Élimination des quantificateurs sur la théorie linéaire des réels

David Monniaux

5 septembre 2008

Nous considérons des formules écrites à l'aide des connecteurs logiques $\vee$, $\wedge$, $\neg$, des quantificateurs $\forall$ et $\exists$ sur un ensemble V de variables, et des formules atomiques $l_1 \leq l_2$, $l_1 < l_2$, $l_1 = l_2$, $l_1 \geq l_2$, $l_1 > l_2$ et $l_1 \neq l_2$ où l_1 et l_2 sont des expressions affines linéaires à coefficients entiers à variables dans V. Par souci de simplicité, nous nous restreindrons sans perte de généralité aux cas où les formules atomiques sont le la forme $l \geq 0$, les inégalités strictes s'obtenant par négation d'inégalités larges.

Ces formules s'interprètent directement sur $\mathbb{Q}$ ou $\mathbb{R}$: une formule à n variables libres a pour modèle un sous-ensemble de $\mathbb{Q}^n$ ou $\mathbb{Q}^n$. Il est évident qu'une formule est satisfiable sur $\mathbb{Q}$ si et seulement si elle l'est sur $\mathbb{R}$. Nous parlerons de « théorie linéaire des réels ».

Cette théorie admet l'élimination des quantificateurs : étant donné une formule F avec des quantificateurs, il existe une formule F' sans quantificateurs et avec les mêmes modèles. Cette transformation est algorithmique.

Tous les algorithmes d'élimination des quantificateurs pour cette théorie se ramènent à l'élimination d'un quantificateur existentiel $\exists x\ F$, où F est une formule sans quantificateurs, parfois en permettant l'élimination simultanée d'un bloc $\exists x_1, \ldots, x_n\ F$. Pour se ramener à ce cas, on procède par récurrence sur la structure syntaxique de la formule : si l'on rencontre un quantificateur existentiel, on l'élimine, et pour un quantificateur universel $\forall x\ F$ on le réécrit en $\neg\exists x\ \neg F$ et on procède de même. Par la suite, nous ne discuterons donc que le cas du quantificateur existentiel unique.

1 Élimination de quantificateurs existentiels sur les conjonctions

Pour nous convaincre qu'il est possible d'éliminer algorithmiquement les quantificateurs, nous allons d'abord donner un algorithme naïf et inefficace. On passe d'abord F en forme normale disjonctive et on se ramène donc à éliminer les quantificateurs dans $\exists x\ (F_1 \vee \cdots \vee F_n)$ où les F_i sont des conjonctions d'inégalités linéaires, donc à $(\exists x\ F_1) \vee \cdots \vee (\exists x\ F_n)$. On se ramène donc à l'élimination de quantificateurs sur des conjonctions d'inégalités linéaires (dans le pire cas, en nombre exponentiel).

1.1 Algorithme de Fourier-Motzkin

Considérons une conjonction C d'inégalités linéaires, larges dans un premier temps ($l \geq 0$). En séparant les inégalités où x apparaît avec un coefficient strictement positif, strictement négatif ou nul, on peut mettre cette conjonction sous la forme :

$$\begin{cases} x \geq l_1^-(y, z, \dots) \\ \vdots \\ x \geq l_a^-(y, z, \dots) \\ x \leq l_1^+(y, z, \dots) \\ \vdots \\ x \leq l_b^+(y, z, \dots) \\ l_1(y, z, \dots) \geq 0 \\ \vdots \\ l_c(y, z, \dots) \geq 0 \end{cases} \tag{1}$$

Autrement dit :

$$\begin{cases} x \geq \max\left(l_1^-(y, z, \dots), \dots, l_a^-(y, z, \dots)\right) \\ x \leq \min\left(l_1^+(y, z, \dots), \dots, l_b^+(y, z, \dots)\right) \\ l_1(y, z, \dots) \geq 0 \\ \vdots \\ l_c(y, z, \dots) \geq 0 \end{cases} \tag{2}$$

$\exists\, xC$ est donc équivalent à :

$$\max\left(l_1^-(y, z, \dots), \dots, l_a^-(y, z, \dots)\right) \leq \min\left(l_1^+(y, z, \dots), \dots, l_b^+(y, z, \dots)\right) \\ \wedge\, l_1(y, z, \dots) \geq 0 \wedge \dots l_c(y, z, \dots) \geq 0 \tag{3}$$

$\max(\alpha_1, \dots, \alpha_a) \leq \min(\beta_1, \dots, \beta_b)$ est équivalent à la conjonction de tous les $\alpha_i \leq \beta_j$. On obtient ainsi un système de $ab+c$ inégalités linéaires. Si $n = a+b+c$ est le nombre total d'inégalités avant élimination, le nombre d'inégalités après élimination est inférieur à $n^2/4$. Notons qu'une grande partie de ces inégalités peut être superflue (c'est-à-dire qu'il s'agit d'inégalités impliquées par les autres et que l'on pourrait supprimer sans changer le résultat).

Le procédé se généralise aux systèmes d'inégalités mixtes larges et strictes : quand on combine deux inégalités large on obtient une inégalité large, sinon on obtient une inégalité stricte.

Si l'on fait q éliminations successives, la borne citée plus haut nous permet de conclure qu'on obtient au maximum $n^{2^q}/4^q$. Cette double exponentielle est de mauvaise augure, et on peut se demander si une passe de simplification, supprimant les inégalités superflues, ne permettrait pas d'améliorer la complexité.

1.2 Polyèdres convexes

L'ensemble des solutions d'un système d'inégalités linéaires à n variables est un polyèdre convexe dans $\mathbb{Q}^n$ ou $\mathbb{R}^n$. Si le polyèdre est d'intérieur non vide, chaque face (de dimension $n-1$) du polyèdre correspond à une inégalité du

système d'origine ; et les inégalités qui ne correspondent à aucune face sont superflues. Un polyèdre non vide mais d'intérieur vide s'obtient si certaines contraintes impliquent des relations d'égalité.

Un polyèdre convexe peut être représenté comme solution d'un système d'inégalités linéaires (représentation par contraintes), ou encore comme enveloppe convexe d'un système de points (sommets du polyèdre), et, pour les polyèdres infinis, de droites et de demi-droites (représentation par générateurs). Passer d'une représentation à l'autre est coûteux, car l'une peut être exponentiellement plus grande que l'autre : il suffit de considérer par exemple un hypercube à n dimensions, qui est donné par $2n$ contraintes mais a 2^n sommets.

Il existe une riche littérature sur l'algorithmique sur les polyèdres convexes, notamment stimulée par les applications en analyse de programmes [Cousot and Halbwachs, 1978] ; Bagnara et al. en donne un panorama. Il existe plusieurs bibliothèques libres manipulant des polyèdres en double représentation (par contraintes et par générateurs), avec des calculs sur des entiers en précision étendue (indispensable dans un contexte où l'on ne peut borner d'avance les coordonnées) ; citons notamment NewPolka, maintenant intégrée dans Apron,[1] et la *Parma Polyhedra Library* (PPL).[2] Nous avons utilisé ces deux bibliothèques dans nos expérimentations.

L'élimination de q variables quantifiées existentiellement revient géométriquement à projeter q dimensions du polyèdre et à prendre la représentation par contraintes du polyèdre projeté. Le calcul de projection est très simple sur la représentation par générateurs (il suffit de projeter les sommets, droites et demi-droites), l'algorithme usuel de projection est donc de passer en représentation par générateurs, de projeter et de repasser en représentation par contraintes. Ceci nous donne une borne sur le nombre de faces du polyèdre projeté : 2^{2^n}, encore une double exponentielle.

Peut-on vraiment atteindre cette borne doublement exponentielle ? Nous n'avons pas trouvé d'exemple dans la littérature [Knox, 1996]. Nous pouvons par contre démontrer que, du moins en dimensions suffisamment grande, projeter q dimensions peut multiplier le nombre de faces par (presque) 2^q :

Lemme 1. *Il existe une famille p_n de polyèdres bornés dans $\mathbb{R}^3$, indexée par n, telle que p_n a $n+3$ faces mais dont la projection sur un plan a $2n+2$ côtés.*

Démonstration. Nous avons dessiné la construction pour $n = 4$ (Fig. 1). On trace d'abord un polygone convexe à $n+1$ côtés dans le plan (x, z) plane (en bas de la figure : côtés H, $1, \ldots, n$), avec un côté $z = 0$ edge et les autres côtés en une sorte de forme semicirculaire (par exemple en prenant la moitié d'un polygone régulier à $2n$ côtés). Ce polygone est ensuite extrudé le long de l'axe y, et le prisme ainsi formé est tranché à angle aigu en ses deux extrémités (lignes en gras A and B à droite de la figure), produisant ainsi un polyèdre à $n+3$ faces. Sa projection sur le plan (x, y) a $2n+2$ faces, comme on le voit bien sur la figure. □

Corollaire 2. *Pour tout $0 < \alpha < 2$ il existe une famille de polyèdres p'_k indexée par k tel que p_k soit de dimension $3k$, avec un nombre de faces f_k, et tel qu'on puisse choisir une direction de projection en espace de dimension $2k$ telle que la projection ait plus que $f_k \alpha^k$ faces.*

[1] `http://apron.cri.ensmp.fr/`
[2] `http://www.cs.unipr.it/ppl/`

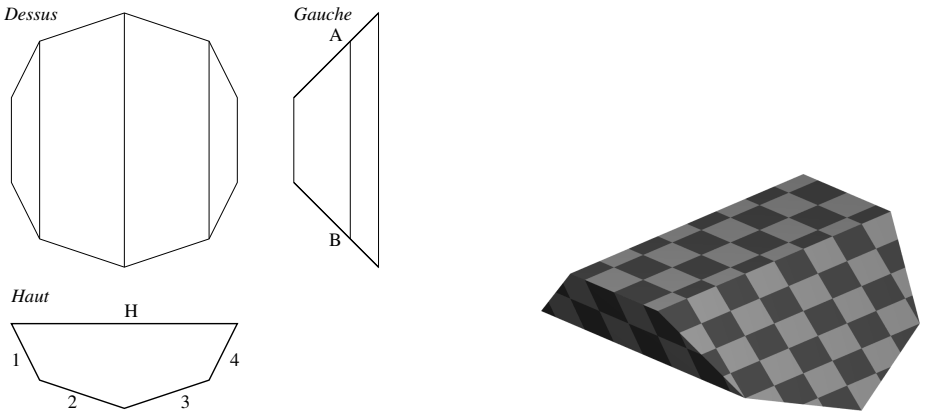

FIG. 1 – Ce polyèdre à 7 faces dans l'espace de dimension 3, dessiné selon les conventions du dessin technique, se projette en dimension 2 en un polygone à 10 côtés. On généralise dans le lemme 1 cette construction à un polyèdre à $n+3$ faces mais dont la projection a $2n+2$ côtés.

Démonstration. Dans un espace à $3k$ dimensions, prendre le produit de k polyèdres à $n+3$ faces dans $\mathbb{R}^3$ construits par le lemme précédent. Ce produit est un polyèdre à $(n+3)^k$ faces. Sa projection éliminant les dimensions d'indices $3+3i$ $(1 \leq i \leq k)$ a $(2n+2)^k$ faces. Avec n suffisamment grand, $\frac{(2n+2)^k}{(n+3)^k} > \alpha^k$. □

1.3 Témoins de vide

Chaque fois que l'on élimine une variable par l'algorithme de Fourier-Motzkin, les inégalités résultantes sont des combinaisons linéaires à coefficients entiers positifs ou nuls des inégalités d'origine. Si l'on élimine toutes les variables $\exists x_1, \ldots, x_n\ C$ d'une conjonction C, on obtient une conjonction d'inégalités triviales, à zéro variables, équivalente à $\exists x_1, \ldots, x_n\ C$, ces inégalités étant combinaisons linéaires à coefficients positifs de celles de C. Si C est non satisfiable, alors une de ces inégalités va être, à une constante multiplicative près, $-1 \geq 0$ ou $0 > 0$. On obtient ainsi une version du lemme de Farkas :

Lemme 3. *Un système d'inégalités linéaires, strictes ($l > 0$) ou larges ($l \geq 0$), n'a pas de solution si et seulement si il existe une combinaison linéaire à coefficients positifs ou nuls de ces inégalités qui soit trivialement fausse ($-1 \geq 0$ ou $0 > 0$).*

Notons que le vecteur de coefficients ainsi produit est un *témoin* du fait que le système d'origine n'ait pas de solution. Ce témoin permet de vérifier rapidement ce fait sans avoir à relancer un algorithme complexe, et fournit une *preuve* simple d'absence de solutions.

Soit n le nombre d'inégalités et m le nombre de variables, $\alpha_1, \ldots, \alpha_n$ les coefficients de la combinaison linéaire. La contrainte que la combinaison donne $-1 \geq 0$ s'exprime par m relations d'égalités, auxquelles on adjoint n relations de la forme $\alpha_i \geq 0$. La contrainte que la combinaison donne $0 > 0$ s'exprime par m relations d'égalités, auxquelles on adjoint n relations de la forme $\alpha_i \geq 0$ et une relation $\sum \alpha_{s_i} = 1$ où les s_i sont les indices des inégalités strictes.

Ainsi, le témoin de l'absence de solutions d'un système d'inégalités linéaires s'exprime lui-même comme la solution d'un système d'inégalités linéaires. La recherche d'un point solution d'un système d'inégalités linéaires est un cas particulier de *programmation linéaire*, problème classique de recherche opérationnelle sur lequel existe une ample littérature [Dantzig, 1998].

2 Méthodes par substitution

L'idée de ces méthodes est de remplacer une disjonction infinie (quantification existentielle) $\exists x\ F(x)$ par une disjonction finie $F(x_1) \vee \cdots \vee F(x_m)$ où $x_1, \ldots, x_m$ soit des « témoins » bien choisis, dont l'expression dépend en général des variables libres de la formule.

Considérons une formule F sans quantificateurs dont les variables libres sont $x, y, z, \ldots$. Sans perte de généralité, nous pouvons repousser les négations aux feuilles de l'arbre syntaxique en $\wedge$ et $\vee$, de sorte que ces feuilles sont de la forme $l \geq 0$ ou $\neg(l \geq 0)$, autrement dit $l < 0$ (*forme normale négative*).

Fixons $y, z, \ldots$ et étudions l'ensemble des modèles de F relativement à x. Cet ensemble est une union (éventuellement vide) d'intervalles dont les bornes sont soit $\pm\infty$, soit les solutions ρ_i pour x des équations $l_i(x, y, z, \ldots) = 0$ où les $l_1, \ldots, l_m$ sont les expression linéaires présentes dans F. L'idée des algorithmes de Ferrante and Rackoff [1975][Bradley and Manna, 2007, §7.3][Nipkow, 2008, §4.2] et Loos and Weispfenning [1993a][Nipkow, 2008, §4.4] est d'utiliser le fait que la valeur de vérité de F ne peut changer qu'en cet ensemble de m points $\rho_1, \ldots, \rho_m$. L'idée est de choisir au moins un représentant x_i par intervalle où la fonction est constante.

Dans les deux cas, on commence par distinguer les intervalles extrêmaux (ceux de la forme $(-\infty, \rho_i)$ ou $(\rho_i, +\infty)$). On obtient la valeur de $F(-\infty)$ et $F(+\infty)$ par simple substitution arithmétique (si on admet l'arithmétique sur les infinités) ou, plus simplement, par *substitution virtuelle* : si la formule atomique $l \geq 0$ ne fait pas intervenir x, on la laisse intacte, si x a un coefficient strictement positif dans l on remplace par vrai (pour $x \mapsto +\infty$) ou faux (pour $x \mapsto -\infty$), et l'inverse pour un coefficient strictement négatif. Le choix des représentants des autres intervalles diffère entre les deux algorithmes.

Points intérieurs Ferrante and Rackoff [1975] remarquent que comme $F(x)$ est constante sur les intervalles (ρ_i, ρ_j), il suffit de prendre comme témoins les milieux de ces intervalles. En général, cela va donner $m(m+1)/2$ valeurs différentes où m est le nombre de formules atomiques. À chaque élimination de variable, la taille de la formule est donc élevée au cube (à constante multiplicative près). Ceci donne une complexité globale en $|F|^{2^{cq}}$ où $|F|$ est la taille de la formule d'origine et q est le nombre de quantificateurs à éliminer.

On rajoute ensuite $F(+\infty)$ et $F(-\infty)$.

Infinitésimaux Chaque intervalle de positivité de F a pour extrémité gauche un des ρ_i, qui doit qui plus est correspondre :

- à une formule atomique de la forme $l \geq 0$ où x a un coefficient strictement positif si l'intervalle est fermé en son extrémité gauche ;
- à une formule atomique de la forme $l < 0$ où x a un coefficient strictement négatif si l'intervalle est ouvert en son extrémité gauche.

Pour obtenir les extrémités gauches fermées, on prend donc dans les x_i tous les ρ_j correspondant à des contraintes $l \geq 0$ où x a un coefficient strictement positif. Pour les extrémités gauches ouvertes, on prend $\rho_j + \epsilon$, où les ρ_j correspondent à des contraintes $l < 0$ où x a un coefficient strictement négatif et ϵ est un infinitésimal, c'est à dire un élément rajouté strictement positif et plus petit que tous les réels strictement positif.[3]

On pourrait bien sûr procéder par calcul sur les infinitésimaux, mais là encore on peut procéder par substitution virtuelle. Considérons une formule atomique de la forme $cx + d(y, z, \dots) \geq 0$, où l'on veut remplacer x par $\rho + \epsilon$. Si $c > 0$, alors on obtient $c\rho + d(y, z, \dots) \geq 0$. Si $c < 0$, on obtient $c\rho + d(y, z, \dots) > 0$.

On rajoute ensuite $F(-\infty)$. La méthode fonctionne également si on considère les extrémités droites d'intervalles (substituer x par $-x$). On pourra choisir la direction qui nécessite le moins de points.

L'avantage de cette méthode est que le nombre de points x_i est linéaire en la taille de la formule, alors que la méthode de Ferrante and Rackoff [1975] utilise un nombre quadratique de points. On a donc une méthode là aussi en $|F|^{2^{cq}}$ mais avec une meilleure constante.

Ces méthodes sont des méthodes syntaxiques, qui conservent (en la dupliquant) la structure de la formule d'origine sans jamais essayer de la simplifier. La géométrie du problème est ignorée.

3 Méthode géométrique

Au §1, nous avions introduit une méthode naïve pour l'élimination des quantificateurs : tout passer d'abord en forme normale disjonctive, puis projeter séparément chaque conjonction. Géométriquement, cela revient à représenter l'ensemble des solutions comme une union explicite de polyèdres convexes, puis à projeter chaque polyèdre (§1.2).

Nous avons proposé un algorithme améliorant cette idée naïve par deux idées [Monniaux, 2008] :

1. On génère les éléments de la disjonction et on les projette au fur et à mesure, en accumulant le résultat dans la solution. On ne génère pas les polyèdres qui se projettent totalement dans la solution déjà générée (ils n'ajoutent rien).
2. Pour obtenir les éléments de la disjonction, on utilise non pas des moyens syntaxiques, mais un algorithme de satisfiabilité modulo théorie (SMT).

Le problème de satisfiabilité booléenne (SAT) est bien connu comme problème NP-complet Cook [1971]. Il existe pour ce problème divers algorithmes, souvent dérivés de l'algorithme DPLL [Davis and Putnam, 1960], de nombreux outils efficaces de résolution, un format de fichier standard [DIM], des bibliothèques de *benchmarks* Hoos and Stützle [2000] et même une compétition.[4] Le problème est simple : étant donné une formule propositionnelle, dire si elle est satisfiable et, si elle l'est, donner un modèle.

[3] On peut définir ainsi le calcul sur les infinitésimaux : les quantités sont de la forme $a + b\epsilon$ où a et b sont des rationnels, et on les ordonne en prenant l'ordre lexicographique sur (a, b).

[4] `http://www.satcompetition.org/`

Ce cadre a été par la suite étendu à des formules non propositionnelles, c'est-à-dire dont les atomes sont pris dans une théorie, par exemple, ce qui nous intéresse ici, la théorie des inégalités linéaires sur les réels. Il s'agit d'un sujet de recherche actif, et les algorithmes usuels [Ganzinger et al., 2004] mêlent une recherche SAT et une procédure de décision pour les conjonctions de la théorie considérée. Nous avons principalement utilisé l'outil YICES[5] mais nous avons également implémenté un algorithme SMT naïf et « paresseux », permettant notamment de vérifier à quel point l'efficacité de notre algorithme d'élimination des quantificateurs dépendait de l'excellence de l'algorithme SMT utilisé. Cet algorithme naïf fonctionne ainsi :

1. Remplacer chaque inégalité linéaire $l \geq 0$ par une variable propositionnelle, à l'aide d'un dictionnaire.
2. Résoudre le problème SAT propositionnel. S'il n'a pas de solution, le problème d'origine n'en a pas.
3. S'il a une solution, celle-ci définit une conjonction de littéraux, donc un polyèdre. Si celui-ci contient un point, on le propose comme solution.
4. Sinon, on calcul un témoin de vide (§1.3) : une conjonction insatisfiable, donc indésirable, de littéraux et on rajoute sa négation au problème.

Ayant à notre disposition un SMT-solveur et une bibliothèque de calcul sur les polyèdres, nous pouvons maintenant proposer un algorithme pour éliminer le quantificateur dans $\exists\, x_1, \ldots, x_n\ F$ et fournir une solution S en forme normale disjonctive :

1. Le SMT-solveur propose un modèle de F ; s'il n'y en a pas, F est insatisfiable et on termine.
2. Ce modèle se généralise immédiatement à tous les points de même valuation par rapport aux formules atomiques de F (les points qui vérifient les mêmes inégalités). On obtient ainsi une conjonction C de littéraux telle que $C \implies F$.
3. On généralise cette conjonction en supprimant des littéraux de C tant que $C \implies F$.
4. On calcule la projection P de C (élimination de $\exists\, x_1, \ldots, x_n\ P$). On rajoute P à S ($S := S \vee P$) et on retranche P de F ($F := F \wedge \neg P$). On reprend au départ.

Cet algorithme termine forcément, en au maximum 2^n itérations où n est le nombre d'inégalités dans F : chaque itération « consomme » un valuation de ces inégalités.

Notons également qu'il est possible d'appliquer l'algorithme pour éliminer 0 quantificateurs, ce qui équivaut à demander une mise sous forme normale disjonctive. En appliquant le même algorithme à la négation de la formule, on obtient une mise en forme normale disjonctive. On peut simplifier la formule en alternant les mises sous les deux formes normales.

Nous avons implémenté notre algorithme, ainsi que ceux de Ferrante and Rackoff [1975] et Loos and Weispfenning [1993a] dans notre outil MJOLLNIR, que nous avons ensuite comparé à des outils existants, d'une part sur quelques instances isolées (table 1) et sur des instances aléatoires (table 2).

[5] `http://yices.csl.sri.com/`

Mjollnir est l'algorithme décrit ici, implémenté avec le SMT-solveur YICES et la bibliothèque de polyèdres NEWPOLKA via APRON, ou optionnellement la *Parma Polyhedra Library*. L'analyse de performances montre que la grande majorité du temps est passé en résolution SMT, de sorte que les éventuelles différences entre bibliothèques de polyèdres sont négligeable

Naïf est une version préliminaire du même algorithme, implémentée avec l'algorithme de SMT naïf et paresseux vu plus haut.

Mjollnir (mod1) calcule une forme normale disjonctive par ALL-SAT, avant de projeter (option `-no-block-projected-model`).

Mjollnir (mod2) est une variante de notre algorithme censée, au prix d'une complication, être plus efficace [Monniaux, 2008, §4.2] (option `-add-blocking-to-g`).

Mjollnir Ferrante-Rackoff , voir §2.

Mjollnir Loos-Weispfenning , voir §2.

Lira[6] est un outil basé sur les automates de Büchi qui traite les problèmes linéaires mixtes rationnels et entiers (donc l'arithmétique de Presburger).

Mathematica[7] est un logiciel d'algèbre symbolique. Sa fonction `Reduce` implémente, semble-t-il, la décomposition cylindrique algébrique Caviness and Johnson [1998], un algorithme d'élimination des quantificateurs dans la théorie des corps réels clos.

Redlog[8] est une extension du logiciel d'algèbre symboliqueREDUCE 3.8.[9] REDLOG implémente divers algorithmes dûs à Volker Weispfenning et son équipe Loos and Weispfenning [1993b].

Benchmark	r. lim. $\mathbb{R}$	r. lim. float	`prsb23`	`blowup5`
MJOLLNIR	1.4	17	0.06	negligible
MJOLLNIR (mod1)	1.6	77[a]	0.06	negligible
MJOLLNIR (mod2)	1.5	34	0.07	negligible
MJOLLNIR Loos-Weispfenning	o-o-m	o-o-m	o-o-m	negligible
Proof-of-concept	n/a	823	n/a	n/a
MJOLLNIR Ferrante-Rackoff	o-o-m	o-o-m	o-o-m	negligible
Proof-of-concept	n/a	823	n/a	n/a
LIRA	o-o-m	o-o-m	8.1	0.6
REDLOG `rlqe`	182	o-o-m	1.4	negligible
REDLOG `rlqe`+`rldnf`	o-o-m	o-o-m	n/a	n/a
MATHEMATICA `Reduce`	(> 12000)	o-o-m	(> 780)	7.36

[a]Memory consumption grows to 1.1 GiB.

TAB. 1 – Comparaison entre les différents logiciels d'élimination de quantificateurs sur des instances isolées provenant de vérification de programmes (deux premières colonnes) et des exemples de LIRA.

Notre algorithme fournit des formules en forme normale disjonctive dont les atomes sont pris parmi un ensemble à au maximum $s^{2^q}/4^q$ éléments (voir §1.1), où q est le nombre de variables à éliminer et s est le nombre d'inégalités dans la

[6]`http://lira.gforge.avacs.org/`
[7]`http://www.wolfram.com/`
[8]`http://www.algebra.fim.uni-passau.de/~redlog/`
[9]`http://www.uni-koeln.de/REDUCE/`

	depth 14			depth 15			depth 16		
	Résolus	Moy	O-o-m	Résolus	Moy	O-o-m	Résolus	Moy	O-o-m
Mjollnir	100	1.6	0	94	9.8	0	73	35.3	0
Mjollnir (mod1)	94	8.2	3	80	27.3	7	39	67.1	25
Mjollnir (mod2)	100	3.8	0	91	13.9	0	65	39.2	0
Mjollnir Loos-W.	93	1.77	4	90	6.42	5	62	17.65	27
Naïf	94	1.4	0	86	2.2	0	55	17.7	0
Mjollnir Ferrante-R.	51	18.2	41	23	23.2	65	3	7.3	85
Naïf	94	1.4	0	86	2.2	0	55	17.7	0
Lira	14	102.4	83	3	77.8	94	1	8	95
Redlog (`rlqe`)	92	13.7	0	53	27.4	0	27	33.5	0
Mathematica	6	30.2	0	1	255.7	0	1	19.1	0

Tab. 2 – Comparaisons de performances sur 3×100 instances aléatoires générées par `randprsb`, un outil associé à Lira, avec des profondeurs de formules n respectivement 14, 15 and 16 (obtenues par `randprsb 0 7 -10 10` n i) avec i parcourant $[0, 99]$. Le tableau donne le nombre d'instances résolues sur les 100, le temps moyen par instance résolue, et le nombre moyen d'instances provoquant une insuffisance de mémoire (o-o-m).

formule F d'origine. La disjonction fournie a donc au maximum $2^{s^{2q}/4^q}$ termes. Au final, nous obtenons une borne de complexité en $2^{2^{2^{|F|}}}$. La complexité des algorithmes de Ferrante and Rackoff [1975] et Loos and Weispfenning [1993b] est seulement en $2^{2^{c|F|}}$; ces algorithmes ne fournissent cependant pas une forme normale disjonctive, dont le calcul rajouterait une exponentielle supplémentaire.

Même si notre algorithme a une borne de complexité supérieure à celle des algorithmes par substitution, les essais montrent qu'il se comporte mieux en pratique. Nous pensons que cela est dû au fait que les algorithmes par substitution ne simplifient jamais les formules qu'ils créent, qui grossissent à chaque étape, alors que notre algorithme restructure les formules à chaque étape et notamment élimine certaines disjonctions inutiles.

Références

Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. *The Parma Polyhedra Library, version 0.9*. available from `http://www.cs.unipr.it/ppl`.

Aaron R. Bradley and Zohar Manna. *The Calculus of Computation : Decision Procedures with Applications to Verification*. Springer, October 2007. ISBN 3540741127.

Bob F. Caviness and Jeremy R Johnson, editors. *Quantifier elimination and cylindrical algebraic decomposition*. Springer, 1998. ISBN 3-211-82794-3.

Stephen A. Cook. The complexity of theorem-proving procedures. In *Symposium on theory of computing (STOC)*, pages 151–158, 1971. doi : http://doi.acm.org/10.1145/800157.805047.

Patrick Cousot and Nicolas Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Principles of Programming Languages (POPL)*, pages 84–96. ACM, 1978. doi : 10.1145/512760.512770.

George Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1998.

Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3) :201–215, 1960. ISSN 0004-5411. doi : http://doi.acm.org/10.1145/321033.321034.

Satisfiability Suggested Format. DIMACS.

Jeanne Ferrante and Charles Rackoff. A decision procedure for the first order theory of real addition with order. *SIAM Journal of Computation*, 4(1) : 69–76, March 1975.

H. Ganzinger, G. Hagen, R. Nieuwenhuis, A. Oliveras, and C. Tinelli. DPLL(T) : Fast Decision Procedures. In R. Alur and D. Peled, editors, *16th International Conference on Computer Aided Verification, CAV'04*, volume 3114 of *Lecture Notes in Computer Science*, pages 175–188. Springer, 2004.

Holger H. Hoos and Thomas Stützle. SATLIB : An online resource for research on SAT. In *I. P. Gent and van Maaren, H. and T. Walsh*, pages 283–292. IOS Press, 2000. Available online at `http://www.satlib.org/`.

Steven Knox. *The Number of Facets of a Projection of a Convex Polytope*. PhD thesis, University of Illinois, 1996.

Rüdiger Loos and Volker Weispfenning. Applying linear quantifier elimination. *The Computer Journal*, 36(5) :450–462, 1993a.

Ruediger Loos and Volker Weispfenning. Applying linear quantifier elimination. *The Computer Journal*, 36(5) :450–462, 1993b. Special issue on computational quantifier elimination.

David Monniaux. A quantifier elimination algorithm for linear real arithmetic. In *LPAR*, LNCS, 2008. To appear.

Tobias Nipkow. Linear quantifier elimination. In *Automated reasoning (IJCAR)*, volume 5195 of *LNCS*, pages 18–33. Springer, 2008.