is unsatisfiable (i.e.* e'_h *is of the form* $0 = c$ *and* $c \neq 0$ *or* $s = 0$ *and* s *is a fresh variable), then the set* $CS := \{e_i | l_i : e_i \text{ s.t. } l_i \text{ occurs in } l'_h\}$ *is a minimal conflict set.*

Proof. Without loss of generality we consider e_i is of the form $t_i = 0$. This implies e'_i is of the form $t'_i = 0$. Whenever GA uncovers unsatisfiability, by properties **LA3** and **LA4** we claim that :

$$t'_h = \sum_{e_i \in CS} \lambda_{h,i} t_i = b \text{ if } t_h \text{ does not contain any slack variable}$$

or

$$t'_h = \sum_{e_i \in CS} \lambda_{h,i} t_i = c * s \text{ if } t_h \text{ contains the slack variable } s$$

where $b, c \neq 0$. Roughly speaking, the first situation corresponds to the case in which the system of equalities has no solution while the latter one corresponds to the case when unsatisfiability is due to a disequality. We consider the proof of the first case, the second can be handled similarly.

Assume that CS is not minimal and there exists a set $CS' \subset CS$ st. CS' also a minimal conflict set. The set CS' must contain e_h since the k equalities $e_1, e_2, \ldots, e_k$ are satisfiable. Again, we have

$$t'_h = \sum_{e_i \in CS'} \lambda'_{h,i} t_i = b'$$

Thus

$$t_h = -\frac{1}{\lambda_{h,h}} \sum_{e_i \in CS - \{e_h\}} \lambda_{h,i} t_i + \frac{b}{\lambda_{h,h}}$$

and

$$t_h = -\frac{1}{\lambda'_{h,h}} \sum_{e_i \in CS' - \{e_h\}} \lambda'_{h,i} t_i + \frac{b'}{\lambda'_{h,h}}$$

The k equalities $e_1, e_2, \ldots, e_k$ must have a most general solution since they are satisfiable, say ρ_k. The substitution of ρ_k to t_h will rule out all t_i s.t. $e_i \in CS - \{e_h\}$, that gives

$$t_h \rho_k = \frac{b}{\lambda_{h,h}} = \frac{b'}{\lambda'_{h,h}} = t_h \rho_k$$

Hence, we have

$$\frac{1}{\lambda_{h,h}} \sum_{e_i \in CS - \{e_h\}} \lambda_{h,i} t_i = \frac{1}{\lambda'_{h,h}} \sum_{e_i \in CS' - \{e_h\}} \lambda'_{h,i} t_i$$

which means there is a linear dependency between the equalities of CS since CS' is assumed to be a proper subset of CS by the property **LA1**, and hence CS has redundant equalities. This contradicts the fact that all redundant equalities (i.e equalities entailed by others) will be eliminated during Gauss elimination by the property **LA2**. □

The minimality of explanation of equalities entailed by a satisfiable conjunction of linear equalities and disequalities for the theory of Linear Arithmetic $\mathcal{LA}$ is stated in the following result.

Lemma 8.3. *Given the satisfiable set of k equalities $l_1 : e_1, l_2 : e_2, \ldots, l_k : e_k$, let $l'_1 : e'_1, l'_2 : e'_2, \ldots, l'_k : e'_k$ be the set of equalities obtained after running* GA *and having performed back-substitution. If there exist two equalities $l'_{h_1} : x_h = t_{h_1}$ and $l'_{h_2} : x_{h_2} = t_{h_2}$ such that $h_1 \neq h_2$ and h_i is in $\{1, ..., k\}$ for $i = 1, 2$ and t_{h_1} is syntactically equal to t_{h_2}, then the set $EX^{x=y} := \{e_i | l_i : e_i \text{ s.t. } l_i \text{ occurs in } l'_{h_1} - l'_{h_2}\}$ is a minimal explanation for the equality $x = y$.*

Proof. We assume without loss of generality that e_i is of the form $t_i = 0$. This implies e'_i is of the form $t'_i = 0$. If $x = y$ we have

$$\sum_{e_i \in EX^{x=y}} \lambda_i t_i = x - y$$

that is, $x - y = 0$ is linear combination of the equalities in $EX^{x=y}$. Assume that $EX^{x=y}$ is not minimal and there exists a proper subset of $EX^{x=y}$, say J, which is also a minimal explanation. Then, we have the following relation

$$\sum_{e_i \in J} \lambda'_i t_i = x - y$$

Thus

$$\sum_{e_i \in EX^{x=y}} \lambda_i t_i = \sum_{e_i \in J} \lambda'_i t_i$$

which means that there is a linear dependency between the equalities of $EX^{x=y}$ since J is a proper subset of $EX^{x=y}$. This dependency contradicts the fact that $EX^{x=y}$ has no redundant equalities by the property **LA2** of the Gauss elimination. □

Finally, we have the following result about minimal explanation of unsatisfiability or of entailed elementary equalities.

Theorem 8.1. *Let Γ be a set of linear equalities and Δ be a set of linear disequalities. If $\Gamma \cup \Delta$ is $\mathcal{LA}$-unsatisfiable, then* GA *returns a minimal $\mathcal{LA}$-conflict set. Otherwise (i.e. if $\Gamma \cup \Delta$ is $\mathcal{LA}$-satisfiable),* GA *returns the minimal $\mathcal{LA}$-explanations of all elementary equalities entailed by $\Gamma \cup \Delta$.*

8.4.3 Explaining Difference Constraints

Difference constraints systems consisting of inequalities of the form $x_i - x_j \leq b_{i,j}$ are ubiquitous in verification problems. Obviously, they can be solved by many existing algorithms for solving linear arithmetic constraints, e.g. by Simplex or by Fourier-Motzkin [Sch86], but these methods are often inefficient since they have all exponential time complexity. Pratt [Pra77] showed that a difference constraint can be represented by a weighted directed graph such that a system is feasible (or satisfiable) if and only if there exists no negative weight cycle in the graph. He also gave an $O(n^3)$ algorithm for solving systems of difference constraints, which uses the shortest path algorithm. In [RSJM99], it is proposed an incremental algorithm which processes the addition of one constraint in $O(m + n \log n)$ amortized time, where m is the number of constraints and n is the number of variables, and the removal of one constraint in constant time. In this work, we investigate how graph based algorithms for solving systems of difference constraints can be augmented to produce minimal explanations of unsatisfiability or of elementqry equality entailed by the input system.

A system of difference constraints $\langle V, C\rangle$ consists of a set V of variables and a set C of linear inequalities of the form $x_i - x_j \leq b_{i,j}$, where $x_i, x_j \in V$ and $b_{i,j}$ is a constant. A system $\langle V, C\rangle$ is satisfiable if there exists an assignment of real values to variables in V that satisfies all constraints in C. A directed, weighted graph $G = \langle V, E, length\rangle$ consists of a set of vertices V, a set of edges E, and a function $length$ form E to reals. We denote an edge from vertex u to vertex v by $u \to v$. The constraint graph of a system of difference constraints $\langle V, C\rangle$ is a directed, weighted graph $G = \langle V, E, length\rangle$ such that

$$E = \{x_j \to x_i | x_i - x_j \leq b_{i,j} \in C\}$$
$$length(x_j \to x_i) = b_{i,j} \text{ iff } x_i - x_j \leq b_{i,j} \in C$$

Given a constraint graph $G = \langle V, E, length\rangle$, the corresponding difference constraints system is $\langle V, \{v - u \leq length(u \to v) | u \to v \in E\}\rangle$. Without loss of generality, we assume that a system of difference constraints contains at most one inequality per ordered pair of variables, i.e. the system can not contain two constraints $x - y \leq a$ and $x - y \leq b$, where $a \neq b$.

The following result gives a necessary and sufficient condition for satisfiability of a system of difference constraints.

Theorem 8.2 ([Pra77])**.** *A system of difference constraints is satisfiable if and only if its constraint graph has no negative cycle.*

The next statement, which follows directly from the previous result, gives the minimality of conflict sets when a system of difference constraints is unsatisfiable.

Corollary 8.1. *The system of difference constraints corresponding to a negative cycle in a constraint graph is a minimal conflict set.*

The corollary gives us an easy way to minimally explain unsatisfiability of a system of difference constraints.

The following result provides a method to minimally explain entailed elementary equalities.

Proposition 8.3. *Given a satisfiable system of difference constraints $\langle V, C\rangle$ and its corresponding constraint graph $G = \langle V, E, length\rangle$. For every pair of elements $u, v \in V$, the following conditions are equivalent :*

(i) $C \models u = v$.

(ii) There exist an elementary path $ep(u, v)$ from u to v and an elementary path $ep(v, u)$ from v to u such that

$$\Sigma_{\{e \in ep(u,v)\}} length(e) = \Sigma_{\{e \in ep(v,u)\}} length(e) = 0$$

Proof. $C \models u = v$ if and only if $C \models u \leq v$ and $C \models v \leq u$. But we have that $C \models u \leq v$ if and only if there exists an elementary path $ep(v, u)$ such that

$$\Sigma_{\{e \in ep(v,u)\}} length(e) = 0$$

Similarly, $C \models v \leq u$ if and only if there exist an elementary path $ep(u, v)$ such that

$$\Sigma_{\{e \in ep(u,v)\}} length(e) = 0$$

And the proof is done. □

8.5 Conflict Sets for Combination of Theories

We adapt the deterministic Nelson-Oppen combination method [NO79] to compute $T_1 \cup T_2$-conflict sets, where the theory T_i is convex and stably-infinite, and for which a satisfiability procedure is available ($i = 1, 2$). We assume also that T_1, T_2 are signature-disjoint.

Informally, the Nelson-Oppen combination method consists in exchanging entailed elementary equalities between the two procedures until either unsatisfiability is derived by one of the two, or no more elementary clauses can be exchanged. In the first case, we derive the unsatisfiability of the input formula ; in the second case, we derive its satisfiability. Thus, the unsatisfiability in the union $T_1 \cup T_2$ can be explained according to two kinds of explanations : (i) the explanation of entailed elementary equalities and (ii) the explanation of the unsatisfiability in a component theory. To generate these explanations, the basic idea is to use specialized satisfiability procedures with the capability of generating explanation graphs in order to store entailed equalities. Formally, we call this kind of satisfiability procedures *explanation engines*. Then, we show how to combine these explanation engines.

8.5.1 Explanation Engines

Definition 8.3. *Let T be a theory. A* T-explanation engine *is a computable function μEX such that given a set Ω in solved form of T-equalities and a set E of elementary equalities, $\mu EX(\Omega, E)$ returns a triplet (Ω', E', G) satisfying the following properties :*

- *$\Omega' \subseteq \Omega$,*
- *G is an* **edge-minimal** *explanation graph of $\Omega \cup E$,*
- *E' is a set of elementary equalities such that $E' \subseteq CP(G)$,*
- *if $\Omega \cup E$ is T-unsatisfiable, then $\Omega' \cup E'$ is a* **minimal** *conflict set,*
- *if $\Omega \cup E$ is T-satisfiable, then $\Omega' = \emptyset$, $E'^* \supseteq E^*$, and ($E'^* = E^*$ iff E is complete for $\Omega \cup E$ modulo T).*

It is quite natural to construct explanation engines for many interesting theories. For the theory of equality $\mathcal{E}$, we simply use the complete edge-minimal explanation graph $CC(\Omega, E)$ computed by the congruence closure with explanation. In general, given a T-satisfiability procedure equipped with the capability of computing minimal conflict sets and minimal explanations of entailed elementary equalities, it is always possible to construct an explanation engine. For instance, this construction applies to Linear Arithmetic $\mathcal{LA}$, where Gauss elimination can be modified to return a minimal conflict set (in case of unsatisfiability) or a minimal explanation of an elementary equality (in case of satisfiability).

8.5.2 Combination Algorithm

Figure 8.2 presents a variant of the Nelson-Oppen combination method for the union of two arbitrary signature-disjoint, stably infinite, and convex theories where explanation engines are used in place of satisfiability procedures. It applies to $\mathcal{E} \cup \mathcal{LA}$ since $\mathcal{E}$ and $\mathcal{LA}$ are known to be stably infinite and convex, and it is possible to build explanation engines for both $\mathcal{E}$ and $\mathcal{LA}$ as shown in Section 8.5.1. The rules of Figure 8.2 (derived from [RRT04]) aim at providing a $T_1 \cup T_2$-satisfiability procedure for sets of literals of the form $\varphi = \Omega_1 \cup \Omega_2 \cup E \cup \Delta_V$, where Ω_i is a solved form of T_i-equalities (for $i = 1, 2$), E is a set of elementary equalities and Δ_V is a set of elementary disequalities.

Configurations manipulated by the rules consist of $\Omega_1, \Omega_2, \Delta_V$ together with an explanation graph G of φ. Initially, G is defined as $UF^{Var(\varphi)}(E)$, the explanation graph obtained via Union-Find with E as input, and whose vertices are $Var(\varphi)$. The combination algorithm works as follows. Each explanation engine computes new entailed equalities stored in a (local) explanation graph. Then, this explanation is used to update the (global) explanation graph G for the union of theories. This update operation (see $Merge$ function defined in Figure 8.2) consists in adding some new edges to G. The unsatisfiability is detected either by an explanation engine (rule $\texttt{Unsat}_{=\texttt{i}}$, for $\texttt{i} = 1, 2$) or by a contradiction between the entailed elementary equalities stored in G and the elementary disequalities in Δ_V (rule $\texttt{Unsat}_{\neq}$). By the results in [RRT04], one can show that the repeated application of rules in Figure 8.2 terminates with *false* if and only if the initial configuration is unsatisfiable.

Theorem 8.3. *Let T_1 and T_2 be two signature-disjoint convex and stably infinite theories such that for each $i = 1, 2$, a T_i-explanation engine is known. Let Ω_i be a set in solved form of T_i-equalities for $i = 1, 2$, E be a set of elementary equalities and Δ_V be a set of elementary disequalities. Consider $\varphi = (\Omega_1 \cup \Omega_2 \cup \Delta_V \cup E)$ and let cf a final configuration obtained by the repeated application of the rules of Figure 8.2 on the initial configuration $\Omega_1; \Delta_V; UF^{Var(\varphi)}(E); \Omega_2$.*

- *If cf is of the form $false\{(\Omega', E', G)\}$, then φ is $T_1 \cup T_2$-unsatisfiable. Furthermore, $\Omega' \cup E'$ is a minimal conflict set such that $E' \subseteq CP(G)$.*
- *Otherwise, cf is of the form $\Omega_1; \Delta_V; G; \Omega_2$ and φ is $T_1 \cup T_2$-satisfiable. Furthermore, G is complete for φ modulo $T_1 \cup T_2$.*

Moreover, G is an **edge-minimal** *explanation graph of φ.*

Theorem 8.3 has two interesting consequences. First, given a T_1-explanation engine, it provides a T_1-satisfiability procedure when T_2 is the empty theory with the empty signature. For instance, it can be used to get an $\mathcal{E}$-satisfiability procedure with

explanation from an $\mathcal{E}$-explanation engine : in case of unsatisfiability, this procedure returns a "minimal" configuration of the form $false\{(\{x \neq y\}, \{x = y\}, G)\}$ since $\mathsf{Unsat}_{=1}$ does not apply when $T_1 = \mathcal{E}$. Another consequence is that the combination algorithm provides a $T_1 \cup T_2$-explanation engine when Δ_V is empty. Thus, we have a modular construction of explanation engines since $T_1 \cup T_2$ is convex and stably infinite when T_1 and T_2 are signature-disjoint convex and stably infinite theories.

$\mathsf{Unsat}_{=1}$

$$\frac{\Omega_1; \Delta_V; G; \Omega_2}{false\{(\Omega_1', E_1', G')\}}$$

$$\textbf{if} \begin{cases} \mu EX_1(\Omega_1, Eq(G)) = (\Omega_1', E_1', G_1) \\ \Omega_1' \neq \emptyset \\ G' = Merge(G, G_1) \end{cases}$$

$\mathsf{Unsat}_{\neq}$

$$\frac{\Omega_1; \Delta_V; G; \Omega_2}{false\{(\{x \neq y\}, \{x = y\}, G)\}}$$

$$\textbf{if} \begin{cases} (x, y) \in CP(G) \\ x \neq y \in \Delta_V \end{cases}$$

$\mathsf{Deduction}_1$

$$\frac{\Omega_1; \Delta_V; G; \Omega_2}{\Omega_1; \Delta_V; G'; \Omega_2}$$

$$\textbf{if} \begin{cases} \mu EX_1(\Omega_1, Eq(G)) = (\emptyset, E_1, G_1) \\ G' = Merge(G, G_1) \\ G' \neq G \end{cases}$$

Symmetric rules can be obtained by changing the subscript 1 into 2 in the rules above. $Merge$ is defined as follows :

Function $Merge(G, G')$
  $G'' := G$
    Foreach $(x, y) \in Edge(G')$
      If $(x, y) \notin CP(G'')$ **Then** $G'' := Insert(G'', x = y, \mathcal{L}_{G'}(x = y))$
    EndForeach
  Return G''

FIGURE 8.2 – Combination of explanation engines

8.6 Quasi-Conflict Sets

In the previous combination algorithm, one can observe that some additional information, encoded as a triplet (ψ, E, G), is returned whenever an unsatisfiable set φ of literals is considered. This triplet contains two sets of literals ψ, E, and an explanation graph G such that $\psi \cup E$ is unsatisfiable, ψ is a satisfiable subset of φ, and E is a set of (entailed) elementary equalities explained in G. Strictly speaking, $\psi \cup E$ is not a conflict set of φ since it may contain literals which are not in φ. However, it is easy to extract a "true" conflict set from $\psi \cup E$ since E is entailed by φ and the related explanations are encoded in the associated explanation graph G.

In the following, we investigate how to formalize this observation by defining the concept of *quasi-conflict set* as a triplet (ψ, E, G). It turns out that it is possible to define an ordering on such triplets and that the quasi-conflict sets computed by the combination algorithm are minimal according to this ordering. Technically, we use the following ordering to compare explanation graphs.

Definition 8.4. *Given two explanation graphs G and G' of φ w.r.t. (the same ordering) $<$, we define the relation $\sqsubseteq$ as follows :*

$$G' \sqsubseteq G \text{ if } Edge(G') \subseteq Edge(G) \text{ and } \forall e \in Edge(G'),\ \mathcal{L}_{G'}(e) \subseteq \mathcal{L}_G(e).$$

Given two explanation graphs G and G' of φ w.r.t. (the same ordering) $<$ and four sets of literals ψ, E, ψ', E', we define the relation $\preceq$ as follows :

$$(\psi', E', G') \preceq (\psi, E, G) \text{ if } \psi' \subseteq \psi,\ E' \subseteq E \text{ and } G' \sqsubseteq G$$

It is easy to see that $\preceq$ is a quasi-ordering. We denote with $\prec$ the strict ordering induced by $\preceq$. In the following, we formally define quasi-conflict sets as triplets (ψ, E, G) satisfying some properties. Then, minimal quasi-conflict sets are defined w.r.t. the ordering $\prec$.

Definition 8.5. *Let φ be an unsatisfiable set of literals, ψ be a satisfiable (strict) subset of φ, G be an explanation graph of φ w.r.t. some $<$, and E be a set of equalities. The triplet (ψ, E, G) is a* quasi-conflict set *of φ w.r.t. $<$ if $E \subseteq CP(G)$ and $\psi \cup E$ is unsatisfiable. The triplet (ψ, E, G) is a* minimal quasi-conflict set *if there is no $(\psi', E', G') \prec (\psi, E, G)$ such that (ψ', E', G') is a quasi-conflict set of φ w.r.t. $<$.*

Proposition 8.4. *If (ψ, E, G) is a quasi-conflict set of φ, then $\psi \cup Lit(G)$ is a conflict set of φ.*

Given a quasi-conflict set (ψ, E, G) of φ, $\psi \cup Lit(G)$ is called the *conflict set associated to* (ψ, E, G). The set of literals $Lit(G)$ provides an explanation of equalities in E, but it is a superset of what we really need. Indeed, it is sufficient to consider the subgraph of G obtained by selecting paths in G which "connect" equalities of E. This subgraph is formally defined below.

Definition 8.6. *Let G be an explanation graph of φ w.r.t. $<$, $x = y \in CP(G)$ and $E \subseteq CP(G)$. The* explanation edges of $x = y$ in G *is the subset of $Edge(G)$ defined as follows :*

$$Exe(G, x = y) = ElemPath(G, x, y) \ \cup \ \Big(\bigcup_{e \in ElemPath(G,x,y)} \ \bigcup_{e' \in \mathcal{L}_G(e) \setminus \varphi} Exe(G, e') \Big).$$

The explanation edges of E *in* G *is*

$$ExE(G,E) = \bigcup_{e \in E} Exe(G,e)\,.$$

The restriction of G to E *is the subgraph* $G_{|E}$ *of* G *such that*

$$Edge(G_{|E}) = ExE(G,E) \text{ and } \forall e \in Edge(G_{|E}),\ \mathcal{L}_{G_{|E}}(e) = \mathcal{L}_G(e)\,.$$

We are now ready to show how to compute minimal quasi-conflict sets.

Theorem 8.4. *Let* (ψ, E, G) *be a quasi-conflict set of* φ *such that* $\psi \cup E$ *is a minimal conflict set. If all edges of* $G_{|E}$ *are minimally explained then* $(\psi, E, G_{|E})$ *is a minimal quasi-conflict set of* φ*.*

Proof. $(\psi, E, G_{|E})$ is a quasi-conflict set since $E \subseteq CP(G_{|E})$ and $\psi \cup E$ is unsatisfiable. Assume there exists a quasi-conflict set (ψ', E', G') of φ such that $(\psi', E', G') \prec (\psi, E, G_{|E})$. Since edges of $G_{|E}$ are minimally explained, we have necessarily $\forall e \in Edge(G'),\ \mathcal{L}_{G'}(e) = \mathcal{L}_{G_{|E}}(e)$, and the following cases :

1. $\psi' \subset \psi$, $E' \subseteq E$, $G' \subseteq G_{|E}$,
2. or $\psi' \subseteq \psi$, $E' \subset E$, $G' \subseteq G_{|E}$,
3. or $\psi' \subseteq \psi$, $E' \subseteq E$, $G' \subset G_{|E}$. In that case, suppose $E' = E$. Then, we have $E \subseteq CP(G')$. But this contradicts the fact that $G' \subset G_{|E}$. Consequently, $E' \subset E$.

In all cases, we get the strict inclusion $\psi' \cup E' \subset \psi \cup E$. Since $\psi \cup E$ is a minimal conflict set, $\psi' \cup E'$ is satisfiable. This contradicts the assumption. □

Theorem 8.4 can be applied to the combination algorithm given in Section 8.5.2. In case of unsatisfiability, the returned triplet (Ω', E', G) leads to a **minimal quasi-conflict** set $(\Omega', E', G_{|E'})$.

Corollary 8.2. *Consider a* T*-explanation engine* μEX*. Assume* $\mu EX(\Omega, E) = (\Omega', E', G)$*. If* $\Omega \cup E$ *is* T*-unsatisfiable then* $\Omega' \cup E'$ *is a minimal conflict set and* $(\Omega', E', G_{|E'})$ *is a minimal quasi-conflict set.*

Moreover, Theorem 8.4 can be applied to the Congruence Closure algorithm (with Explanation). In that case, one can construct a quasi-conflict set having the property to be minimal.

Corollary 8.3. *Let* φ*,* Δ_V *and* G *be defined as in Theorem 8.4. If* φ *is unsatisfiable, then there exist* x, y *such that* $x \neq y \in \Delta_V$*,* $(x,y) \in CP(G)$*, and* $(\{x \neq y\}, \{x = y\}, G_{|\{x=y\}})$ *is a minimal quasi-conflict set.*

Finally, Theorem 8.4 can also be applied to the combination algorithm. In that case also, it provides us with a characterization of the output when unsatisfiability is reported.

Corollary 8.4. *Given the assumptions of Theorem 8.3, the combination algorithm depicted in Figure 8.2 returns in case of unsatisfiability a configuration of the form* $false\{(\Omega', E', G)\}$ *such that* $(\Omega', E', G_{|E'})$ *is a minimal quasi-conflict set.*

8.7 Discussion

In this chapter, we studied the problem of augmenting decision procedures with the capability of computing conflict sets, as well as the problem of modularly constructing conflict sets for a combined theory, when this is obtained as the disjoint union of many component theories for which satisfiability procedures capable of computing conflict sets are assumed available. The key concept is that of explanation graph, which allows us to encode the fact that a certain elementary equality is a logical consequence of a set of elementary equalities. We show how explanation graphs can be used to build satisfiability procedures for the theory of equality and Linear Arithmetic which are capable of explaining unsatisfiability as well as entailed equalities. We call theses procedures explanation engines. We also provide a procedure to build conflict sets in the union of two theories. By refining the Nelson and Oppen method, we obtain a truly modular combination method to build conflict sets in unions of n theories (with $n \geq 2$) by combining explanation engines. The minimality of quasi-conflict sets in the union of theories is also investigated.

In the near future, we plan to implement the proposed algorithms and evaluate their viability in haRVey [DR03]. In particular, we envisage to implement *explanation graphs* by "off-the-shelf" optimized graph algorithms [EGI97], which would allow us to make the procedure fully incremental. Another interesting line of future work is to study how to combine explanation graphs with the directed acyclic graphs encoding the refutation of a SAT solver introduced in [Gel02]. This would allow us to go a step further in the direction of building truly certifiable SMT tools.

Chapitre 9

Conclusion et Perspectives

Nous avons montré différents résultats dans le domaine de la conception de procédures de satisfiabilité et de leurs intégrations dans des solveurs SMT. Nous avons proposé quelques techniques de combinaison dans l'optique d'avoir une intégration à la fois efficace, flexible et sûre de procédures de satisfiabilité. En résumé, les résultats obtenus dans cette thèse concernent les points suivants.

Nelson-Oppen, Shostak vs. Canoniseur étendu

Nous avons apporté une contribution au débat qui consiste à se positionner par rapport aux deux grand schémas de combinaison disjointe (i.e. les théories combinées ne partagent pas de symboles) pour la satisfiabilité : Nelson-Oppen et Shostak. Nous avons proposé une troisième voie héritant des avantages respectifs des deux schémas ci-dessus. En effet, nous avons travaillé sur l'utilisation d'outils de canonisation de termes modulo une théorie et une conjonction d'égalités, qu'on appelle canoniseurs étendus. Ce nouveau concept présente plusieurs avantages : (i) la construction de canoniseurs étendus est modulaire ; (ii) un canoniseur étendu décide la satisfiabilité pour une large classe de théories, à savoir les théories convexes ; (iii) les canoniseurs étendus peuvent être construits en utilisant des techniques de réécriture dans certains cas.

Le résultat ci-dessus suppose que les théories composantes satisfont l'hypothèse de convexité et de stable infinité. Nous avons étendu ces résultats aux cas où les théories sont convexes mais pas nécessairement stablement infinies. Nous avons montré que sous l'hypothèse de convexité et la condition de savoir si les théories ont un modèle trivial (cf. Chapitre 7), nous pouvons nous passer de l'hypothèse de stable infinité tout en assurant la correction des schémas de combinaison.

Combinaison des procédures de satisfiabilité dérivées par saturation

Le travail ci-dessus nous a conduit par la suite à étudier la combinaison des procédures de satisfiabilité construites par des techniques de réécriture. En effet, nous avons étendu les travaux existants afin d'obtenir une méthodologie pour construire, en utilisant le Calcul de Superposition, des procédures de satisfiabilité qui sont capables de produire des formules partagées entre les théories combinées pour assurer la complétude et l'efficacité de la combinaison des procédures de satisfiabilité. Nous avons montré que les procédures de satisfiabilité basées sur le Calcul de Superposition

satisfont cette propriété pour des théories intéressantes en vérification, notamment la théorie de l'égalité et la théorie de des listes. Pour la théorie des tableaux nous avons montré qu'une variante du Calcul de Superposition, à savoir le Calcul de Superposition avec une règle d'analyse explicite de cas, conjointement avec un post-traitement, nous permettent d'avoir un résultat similaire. Nous avons utilisé la version d'analyse de cas sans retour arrière (splitting without backtracking) au lieu de celle avec retour arrière. Nous avons ensuite montré que le Calcul de Superposition peut aussi être utilisé avec la Clôture de Congruence pour construire des procédures de satisfiabilité pour la théorie des listes et la théorie des tableaux au lieu d'utiliser seulement le Calcul de Superposition. Notre travail a été motivé par le souhait d'avoir à la fois l'expressivité offerte par la méthodologie par saturation et l'efficacité et la flexibilité de la Clôture de Congruence. De plus, avec une telle architecture, nous pouvons intégrer de façon encore plus efficace des procédures de satisfiabilité dans la méthode de combinaison de Nelson-Oppen car celles-ci peuvent traiter efficacement de nouvelles formules partagées.

Une méthode de preuve automatique par méta-saturation

Jusque là, nous avons étudié de façon "ad hoc" mais très similaire la propriété de saturation finie et la complétude vis-à-vis de la déduction pour chaque théorie considérée. Ce travail nous a conduit à déterminer une classe de théories pour lesquelles ces résultats s'appliquent et par la suite à développer une méthode automatique et uniforme pour la preuve de la propriété de saturation finie, la stable infinité et de la complétude vis-à-vis de la déduction des procédures de satisfiabilité des théories de cette classe. Dans cette optique, nous avons défini des conditions permettant de vérifier si les théories admettent une procédure de satisfiabilité basée sur le calcul de Superposition avec la capacité de produire des formules partagées et si ces procédures peuvent être combinées avec d'autre procédures de satisfiabilité. De plus, nous avons développé une méthode basée sur la méta-saturation de Lynch et Morawska pour vérifier automatiquement ces conditions. Nous avons pu identifier, en utilisant notre méthode automatique, une classe de théories satisfaisant ces conditions, qui inclut bien évidemment la théorie de l'égalité, la théorie des listes, la théorie des tableaux et leur combinaisons.

Procédures de satisfiabilité avec justification

Nous avons également travaillé sur le problème de l'intégration efficace de procédures de satisfiabilité dans le système haRVey où il est fait usage d'un solveur propositionnel pour décider la satisfiabilité d'une formule arbitrairement complexe. Nous avons adopté la direction consistant à équiper les procédures de satisfiabilité de la capacité à justifier l'insatisfiabilité au moyen d'un petit ensemble conflit (conflict set), ce qui permet de réduire l'espace de solutions propositionnelles. Nous avons proposé une procédure de satisfiabilité avec justification de l'insatisfiabilité minimale (dans le sens où tout sous-ensemble strict de la justification est satisfiable) pour l'arithmétique linéaire (partie équationnelle), qui est basée sur l'élimination de Gauss. Dans le cas de la théorie de l'égalité ou l'union de théories, la minimalité des justifications produites par cette méthode ne peut être qu'approchée. Nous avons également étendu la méthode de combinaison de Nelson-Oppen afin de construire de façon mo-

dulaire les procédures de satisfiabilité ayant la capacité de justifier les preuves que nous appelons les *moteurs d'explication* (*explanation engines*). Le concept principal de notre approche est celui de *graphe d'explication* (*explanation graph*) qui intuitivement encode de façon minimale des preuves d'équivalence entre constantes. Nous avons illustré par des exemples de théories intéressantes telles que la théorie de l'égalité, l'arithmétique linéaire équationnelle, les systèmes de contraintes de différence pour lesquelles des moteurs d'explication peuvent être obtenues sans surcoût en étendant des procédures de satisfiabilité existantes avec les graphes d'explication.

Finalement nous avons étudié une caractérisation de la minimalité des ensembles conflits calculés par des moteurs d'explication. Nous avons introduit le concept de *quasi-conflit*, qui nous permet de caractériser précisément une forme de minimalité satisfaite par les explications extraites des graphes d'explication. Nous avons montré que les moteurs d'explication pour la théorie de l'égalité, l'arithmétique linéaire équationnelle, les systèmes de contraintes de différence ainsi que la combinaison de ces moteurs d'explication produisent des quasi-conflits qui sont minimaux par rapport à un ordre clairement défini sur les quasi-conflits.

Perspectives

Les travaux menés durant la thèse ouvrent plusieurs directions de recherche intéressantes.

Méthodologie par méta-saturation

Nous comptons étudier des possibilités d'extension de notre méthodologie par méta-saturation pour la combinaison de théories non stablement infinies. A l'heure actuelle, nous avons seulement étudié la combinaison des procédures de satisfiabilité dérivées par saturation en adoptant la méthode de Nelson-Oppen. Cette étude nous amené à développer une méthode automatique pour décider des conditions suffisantes pour lesquelles la stable infinité est assurée. Nous pensons pouvoir étendre la méthode par méta-saturation à d'autres méthodes de combinaison, comme par exemple des méthodes de type asymétrique. En effet, dans un contexte multi-sorté, [RRZ05, FRZ05, FG04, Fon04] ont montré que la théorie des listes et la théorie des tableaux possèdent la propriété de "politeness" par rapport à la sorte des éléments et ceci permet de combiner la théorie des listes et la théorie des tableaux avec n'importe quelle théorie décidable qui contient la théorie des éléments. L'objectif est de trouver des conditions syntaxiques qui garantissent la "politeness" des théories axiomatisées par un ensemble fini de clauses et d'espérer pouvoir les vérifier par méta-saturation.

Combinaison

Un premier problème à considérer est de prouver la Conjecture 7.1 dans le Chapitre 7 ou de trouver un contre-exemple de celle-ci. Ceci est intéressant car si nous arrivons à prouver la conjecture, nous aurons un résultat de combinaison de théories universelles sans avoir à supposer l'hypothèse de satisfiabilité dans des modèles infinis et arbitrairement finis sur les théories composantes. Ce résultat compléterait ceux de [BGN$^+$06] et [KRRT06a] sur la combinaison des théories axiomatisées par un ensemble fini de clauses, qui ont la propriété de saturation finie.

A plus long terme, il sera intéressant d'étudier la combinaison du problème de satisfiabilité dans le cadre de la vérification des protocoles cryptographiques. Plus précisément, on pourra étudier des mélanges de comportements d'intrus cherchant à obtenir des informations secrètes à partir des informations dont ils ont connaissance. Un comportement d'intrus peut être modélisé par un ensemble de règles de déduction et une théorie équationnelle sur une certaine signature et un ensemble de règles de déduction. Ainsi raisonner sur un comportement d'intrus revient à raisonner sur un problème de décision dans un système de déduction modulo une congruence. Et l'étude de la modularité d'un problème de décision dans un système de déduction modulo une congruence nous permettra de connaître certaines propriétés d'un mélanges de comportements d'intrus. On pourra par exemple partir des travaux décrits dans [CR05, CR06, LLT05].

Minimisation d'ensembles conflits par réécriture

Nous avons proposé des techniques permettant d'étendre des procédures de satisfiabilité connues pour la théorie de l'égalité et l'arithmétique linéaire dans le but de rendre celles-ci capables de produire des preuves (nous les appelons des moteurs d'explication). Nous avons également donné une procédure qui permet de combiner ces moteurs d'explication. Les ensembles conflits produits par ces moteurs d'explication sont utilisés pour élaguer l'espace de solutions propositionnelles. Cette approche peut être qualifiée comme *interne* car elle demande de modifier de façon rigoureuse des procédures de satisfiabilité existantes afin de pouvoir produire des ensembles conflits "minimaux". La nouvelle direction adopte une approche plutôt *externe*. Elle consiste à considérer le problème de minimisation d'ensembles conflits comme la normalisation des termes dans une algèbre de preuves.

Comme observé dans [ST05], les preuves d'égalités peuvent être vues comme des éléments d'une théorie de groupe dont l'élément neutre est la réflexivité, l'inverse est la symétrie et la multiplication est la transitivité. En utilisant la complétion des axiomes de groupe, on peut normaliser les preuves d'égalités et les preuves en forme normale nécessitent le moins d'hypothèses possibles. Cette technique nous permet d'obtenir des ensembles conflits minimaux dans le cas de la théorie de la relation d'équivalence. On peut imaginer que cette démarche s'applique également à l'arithmétique linéaire. En effet, nous avons vu dans le Chapitre 8 que les étiquettes utilisées pour enregistrer les opérations de lignes pendant l'élimination de Gauss peuvent être vues comme des preuves élémentaires avec lesquelles nous pouvons engendrer une algèbre de preuves avec l'addition de deux preuves et la multiplication d'une preuve par une constante (en fait, nous avons une algèbre linéaire de preuves). Les preuves définies dans cette algèbre peuvent ensuite être mises en forme normale de façon similaire à la simplification de polynômes linéaires à plusieurs variables. On retrouvera le même résultat sur la minimalité de preuves que celui énoncé dans le Chapitre 8. Cette technique peut se généraliser ensuite à l'arithmétique non linéaire où on aura à considérer une algèbre de preuves qui correspond à un corps de polynômes à plusieurs variables. Simplifier une preuves dans cette algèbre revient à appliquer l'algorithme de Büchberger pour trouver une base de Gröbner minimale.

Par ailleurs, nous pouvons considérer aussi le problème de combinaison de normalisation de preuves, typiquement, en considérant le mélange de l'arithmétique et de fonctions unaires non interprétées. Une procédure de combinaison de normalisa-

tion de preuve peut être envisagée en considérant les preuves de congruence. Dans le cas des fonctions unaires non interprétées, une preuve de congruence se comporte comme un homomorphisme : $Congr(Trans(p_1, p_2)) = Trans(Congr(p_1), Congr(p_2))$, où $Congr$ et $Trans$ désignent respectivement une preuve par congruence et une preuve par transitivité. Nous devons alors considérer un mélange d'algèbres de preuves modulo associativité, commutativité et homomorphisme et dans ce cas une piste intéressante consiste à utiliser les résultats sur la réécriture modulo un ensemble d'équations [Mar96, LLT05].

Implantations

Sur le plan expérimental, il est important de mettre en oeuvre les différentes idées proposées dans cette thèse.

Une implantation de la saturation avec fragmentation nous permettrait de réaliser une étude comparative entre l'approche de combinaison par propagation explicite de disjonctions d'égalités (e.g. des variantes de la méthode de Nelson-Oppen) avec celle utilisant un SAT solveur pour traiter des disjonctions d'égalités ([BBC$^+$06]). Cette étude pourra effectivement mettre en évidence l'intérêt d'avoir des procédures de satisfiabilité complètes vis-à-vis de la déduction.

Pour rendre efficace les procédures de satisfiabilité dérivées par saturation et leur combinaison, on peut imaginer d'implanter l'architecture stratifiée qui consiste à utiliser la saturation comme un "front-end" de la clôture de congruence pour traiter les superpositions entre des littéraux clos et les axiomes des théories. La combinaison se fait ensuite au niveau de la saturation. Ce type d'implantation ne demandera pas beaucoup de travail car plusieurs implantations efficaces de la clôture de congruence [DNS03, NO03, Fon04] et de la saturation [Sch02, Wei97, RV02] peuvent être réutilisées. Il suffit de les coupler en définissant une interface de coopération.

Il est intéressant d'expérimenter la méthodologie par méta-saturation que nous avons développée dans le Chapitre 5. Cela revient à implanter une version de Calcul de Superposition avec contrainte de constante. Nous pensons qu'il est possible d'adapter une implantation du Calcul de Superposition avec des sortes [Wei97] car des contraintes de constantes peuvent être exprimées par des contraintes de sortes. En effet pour indiquer qu'un symbole est une constante il suffit d'imposer que ce symbole soit de sorte "constante" prédéfinie.

Finalement, nous souhaitons expérimenter les idées décrites dans le Chapitre 8, concernant des moteurs d'explication et leur combinaison, dans haRVey pour étudier en pratique leur intérêt. Nous envisageons d'implanter également des graphes d'explication en utilisant "sur étagère" les algorithmes de graphes dynamiques [EGI97]. Ceci permet d'avoir une procédure de combinaison qui est incrémentale. Une étude comparative d'une telle implantation et de [BBC$^+$06] est aussi envisageable.

Bibliographie

[ABRS05] A. Armando, M. P. Bonacina, S. Ranise, and S. Schulz. On a rewriting approach to satisfiability procedures : extension, combination of theories and an experimental appraisal. In B. Gramlich, editor, *Proc. of the 5th Int. Workshop on Frontiers of Combining Systems (FroCoS'2005)*, volume 3717 of *Lecture Notes in Computer Science*, pages 65–80, Vienna, Austria, September 2005. Springer.

[ABRS06] A. Armando, M. P. Bonacina, S. Ranise, and S. Schulz. New results on rewrite-based satisfiability procedures, 2006. Available at `http://arxiv.org/abs/cs.AI/0604054`.

[Ack54] W. Ackerman. Solvable Cases of the Decision Problem. *Studies in Logic and the Foundations of Mathematics*, 1954.

[ARR03] A. Armando, S. Ranise, and M. Rusinowitch. A Rewriting Approach to Satisfiability Procedures. *Journal of Information and Computation*, 183(2) :140–164, June 2003.

[BB04] Clark W. Barrett and Sergey Berezin. CVC lite : A new implementation of the cooperating validity checker. In *Proc of. Computer Aided Verification, 16th International Conference, (CAV'04), Boston, MA, USA, July 13-17*, Lecture Notes in Computer Science, pages 515–518. Springer, 2004.

[BBC+96] N. Bjorner, A. Browne, E. Chang, M. Colon, A. Kapur, Z. Manna, H. Sipma, and T. Uribe. Step : Deductive-algorithmic verification of reactive and real-time systems. In R. Alur and T.A. Henzinger, editors, *Proc. of Computer Aided Verification, 8th International Conference, (CAV'96), New Brunswick, NJ, USA, July 31 - August 3*, volume 1102, pages 415–418, New Brunswick, NJ, July/August 1996. Springer.

[BBC+06] M. Bozzano, R. Bruttomesso, A. Cimatti, T.A. Junttila, S. Ranise, P. van Rossum, and R. Sebastiani. Efficient Theory Combination via Boolean Search. *Journal of Information and Computation*, 10(204) :1411–1596, 2006. Special Issue on Combining Logical Systems.

[BCLZ04] T. Ball, B. Cook, S. K. Lahiri, and L. Zhang. Zapato : Automatic theorem proving for predicate abstraction refinement. In *Proc. of Computer Aided Verification, 16th International Conference, (CAV'04), Boston, MA, USA*, Lecture Notes in Computer Science, pages 457–461. Springer, 2004.

[BDL96] C. W. Barrett, D. L. Dill, and J. R. Levitt. Validity Checking for Combinations of Theories with Equality. In M. Srivas and A. Camilleri, editors, *Proc. of Formal Methods In Computer-Aided Design (FMCAD'96,*

Palo Alto, California, USA), volume 1166 of *Lecture Notes in Computer Science*, pages 187–201. Springer, November 1996.

[BDS02] C. W. Barrett, D. L. Dill, and A. Stump. A generalization of Shostak's method for combining decision procedures. In A. Armando, editor, *Proc. of the 4th International Workshop on Frontiers of Combining Systems, FroCoS'02 (Santa Margherita Ligure, Italy)*, volume 2309 of *Lecture Notes in Computer Science*, pages 132–147. Springer, apr 2002.

[BG96] D. Basin and H. Ganzinger. Complexity analysis based on ordered resolution. In *Proc. of 11th IEEE Symposium on Logic in Computer Science, (LICS'96)*, pages 456–465. IEEE Computer Society Press, 1996.

[BGN+06] M. P. Bonacina, S. Ghilardi, E. Nicolini, S. Ranise, and D. Zucchelli. Decidability and undecidability results for nelson-oppen and rewrite-based decision procedures. In *Proc. of the 3rd Int. Conference on Automated Reasoning (IJCAR'06), Seattle, WA, USA*, number 4130 in Lecture Notes in Artificial Intelligence, pages 513–527. Springer, August 2006.

[BGT04] F. Baader, S. Ghilardi, and C. Tinelli. A new combination procedure for the word problem that generalizes fusion decidability results in modal logics. In *Proc. of Automated Reasoning - Second International Joint Conference, (IJCAR'04), Cork, Ireland, July 4-8*, volume 3097 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 2004.

[Bjø98] N. S. Bjørner. *Integrating Decision Procedures for Temporal Verification.* PhD thesis, Computer Science Department, Stanford University, 1998.

[BJSS88] A. Boudet, J.-P. Jouannaud, and M. Schmidt-Schauß. Unification in boolean rings and abelian groups. In *Proc. of Third Annual Symposium on Logic in Computer Science, (LICS'88), 5-8 July 1988, Edinburgh, Scotland, UK*, pages 121–130. IEEE Computer Society, 1988.

[BLH96] J. Burg, S.-D. Lang, and C. E. Hughes. Intelligent Backtracking in CLP(R). *Ann. Math. Artif. Intell.*, 17(3-4) :189–211, 1996.

[BN98] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, United Kingdom, 1998.

[Bou93] A. Boudet. Combining unification algorithms. *Journal of Symbolic Computation*, 16(6) :597–626, 1993.

[BS95a] F. Baader and K.U. Schulz. Combination of constraint solving techniques : An algebraic point of view. In *Proceedings of the 6th International Conference on Rewriting Techniques and Applications, (RTA'95)*, volume 914 of *Lecture Notes in Artificial Intelligence*, pages 352–366, Kaiserslautern, Germany, 1995. Springer.

[BS95b] F. Baader and K.U. Schulz. On the combination of symbolic constraints, solution domains, and constraint solvers. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming, (CP'95)*, volume 976 of *Lecture Notes in Artificial Intelligence*, pages 380–397. Springer, 1995.

[BS96] F. Baader and K. U. Schulz. Unification in the union of disjoint equational theories : Combining decision procedures. *Journal of Symbolic Computation*, 21 :211–243, 1996.

[BS98] F. Baader and K. Schulz. Combination of constraint solvers for free and quasi-free structures. *Theoretical Computer Science*, 192 :107–161, 1998.

[BT97] F. Baader and C. Tinelli. A new approach for combining decision procedures for the word problem, and its connection to the Nelson-Oppen combination method. In W. McCune, editor, *Proc. of the 14th International Conference on Automated Deduction, (CADE'97),Townsville, Australia*, volume 1249 of *Lecture Notes in Artificial Intelligence*, pages 19–33. Springer-Verlag, 1997.

[BT98] F. Baader and C. Tinelli. Deciding the word problem in the union of equational theories. Technical Report UIUCDCS-R-98-2073, Department of Computer Science, University of Illinois at Urbana-Champaign, October 1998.

[BTV03] L. Bachmair, A. Tiwari, and L. Vigneron. Abstract Congruence Closure. *Journal of Automated Reasoning*, 31(2) :129–168, 2003.

[CK03a] S. Conchon and S. Krstić. Strategies for combining decision procedures. In *Proc. of the 9th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems, (TACAS'03)*, volume 2619 of *Lecture Notes in Computer Science*, pages 537–553. Springer, April 2003.

[CK03b] S. Conchon and Sava Krstić. Canonization for disjoint unions of theories. In F. Baader, editor, *Proc. of the 19th International Conference on Automated Deduction (CADE'03)*, volume 2741 of *Lecture Notes in Computer Science*, Miami Beach, FL, USA, July 2003. Springer.

[CLS96] D. Cyrluk, P. Lincoln, and N. Shankar. On Shostak's decision procedure for combinations of theories. In M. A. McRobbie and J.K. Slaney, editors, *Proc. of the 13th International Conference on Automated Deduction, (CADE'96), New Brunswick, NJ*, volume 1104 of *Lecture Notes in Artificial Intelligence*, pages 463–477. Springer, 1996.

[Coo72] D.C. Cooper. Theorem Proving in Arithmetic without Multiplication. *Machine Intelligence*, 7 :91–99, 1972.

[CR05] Y. Chevalier and M. Rusinowitch. Combining intruder theories. In *Proceedings ofAutomata, Languages and Programming, 32nd International Colloquium, (ICALP'05), Lisbon, Portugal, July 11-15*, volume 3580 of *Lecture Notes in Computer Science*, pages 639–651. Springer, 2005.

[CR06] Y. Chevalier and M. Rusinowitch. Hierarchical combination of intruder theories. In Frank Pfenning, editor, *Proc. of Rewriting Technics and Applications, 17th International Conference, (RTA'06), Seattle, WA, USA, August 12-14*, volume 4098 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2006.

[DKR94] E. Domenjoud, F. Klay, and C. Ringeissen. Combination techniques for non-disjoint equational theories. In A. Bundy, editor, *Proc. of 12th International Conference on Automated Deduction, (CADE'94), Nancy, France*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 267–281. Springer, 1994.

[DLNS98] David L. Detlefs, K. Rustan M. Leino, G. Nelson, and James B. Saxe. Extended static checking. Research Report #159, Compaq Systems Research Center, Palo Alto, USA, December 1998.

[dMRS04] L. de Moura, H. Rueß, and N. Shankar. Justifying equality. In C. Tinelli and S. Ranise, editors, *Proc. of Second Workshop on Pragmatics of Decision Procedures in Automated Reasoning, (PDPAR'04), Workshop affiliated to IJCAR'04, July 05, University College Cork, Cork, County Cork, Ireland*, 2004. Also in Electronic Notes in Theoretical Computer Science (ENTCS), Volume 125, Issue 3.

[DNS03] D. Detlefs, G. Nelson, and J. B. Saxe. Simplify : A Theorem Prover for Program Checking. Technical Report HPL-2003-148, HP Laboratories, 2003.

[DR03] D. Déharbe and S. Ranise. Light-Weight Theorem Proving for Debugging and Verifying Units of Code. In IEEE Comp. Soc. Press, editor, *Proc. of the Int. Conf. on Software Engineering and Formal Methods (SEFM'03)*, 2003.

[DST80] P. J. Downey, R. Sethi, and R. E. Tarjan. Variations on the common subexpression problem. *J. ACM*, 27(4) :758–771, 1980.

[EGI97] D. Eppstein, Z. Galil, and G. Italiano. Dynamic graph algorithms. In CRC Handbook of Algorithms and Theory of Computation, Chapter 22. CRC Press, 1997.

[End72] H. B. Enderton. *A Mathematical Introduction to Logic*. Ac. Press, Inc., 1972.

[FG03] C. Fiorentini and S. Ghilardi. Combining word problems through rewriting in categories with products. *Theoretical Computer Science*, 294(1/2) :103–149, 2003.

[FG04] P. Fontaine and E. P. Gribomont. Combining non-stably infinite, non-first order theories. In C. Tinelli and S. Ranise, editors, *Proc. of Second Workshop on Pragmatics of Decision Procedures in Automated Reasoning, (PDPAR'04), Workshop affiliated to IJCAR 2004, July 05, University College Cork, Cork, County Cork, Ireland*, July 2004.

[Fon04] P. Fontaine. *Techniques for Verification of Concurrent Systems with Invariants*. PhD thesis, Université de Liège, 2004.

[FORS01] J.-C. Filliâtre, S. Owre, H. Rueß, and N. Shankar. ICS : Integrated Canonization and Solving (Tool presentation). In G. Berry, H. Comon, and A. Finkel, editors, *Proc. of Computer Aided Verification, 13th International Conference, (CAV'01), Paris, France, July 18-22*, volume 2102 of *Lecture Notes in Computer Science*, pages 246–249. Springer, 2001.

[FR74] M. J. Fischer and M. O. Rabin. Super-exponential complexity of presburger arithmetic. In *Proc. of Symposium in Applied Mathematics of the American Mathematical Society and the Society for Industrial and Applied Mathematics*, volume 7, pages 27–42. American Mathematical Society, 1974.

[Fre99] Ralph Freese. Computing congruences efficiently. preprint, 1999.

[FRZ05] P. Fontaine, S. Ranise, and C. G. Zarba. Combining lists with non-stably infinite theories. In Franz Baader and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'04)*, volume 3452 of *Lecture Notes in Computer Science*, pages 51–66. Springer, 2005.

[Gan01] H. Ganzinger. Relating semantic and proof-theoretic concepts for polynomial time decidability of uniform word problems. In *Proc. of 16th IEEE Symposium on Logic in Computer Science, (LICS'01)*, pages 81–92. IEEE Computer Society Press, 2001.

[Gan02] H. Ganzinger. Shostak light. In A. Voronkov, editor, *Proc. of the 18th International Conference on Automated Deduction, (CADE'02)*, volume 2392 of *Lecture Notes in Computer Science*, pages 332–346. Springer, jul 2002.

[Gel02] A. Van Gelder. Extracting (Easily) Checkable Proofs from a Satisfiability Solver that Employs both Preorder and Postorder Resolution. In *Proc. of 7th Int. Symp. on Artificial Intelligence and Mathematics (AMAI'02), Florida, USA*, 2002.

[Ghi04] S. Ghilardi. Model-theoretic methods in combined constraint satisfiability. *Journal of Automated Reasoning*, 33(3-4) :221–249, 2004.

[GHN+04] H. Ganzinger, G. Hagen, R. Nieuwenhuis, A. Oliveras, and C. Tinelli. DPLL(T) : Fast decision procedures. In R. Alur and D. Peled, editors, *Proceedings of the 16th International Conference on Computer Aided Verification, (CAV'04), Boston, Massachusetts*, volume 3114 of *Lecture Notes in Computer Science*, pages 175–188. Springer, 2004.

[GHW03] H. Ganzinger, T. Hillenbrand, and U. Waldmann. Superposition modulo a Shostak theory. In F. Baader, editor, *Proc. of Conference on Automated Deduction, (CADE'03), Miami, Florida, USA*, volume 2741 of *Lecture Notes in Computer Science*, pages 182–196. Springer, 2003.

[GK03] H. Ganzinger and K. Korovin. New directions in instantiation-based theorem proving. In *Proc. 18th IEEE Symposium on Logic in Computer Science, (LICS'03)*, pages 55–64. IEEE Computer Society Press, 2003.

[GK04] H. Ganzinger and K. Korovin. Integrating equational reasoning into instantiation-based theorem proving. In *Proc. of Computer Science Logic, (CSL'04)*, volume 3210 of *Lecture Notes in Computer Science*, pages 71–84. Springer, 2004.

[GNZ05] S. Ghilardi, E. Nicolini, and D. Zucchelli. A comprehensive framework for combined decision procedures. In *Proc. of Frontiers of Combining Systems, 5th International Workshop, (FroCoS'05), Vienna, Austria, September 19-21*, volume 3717 of *Lecture Notes in Computer Science*, pages 1–30. Springer, 2005.

[Her86] A. Herold. Combination of unification algorithms. In *8th International Conference on Automated Deduction*, pages 450–469, 1986.

[HJMM04] T. A. Henzinger, R. Jhala, R. Majumdar, and K. L. McMillan. Abstractions from proofs. In *Proc. of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, (POPL'04), Venice, Italy, January 14-16*, pages 232–244. ACM, 2004.

[Hod94] W. Hodges. *Model theory.* Cambridge University Press, Great Britain, second edition, 1994.

[Kap97] D. Kapur. Shostak's congruence closure as completion. In *Proc. of Rewriting Techniques and Applications, 8th International Conference,*

(RTA'97), Sitges, Spain, June 2-5, volume 1232 of *Lecture Notes in Computer Science*. Springer, 1997.

[Kap02] D. Kapur. A rewrite rule based framework for combining decision procedures. In *Proc. of the 4th Int. Workshop on Frontiers of Combining Systems, (FroCos'02)*, volume 2309 of *Lecture Notes in Computer Science*, pages 87–102. Springer, 2002.

[Kir89] C. Kirchner. From unification in combination of equational theories to a new AC-unification algorithm. In *Resolutions of Equations in Algebraic Structures : Rewriting Tehcniques*, volume 2, pages 171–210. Academic Press, New York, 1989.

[KK90] M. Kurihara and I. Kaji. Modular term rewriting systems and the termination. *Information Processing Letters*, 34(1) :1–4, 1990.

[KN94] D. Kapur and X. Nie. Reasoning about Numbers in Tecton. In *Proc.* 8^{th} *Inl. Symp. Methodologies for Intelligent Systems*, pages 57–70, 1994.

[KR92] H. Kirchner and C. Ringeissen. A constraint solver in finite algebras and its combination with unification algorithms. In K. Apt, editor, *Proc. of the Joint International Conference and Symposium on Logic Programming, Washington (USA)*, pages 225–239. The MIT Press, nov 1992.

[KR94a] H. Kirchner and C. Ringeissen. Combining symbolic constraint solvers on algebraic domains. *Journal of Symbolic Computation*, 18(2) :113–155, Aug 1994.

[KR94b] H. Kirchner and C. Ringeissen. Constraint solving by narrowing in combined algebraic domains. In P. Van Hentenryck, editor, *Proceedings International Conference on Logic Programming, (ICLP'94), Santa Margherita (Italy)*, pages 617–631. MIT Press, jun 1994.

[KRRT05] H. Kirchner, S. Ranise, C. Ringeissen, and D.-K. Tran. On superposition-based satisfiability procedures and their combination. In *Proc. of Second International Colloquium on Theoretical Aspects of Computing, (ICTAC'05), Hanoi, Vietnam*, volume 3722 of *Lecture Notes in Computer Science*, pages 594–608. Springer, Oct 2005.

[KRRT06a] H. Kirchner, S. Ranise, C. Ringeissen, and D.-K. Tran. Automatic Combinability of Rewriting-Based Satisfiability Procedures. In *Proc. of the 13th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning, (LPAR'06)*, volume 4246 of *Lecture Notes in Artificial Intelligence (subseries of Lecture Notes in Computer Science)*, pages 542–556, Phnom Penh, Cambodia, November 2006. Springer.

[KRRT06b] H. Kirchner, S. Ranise, C. Ringeissen, and D.-K. Tran. Building and Combining Satisfiability Procedures for Software Verification. In *Proc. of 3rd Taiwanese-French Conference on Information Technology, (TFIT'06)*, pages 125–139, Nancy, France, March 2006.

[LLT05] Pascal Lafourcade, Denis Lugiez, and Ralf Treinen. Intruder deduction for AC-like equational theories with homomorphisms. In J. Giesl, editor, *Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA'05)*, volume 3467 of *Lecture Notes in Computer Science*, pages 308–322, Nara, Japan, April 2005. Springer.

[LM02] C. Lynch and B. Morawska. Automatic decidability. In *Proc. of 17th IEEE Symposium on Logic in Computer Science, (LICS'02), Copenhagen, Denmark, July 22-25*, pages 7–. IEEE Computer Society Press, 2002.

[Mar96] C. Marché. Normalized rewriting : An alternative to rewriting modulo a set of equations. *Journal of Symbolic Computation*, 21(3) :253–288, 1996.

[McM03] K. L. McMillan. Interpolation and sat-based model checking. In *Proc. of Computer Aided Verification, 15th International Conference, (CAV'03), Boulder, CO, USA, July 8-12*, volume 2725 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2003.

[McM04] K. L. McMillan. An interpolating theorem prover. In *Proc. of Tools and Algorithms for the Construction and Analysis of Systems, 10th International Conference, (TACAS'04), Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29 - April 2*, volume 2988 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 2004.

[McM05] K. L. McMillan. Applications of craig interpolants in model checking. In *Proc. of Tools and Algorithms for the Construction and Analysis of Systems, 11th International Conference, (TACAS'05), Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8*, volume 3440 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005.

[Mid89] A. Middeldorp. A sufficient condition for the termination of the direct sum of term rewriting systems. In *Proc. of Fourth Annual Symposium on Logic in Computer Science, 5-8 June, (LICS'89), Asilomar Conference Center, Pacific Grove, California, USA*, pages 396–401. IEEE Computer Society, 1989.

[MT91] A. Middeldorp and Yoshihito Toyama. Completeness of combinations of constructor systems. In *Proc. of Rewriting Techniques and Applications, 4th International Conference, (RTA'91), Como, Italy, April 10-12*, volume 488 of *Lecture Notes in Computer Science*, pages 188–199. Springer, 1991.

[MT93] A. Middeldorp and Yoshihito Toyama. Completeness of combinations of constructor systems. *Journal of Symbolic Computation*, 15(3) :331–348, 1993.

[MZ03] Z. Manna and C. G. Zarba. Combination. In *Formal Methods at the Cross Roads : From Panacea to Foundational Support*, Lecture Notes in Computer Science. Springer, 2003.

[Nip91] T. Nipkow. Combining matching algorithms : The regular case. *Journal of Symbolic Computation*, 12(6) :633–654, 1991.

[NO79] G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *ACM Transations on Programming Languages and Systems*, 1(2) :245–257, October 1979.

[NO80] G. Nelson and D. C. Oppen. Fast decision procedures based on congruence closure. *J. ACM*, 27(2) :356–364, 1980.

[NO03] R. Nieuwenhuis and A. Oliveras. Congruence closure with integer offsets. In M. Vardi and A. Voronkov, editors, *Proc. of 10h Int. Conf. Logic for Programming, Artif. Intell. and Reasoning, (LPAR'03), Almaty, Kazakhstan, September 22-26th*, volume 2850 of *Lecture Notes in Artificial Intelligence*, pages 78–90, 2003.

[NO05] R. Nieuwenhuis and A. Oliveras. Proof-Producing Congruence Closure. In *Proc. of the 16th International Conference on Rewriting Techniques and Applications, (RTA'05), Nara, Japan*, volume 3467 of *Lecture Notes in Computer Science*, pages 453–468. Springer, 2005.

[NR01] R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In A. Robinson and A. Voronkov, editors, *Hand. of Automated Reasoning*. Elsevier and MIT Press, 2001.

[Opp80] D. C. Oppen. Complexity, convexity and combinations of theories. *Theoretical Computer Science*, 12 :291–302, 1980.

[ORR+96] S. Owre, S. Rajan, J.M. Rushby, N. Shankar, and M.K. Srivas. PVS : combining specification, proof checking, and model checking. In R. Alur and T.A. Henzinger, editors, *Proc. of Computer Aided Verification, 8th International Conference, (CAV'96)*, number 1102 in Lecture Notes in Computer Science, pages 411–414, New Brunswick, NJ, July/August 1996. Springer.

[Pap81] Christos H. Papadimitriou. On the complexity of integer programming. *Journal of ACM*, 28(4) :765–768, 1981.

[Pig74] D. Pigozzi. The joint of equational theories. In *Colloquium Mathematicum*, pages 15–25, 1974.

[Pra77] V.R. Pratt. Two easy theories whose combinantion is hard. Research report, MIT, 1977.

[Rin92] C. Ringeissen. Unification in a combination of equational theories with shared constants and its application to primal algebras (extended version). In A. Voronkov, editor, *Proc. of International Conference Logic Programming and Automated Reasoning, St Petersburg (Russia)*, pages 261–272. Springer, jul 1992. Lecture Notes in Artificial Intelligence, Subseries of LNCS, 624.

[Rin93] C. Ringeissen. *Combinaison de Résolution de Contraintes*. PhD thesis, Université Nancy 1, Nancy, France, 1993.

[Rin96a] C. Ringeissen. Combining decision algorithms for matching in the union of disjoint equational theories. *Journal of Information and Computation*, 126(2) :144–160, May 1996. Journal version of 93-R-249.

[Rin96b] C. Ringeissen. Cooperation of decision procedures for the satisfiability problem. In F. Baader and K.U. Schulz, editors, *Frontiers of Combining Systems : Proceedings of the 1st International Workshop, Munich (Germany)*, Applied Logic, pages 121–140. Kluwer Academic Publishers, March 1996.

[Rin03] C. Ringeissen. Matching in a class of combined non-disjoint theories. In *Proc. of Automated Deduction - (CADE'03), 19th International Conference on Automated Deduction Miami Beach, FL, USA, July 28 - August*

2, volume 2741 of *Lecture Notes in Computer Science*, pages 212–227. Springer, 2003.

[RRT03] S. Ranise, C. Ringeissen, and D.-K. Tran. Rule-Based Algorithms for Combining Nelson-Oppen Theories and Shostak Theories. In *Proc. of the Third Workshop on Cooperative Solvers in Constraint Programming (Satellite Event to CP'2003)*, Kinsale, County Cork, Ireland, September 2003.

[RRT04] S. Ranise, C. Ringeissen, and D.-K. Tran. Nelson-Oppen, Shostak and the Extended Canonizer : A Family Picture with a Newborn. In *Proc. of First International Colloquium on Theoretical Aspects of Computing, (ICTAC'04), Guiyang, China*, volume 3407 of *Lecture Notes in Computer Science*, pages 372–386. Springer, Sep 2004.

[RRT06] S. Ranise, C. Ringeissen, and D.-K. Tran. Producing Conflict Sets for Combination of Theories. In B. Cook and R. Sebastiani, editors, *Pragmatics of Decision Procedures in Automated Reasoning (PDPAR'06)*, Seattle (WA), August 2006. Workshop affiliated to the 3rd International Joint Conference on Automated Reasoning, IJCAR.

[RRZ05] S. Ranise, C. Ringeissen, and C. G. Zarba. Combining data structures with nonstably infinite theories using many-sorted logic. In B. Gramlich, editor, *Proc. of the 5th Int. Workshop on Frontiers of Combining Systems (FroCoS'05)*, volume 3717 of *Lecture Notes in Computer Science*, pages 48–64, Vienna, Austria, September 2005. Springer.

[RS01] H. Rueß and N. Shankar. Deconstructing Shostak. In *Proc. of the 16th Annual IEEE Symposium on Logic in Computer Science, (LICS'01), Boston, Massachusetts, USA*, pages 19–28. IEEE Computer Society, June 2001.

[RS02] H. Rueß and N. Shankar. Combining shostak theories. In S. Tison, editor, *Proc. of the 13th International Conference on Rewriting Techniques and Applications (Copenhagen, Denmark)*, volume 2378 of *Lecture Notes in Computer Science*, pages 1–18. Springer, July 2002.

[RSJM99] G. Ramalingam, J. Song, L. Joscovicz, and R. E. Miller. Solving Difference Constraints Incrementally. *Algorithmica*, 23 :261–275, 1999.

[RT03] C. Ringeissen and C. Tinelli. Unions of non-disjoint theories and combinations of satisfiability procedures. *Theoretical Computer Science*, 290(1) :291–353, January 2003.

[Rus87] M. Rusinowitch. On termination of the direct sum of term-rewriting systems. *Information Processing Letters*, 26(2) :65–70, 1987.

[RV95] M. Rusinowitch and L. Vigneron. Automated Deduction with Associative-Commutative Operators. *Applicable Algebra in Engineering, Communication and Computation*, 6(1) :23–56, January 1995.

[RV01] A. Riazanov and A. Voronkov. Splitting without backtracking. In *Proc. of 7th International Joint Conference on Artificial Intelligence, (IJCAI'01) Seattle, Washington, USA, August 4–10*, pages 611–617, 2001.

[RV02] A. Riazanov and A. Voronkov. The design and implementation of VAMPIRE. *AI Commun.*, 15(2) :91–110, 2002.

[RW69] A. Robinson and L. Wos. Paramodulation and Theorem Proving in first-order theories with equality. *Machine Intelligence*, 4 :135–150, 1969.

[SBD02] Aaron Stump, Clark W. Barrett, and David L. Dill. CVC : a cooperating validity checker. In J. C. Godskesen, editor, *Proc. of Computer Aided Verification, 14th International Conference, (CAV'02),Copenhagen, Denmark, July 27-31*, Lecture Notes in Computer Science. Springer, 2002.

[SBDL01] A. Stump, C. Barrett, D. Dill, and J. Levitt. A Decision Procedure for an Extensional Theory of Arrays. In *Proc. of 16th IEEE Symposium on Logic in Computer Science, (LICS'01), Boston University, USA, June 16–19*, pages 29–37. IEEE Computer Society, 2001.

[Sch86] A. Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, 1986.

[Sch02] S. Schulz. E – A Brainiac Theorem Prover. *Journal of AI Communications*, 15(2/3) :111–126, 2002.

[Sho79] R. E. Shostak. A practical decision procedure for arithmetic with function symbols. *Journal of the ACM*, 26(2) :351–360, April 1979.

[Sho84] R. E. Shostak. Deciding combinations of theories. *Journal of the ACM*, 31 :1–12, 1984.

[SS89] M. Schmidt-Schauß. Unification in a combination of arbitrary disjoint equational theories. *Journal of Symbolic Computation*, 8 :51–99, 1989.

[SS06] V. Sofronie-Stokkermans. Interpolation in local theory extensions. In U. Furbach and N. Shankar, editors, *Proc. of the third International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 235–250, Seattle, USA, 2006. Springer.

[ST05] A. Stump and L.-Y. Tang. The Algebra of Equality Proofs. In *Proc. of the 16th International Conference on Rewriting Techniques and Applications (RTA)*, volume 3467 of *LNCS*, pages 469–483. Springer, 2005.

[Sti81] M. E. Stickel. A unification algorithm for associative-commutative functions. *Journal of ACM*, 28(3) :423–434, 1981.

[TH96] C. Tinelli and M. T. Harandi. A new correctness proof of the Nelson–Oppen combination procedure. In F. Baader and K. U. Schulz, editors, *Proc. of Frontiers of Combining Systems : Proceedings of the 1st International Workshop, (FroCos'96) Munich, Germany*, Applied Logic, pages 103–120. Kluwer Academic Publishers, March 1996.

[Tid86a] E. Tidén. Unification in combinations of collapse-free theories with disjoint sets of function symbols. In *Proc. of the 8th International Conference on Automated Deduction, (CADE'86), Oxford, England, July 27 - August 1*, volume 230 of *Lecture Notes in Computer Science*, pages 431–449. Springer, 1986.

[Tid86b] Erik Tidén. *First order unification in combinations of equational theories.* PhD thesis, The Royal Institute of Technology, Stockholm, 1986.

[Toy87a] Y. Toyama. Counterexamples to termination for the direct sum of term rewriting systems. *Information Processing Letters*, 25(3) :141–143, 1987.

[Toy87b] Y. Toyama. On the church-rosser property for the direct sum of term rewriting systems. *Journal of ACM*, 34(1) :128–143, 1987.

[TZ05] C. Tinelli and C. G. Zarba. Combining nonstably infinite theories. *Journal of Automated Reasoning*, 34(3) :209–238, April 2005.

[vHG92] P. van Hentenryck and T. Graf. Standard Forms for Rational Linear Arithmetics in Constraint Logic Programming. *Annals of Mathematics and Artificial Intelligence*, 5 :303–319, 1992.

[Vig94] L. Vigneron. Superposition in AC Theories : Proof of Completeness by Semantic Trees. Research Report 94-R-045, Centre de Recherche en Informatique de Nancy, Vandœuvre-lès-Nancy (France), March 1994.

[Wei97] Christophe Weidenbach. Spass version 0.49. *Journal of Automated Reasoning*, 14(2) :247–252, 1997.

[Wei01] C. Weidenbach. Combining superposition, sorts and splitting. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, chapter 27, pages 1965–2012. Elsevier, 2001.

[Yel87] K. A. Yelick. Unification in combinations of collapse-free regular theories. *Journal of Symbolic Computation*, 3(1/2) :153–181, 1987.

[YM05] G. Yorsh and M. Musuvathi. A combination method for generating interpolants. In *Proc. of Automated Deduction, (CADE'05), 20th International Conference on Automated Deduction, Tallinn, Estonia, July 22-27*, volume 3632 of *Lecture Notes in Computer Science*, pages 353–368. Springer, 2005.

[Zar04] C. G. Zarba. C-tableaux. Research Report RR-5229, Institut National de Recherche en Informatique et Automatique, 2004.

Index