Mon projet professionnel

Ali AYAD
Docteur en Mathématiques et leurs applications
Université de rennes 1

Post-Doctorant en informatique
CEA-LIST-Saclay

ali.ayad@cea.fr
http://ali.ayad.free.fr/
Tél : 06 32 55 16 04

Table des matières

1 Situation actuelle

Actuellement, je suis en post-doc en informatique au CEA-LIST à Saclay. Je fais partie du laboratoire LSL (Laboratoire de Sûreté des Logiciels). Mon domaine d'intérêt actuel est la vérification et la certification de programmes C flottants par des méthodes formelles. Plus précisément, je travaille sur la vérification déductive de programmes flottants et sur l'analyse des valeurs flottantes de programmes C par interprétation abstraite dans la plateforme `Frama-C` [14], un outil de vérification formelle des codes C, avec des applications de nature industrielle dans le but de garantir la correction de logiciels critiques. `Frama-C` est construit par un ensemble de plugins dédiés à faire des tâches spécifiques. Par exemple, le plugin `Jessie` [25, 26] est destiné à la vérification déductive des programmes par la logique de Hoare et le calcul de la plus faible précondition, le plugin `ValViewer` pour l'analyse des valeurs des variables de programmes C par interprétation abstraite.

2 Activités d'enseignement

2.1 Description des enseignements

J'ai été en poste d'ATER (Attaché Temporaire d'Enseignement et de Recherche) pendant deux ans, d'abord à temps partiel à l'université de Rennes 1 pendant l'année 2006-2007, puis à temps complet à l'université de Rennes 2 pendant l'année 2007-2008. J'ai eu aussi l'occasion de donner des cours particuliers en mathématiques pour des élèves en Seconde, Première S et Terminal S. Dans le cadre de ces fonctions, j'ai assuré les travaux dirigés de plusieurs modules d'analyse, d'algèbre et du statistique en L1 et L2. J'ai par ailleurs rédigé plusieurs feuilles d'exercices en algèbre et en analyse ainsi que des sujets de partiels et des devoirs à la maison. J'ai également fait passer des interrogations orales durant mes deux années d'ATER.

2.2 Détails des enseignements

2.2.1 ATER à temps complet à l'université de Rennes 2 (2007-2008)

- TD Analyse MASS en Licence 1, S1 (24 heures).
  Fonctions usuelles, fonctions exponentielle, logarithme, trigonométriques et inverses, trigonométriques, hyperboliques, suites numériques, limite et continuité des fonctions d'une seule variable, équivalence des fonctions en un point et à l'infini.

- CM et TD Statistique AES en Licence 1, S2 (48 heures).
  Pourcentages et indices, statistique descriptive pour une et deux variables : représentations graphiques, résumés numériques, mesures de tendance centrale et mesures de dispersion.

– CM et TD "Préparation aux concours d'entrée IUFM, QCM de mathématiques" en Licence 3 (18 heures).
Rappels cours de base : calcul numérique, géométrie, arithmétique.

– TD "Initiation à la statistique" en Licence 2 Géographie (12 heures).
Statistique univariée et bivariée : mesures de tendance centrale et de dispersion, indépendance, moyenne conditionnelle, variance des moyennes conditionnelles, régression, méthode des moindres carrés.

– CM et TD Mathématiques Appliquées aux Sciences Sociales en Licence 1 AES.
Fonctions affines, exemples de fonctions usuelles, application de la dérivée, fonctions de deux variables, optimisation sous contraintes.

– CM et TD du cours "UED Mathématiques" en Licence 2, S4 (84 heures).
Calcul numérique, géométrie du plan, arithmétique.

2.2.2 ATER à temps partiel à l'université de Rennes 1 (2006-2007)

– TD Algèbre du cours "Ensembles, applications, relations et groupes" en Licence 2 (24 heures).
Théorie des ensembles, applications injectives, bijectives, relations d'ordre et relations d'équivalence, étude de la théorie de groupe.

– TP Algèbre (sous Maple) du cours "Matrices et systèmes linéaires" en Licence 2 (12 heures).
Opérations sur les matrices, calcul des valeurs propres et des vecteurs propres, élimination de Gauss, résolution des systèmes linéaires.

– CM et TD "Préparation aux concours d'entrée IUFM, QCM de mathématiques" en Licence 3 (36 heures).
Rappels cours de base : calcul numérique, géomérie, arithmétique.

– CM et TP du cours "Éducation aux choix professionnels" en Licence 1 (30 heures).
Recherche sur les métiers.

3 Programme de recherche

Cette partie résume les lignes directrices de mes activités actuelles de recherche ainsi que tout ce que je souhaite entreprendre lors des années à venir, et explique comment elles se situent dans la continuité de mes travaux de thèse et de mon post-doc.

Ma thèse se situe dans le domaine de `calcul formel` et complexité, intitulée "Com-

plexité de la résolution des systèmes algébriques paramétrés" [4, 5]. Un résumé détaillé de mes travaux de thèse se trouve dans la section 4.

À l'issue de ma thèse et pendant ma première année d'ATER à l'université de Rennes 1, je suis toujours resté dans le domaine de `calcul formel` et je me suis intéressé à étudier les bases de Gröbner paramétriques, la complexité de la factorisation des opérateurs différentiels linéaires et la résolution des équations différentielles polynomiales [8, 7]. Ces sujets seront détaillés dans la section 5.

Pendant ma deuxième année d'ATER à l'université de Rennes 2 et dans le but de trouver des applications de mes travaux en thèse, j'avais un poste d'un chercheur invité à l'IRISA de Rennes pour travailler sur la représentation des signaux par un petit nombre de paramètres en se basant sur les techniques de la théorie de diffusion (Scattering theory). Ce sujet sera détaillé dans la section 5. C'était la première fois que je m'affronte à l'implémentation des algorithmes numériques dans Matlab.

Pendant mon séjour postdoctoral de 5 mois dans l'équipe ProVal de l'INRIA-Île-de-France et pendant mon post-doc au CEA-List en 2008-2009, j'ai décrit deux modèles formels de vérification déductive des programmes C flottants. J'ai travaillé aussi sur l'analyse des valeurs des variables flottantes de programmes C. Ces deux sujets sont détaillés dans les deux paragraphes suivants :

3.1 Vérification déductive de programmes C flottants

Mon sujet d'intérêt pendant mon séjour postdoctoral au LRI de l'université Paris 11 et au CEA-List s'inscrit dans le contexte de la vérification formelle de sûreté de fonctionnement du logiciel. Les logiciels embarqués sont de plus en plus présents dans notre environnement, et remplissent souvent des fonctions critiques : énergie nucléaire, automobile, ferroviaire, aéronautique, satellites et robots. Pour ces logiciels, il est crucial d'offrir des outils permettant de garantir leur bon fonctionnement. La plateforme `Why` [33] développée dans l'équipe-projet ProVal de l'Inria est un outil d'analyse statique de programmes C (`Caduceus`) et Java (`Krakatoa`) par la logique de `Hoare`. `Why` génère des obligations de preuve (OP) à partir d'un programme annoté, celles-ci décrivent les variables du programme dans un point donné. Les OP générées doivent être vérifiées par des prouveurs automatiques (Simplify, Ergo, Yices, CVC Lite, CVC3, haRVey, Zenon) ou interactifs (Coq, PVS, Isabelle/HOL, HOL 4, HOL-Light, Mizar) supportés par `Why` pour assurer la validation du programme en question.

Mon travail était consacré sur les aspects flottants dans `Why`. Des méthodes formelles sont utilisées pour montrer la correction des programmes traitant de calculs à virgule flottante. J'ai développé deux modèles formels des flottants dans `Why` :

Le premier est appelé le modèle "**Défensive**", qui est une amélioration du modèle

présenté dans [10]. Ce modèle vérifie qu'il n'y a pas des débordements en mémoire (`overflow`) à l'exécution des opérations flottantes. Des obligations de preuve sont générées par Why pour prouver qu'il n'y a pas d'`overflow` à chaque fois qu'il y a une opération arithmétique flottante dans les programmes. Par exemple, la spécification complète de la division de deux variables x et y de type `double` est donnée par :

```
/*@ requires y <> 0 && \no_overflow(\Double,\NearestEven,x/y);
  @ ensures \result == \round(\Double,\NearestEven,x/y);
  @*/
```

où `\result` est le résultat de la division flottante de x par y. `\no_overflow(x/y)` est le prédicat de non-débordement du réel x/y en double précision, i.e., la valeur absolue de l'arrondi correct au plus près pair du réel x/y (i.e., `\round(\Double,\NearestEven,x/y)`) ne dépasse pas le plus grand réel représentable dans le format `double`. Les comparaisons sont faites sur les réels, les variables flottantes sont converties implicitement en réels. Cette spécification est écrite avec les syntaxes du langage `ACSL` [9], le langage de spécification de programmes C dans `Frama-C`.

Le deuxième modèle est appelé le modèle "**Full**". Ce modèle étend le modèle "**Défensive**" et permet d'avoir des débordements en mémoire dans les programmes en déclenchant des exceptions. Infinis, `NaN`s (Not-a-Number) et zéros signés sont considérés comme de valeurs spéciales comme dans la norme IEEE-754. Des prédicats spéciaux sont définis pour décrire la classe des variables flottantes comme par exemple, `\is_finite(x)`, `\is_infinity(x)`, `\is_NaN(x)`, etc ...

Ces deux modèles sont détaillés dans mon rapport [2] et dans mes deux articles [1, 3].

3.2 Analyse des valeurs flottantes de programmes C par interprétation abstraite

L'interprétation abstraite est une autre méthode formelle d'analyse statique de programmes. Dans le même esprit de la vérification déductive, l'interprétation abstraite permet d'avoir automatiquement des propriétés des variables de programmes. Je m'intéresse aux domaines numériques abstraits et plus précisement aux variables flottantes des programmes C. On distingue deux catégories de domaines numériques abstraits : les domaines non-relationnels (comme les intervalles) et les domaines relationnels (comme les octogones et les polyèdres).

En ce qui concerne les flottants, l'outil `ValViewer` de `Frama-C` implémente le domaine des intervalles. À l'heure actuelle, `ValViewer` permet d'avoir des bonnes bornes sur les expressions flottantes linéaires par l'arithmétique des intervalles. Je m'intéresse ici à implémenter des algorithmes d'optimisation des expressions flottantes non-linéaires. Ces

expressions sont celles qui apparaisent dans les programmes.

Dans un deuxième temps, je chercherai à exprimer de relations polynomiales entre les variables flottantes de programmes contenant des affectations polynomiales en décrivant un domaine numérique abstrait relationnel. Un tel domaine existe et s'appelle le domaine des variétés [27]. Il est basé sur le calcul des bases de Gröbner, la théorie d'élimination et la décomposition de variétés algébriques en composantes irréductibles que j'ai largement étudié dans ma thèse (voir ci-dessous).

4 Présentation analytique de mes travaux de recherche en thèse

Ma thèse se situe dans le domaine de calcul formel et analyse de complexité sur un sujet qui est à la base de la géomètrie algébrique, à savoir la résolution des systèmes polynomiaux, i.e., la décomposition des variétés algébriques en composantes irréductibles. Je me suis intéressé au cas où les coefficients des systèmes polynomiaux dépendent d'une manière polynomiale d'un nombre fini de paramètres. Dans ce cas il faut distinguer les valeurs des paramètres qui donnent de systèmes algébriques consistents, i.e., qui admettent de solutions. Ensuite décrire les solutions associées d'une manière uniforme en les paramètres. Mon but était de réaliser un algorithme qui fait ce travail et d'analyser et comparer sa complexité avec les algorithmes existants.

Plus précisement, soit F un corps commutatif, $f_1, \ldots, f_k \in F[u_1, \ldots, u_r, X_0, \ldots, X_n]$, k polynômes homogènes en $X = (X_0, \ldots, X_n)$ paramétrés par les variables $u = (u_1, \ldots, u_r)$. Le système $f_1 = \cdots = f_k = 0$ est appelé un système polynomial paramétré. L'espace $\overline{F}^r$ est appelé l'espace des paramètres où $\overline{F}$ est une clôture algébrique de F. Pour chaque $a = (a_1, \ldots, a_r) \in \overline{F}^r$ (spécialisation des paramètres), on note par $V^{(a)}$ la variété projective de $P^n(\overline{F})$ définie par les polynômes homogènes $f_1^{(a)}, \ldots, f_k^{(a)} \in \overline{F}[X_0, \ldots, X_n]$ où $f_i^{(a)} := f_i(a_1, \ldots, a_r, X_0, \ldots, X_n)$ pour tout $1 \leq i \leq k$, i.e., l'ensemble des zéros communs des polynômes $f_1^{(a)}, \ldots, f_k^{(a)}$ dans $P^n(\overline{F})$.

Il s'agit de résoudre les systèmes algébriques $f_1^{(a)} = \cdots = f_k^{(a)} = 0$ (i.e., décomposer les variétés $V^{(a)}$ en composantes absolument irréductibles) d'une manière uniforme en les paramètres dans le sens de l'exemple suivant (pour plus d'exemples et de détails sur les outils et les techniques utilisés dans la thèse, veuillez vous reporter au rapport de ma thèse à l'adresse http://ali.ayad.free.fr/Thse_Ayad.pdf).

Exemple 4.1 *Considérons le système algébrique paramétré suivant qui décrit l'état d'un*

bras articulé constitué de deux tiges de longueur 1 [17, 28, 29] :

$$\begin{cases} x_1 + x_2 &= u \\ x_3 + x_4 &= v \\ x_1^2 + x_3^2 &= 1 \\ x_2^2 + x_4^2 &= 1 \end{cases}$$

Les variables u et v sont les paramètres, les inconnues sont les variables $x_1, \ldots, x_4$. Une résolution paramétrique de ce système est donnée par [5, 28, 29] :

$$x_4^2 - vx_4 + \frac{1}{4}\frac{u^4 + 2u^2v^2 - 4u^2 + v^4}{u^2 + v^2} = 0, \quad \begin{cases} x_1 &= \frac{v}{u}x_4 + \frac{1}{2}\frac{u^2 - v^2}{u} \\ x_2 &= -\frac{v}{u}x_4 + \frac{1}{2}\frac{u^2 + v^2}{u} \\ x_3 &= -x_4 + v \end{cases}$$

Soit le sous-ensemble constructible U_1 de l'espace des paramètres $\mathbb{C}^2$ défini par les inéquations suivantes :

$$u \neq 0, \quad u^2 + v^2 \neq 0$$

Pour toute spécialisation (a, b) des paramètres (u, v) dans U_1 les solutions $(x_1, \ldots, x_4) \in \mathbb{C}^4$ du système correspondant sont obtenues par extraction des racines (i.e., x_4) d'une équation du second degré à coefficients dans $\mathbb{C}$. Les valeurs de (x_1, x_2, x_3) sont données en fonction de ces racines.
On se demande que se passe-t-il pour des valeurs (a, b) des paramètres dans $\mathbb{C}^2 \backslash U_1$, i.e., $a = 0$ ou $a^2 + b^2 = 0$?. En effet, on peut décomposer $\mathbb{C}^2$ en 4 ensembles constructibles $U_1, \ldots, U_4$ donnés par les équations et les inéquations qui les définissent ainsi pour chacun d'eux on donne une représentation paramétrique des solutions valable pour toute spécialisation des paramètres dans cet ensemble. Plus précisement,

$$U_1 = \{u \neq 0, u^2 + v^2 \neq 0\}, \quad \theta^2 - \frac{-u^4 + 4u^2 - u^2v^2}{u^2 + v^2} = 0, \quad \begin{cases} x_1 &= \frac{v}{2u}\theta + \frac{u}{2} \\ x_2 &= -\frac{v}{2u}\theta + \frac{u}{2} \\ x_3 &= -\frac{1}{2}\theta + \frac{3v}{2} \\ x_4 &= \frac{1}{2}\theta + \frac{v}{2} \end{cases}$$

$$U_2 = \{u \neq 0, u^2 + v^2 = 0\}, \quad \textit{pas de solutions.}$$

$$U_3 = \{u = 0, v \neq 0\}, \quad \theta^2 + \frac{v^2}{4} - 1 = 0, \quad \begin{cases} x_1 &= \theta \\ x_2 &= -\theta \\ x_3 &= \frac{v}{2} \\ x_4 &= \frac{v}{2} \end{cases}$$

$$U_4 = \{(0, 0)\}, \quad \theta^2 + t^2 - 1 = 0, \quad \begin{cases} x_1 &= \theta \\ x_2 &= -\theta \\ x_3 &= -t \\ x_4 &= t \end{cases}$$

On remarque que le système est zéro-dimensionnel sur U_1, U_2, U_3 (i.e., nombre fini de solutions) et de dimension 1 sur U_4 (i.e., nombre infini de solutions) où t est un paramètre qui prend des valeurs dans $\mathbb{C}$.

4.1 État de l'art du problème de résolution des systèmes algébriques paramétrés

Le problème de décomposition d'une variété algébrique en composantes irréductibles remonte à Bézout, Cayley, Kronecker, Macauley, Hermann, Van-der-Waerden et d'autres mathématiciens classiques. La résolution des systèmes polynomiaux est l'un des problèmes importants de la géométrie algébrique et du calcul formel. La simulation de plusieurs phénomènes physiques, chimiques, optimisation, interpolation, robots et de problèmes géométriques conduit à des systèmes d'équations polynomiales à paramètres. La théorie d'élimination permet de résoudre des systèmes polynomiaux (voir les travaux de Macauley, Van-der-Waerden et Lazard). Les bases de Gröbner inventé par Bruno Buchberger en 1965 réduit un système polynomial à un autre système équivalent dont sa résolution se fait par un calcul des racines des polynômes univariés. La complexité du calcul des bases de Gröbner est simplement exponentielle en le nombre n des variables si le système est zéro-dimensionnel (voir les travaux de Lakshman, Lazard en 1991 et Hashemi en 2005), doublement exponentielle en n si le système est de dimension positive (Giusti, Mora et Möller).

Le premier algorithme de complexité sous-exponentielle en la taille de l'entrée pour la décomposition d'une variété algébrique en composantes irréductibles a été établi par Chistov et Grigoriev en 1982. Leur algorithme est basé sur un algorithme polynomial de factorisation des polynômes à plusieurs variables et représente chaque composante irréductible par un point générique et une famille de polynômes qui la définit. Un point générique donne les coordonnés des éléments de cette composante comme de fonctions rationnelles des racines d'un certain polynôme univarié, cette représentation est baptisée le nom RUR (Représentation univariée rationnelle) par Rouillier en 1998. D'autres mathématiciens ont fait des algorithmes avec la même borne de complexité, notamment Giusti, Heintz (1990, 1992), Elkadi, Mourrain (1998), Jeronimo, Sabia (2002) et Lecerf (2002, 2003).

Dans le cas paramétrique, Giusti et Schost (2001, 2002) ont décrit une résolution géométrique des systèmes paramétrés. Weispfenning (1991) décompose l'espace des paramètres en un nombre fini d'ensembles constructibles, chacun d'eux avec une base de Gröbner paramétrique à coefficients rationnels en les paramètres, par contre il n'y a pas une analyse de la complexité de son algorithme (voir [31, 32]). En 2000, Grigoriev et Vorobjov [19] ont trouvé une telle partition de l'espace des paramètres avec une complexité simplement exponentielle en n et r dans le cas zéro-dimensionnel. D'autres méthodes basées sur le calcul d'ensembles triangulaires et caractéristiques paramétriques (Gao [15], Wang [30], Dahan et Schost [12]) et sur les variétés discriminantes (Lazard et Rouillier [23]).

Dans ma thèse, j'ai présenté trois algorithmes qui sont cités ici en bref dans les trois sous-sections suivantes (voir mon rapport de thèse [5, 4] pour plus de détails) :

4.2 Systèmes algébriques paramétrés zéro-dimensionnels

Le premier algorithme résout des systèmes polynomiaux homogènes paramétrés zéro-dimensionnels avec un temps simplement exponentiel en le nombre n des inconnues, cet algorithme décompose l'espace des paramètres en un nombre fini d'ensembles constructibles et calcule le nombre fini des solutions par des représentations rationnelles paramétriques uniformes sur chaque ensemble constructible. En plus, le nombre des solutions ainsi que le vecteur des multiplicités sont constants sur chaque ensemble constructible de cette partition de l'espace des paramètres (i.e., indépendants des valeurs des paramètres dans cet ensemble). L'outil de base de cet algorithme est le calcul de ce qu'on appelle un U-résultant paramétrique du système $f_1 = \cdots = f_k = 0$ où $U = (U_0, \ldots, U_n)$ sont de nouvelles variables algébriquement indépendantes sur le corps $F(u_1, \ldots, u_r, X_0, \ldots, X_n)$. Chaque U-résultant paramétrique R est un polynôme de $F[u_1, \ldots, u_r, U_0, \ldots, U_n]$ homogène en $U_0, \ldots, U_n$, associé à un ensemble constructible W tel que pour toute valeur des paramètres dans W, la spécialisation de R est un U-résultant du système spécialisé. Lorsqu'un U-résultant paramétrique (W, R) est calculé, on réduit R à des polynômes univariés paramétrés (par des spécialisations convenables des variables $U_0, \ldots, U_n$) qui nous permettent par un calcul d'un P.G.C.D. paramétrique de trouver le vecteur des multiplicités des solutions du système d'une manière uniforme sur les valeurs des paramètres dans W. Ensuite l'algorithme décrit les solutions du système par ce qu'on appelle le "Shape lemma paramétrique".

4.3 Factorisation absolue des polynômes paramétrés à plusieurs variables

Le deuxième algorithme factorise absolument des polynômes paramétrés à plusieurs variables. La factorisation des polynômes est l'un des problèmes fondamentaux en algèbre et en calcul formel, ce problème remonte à Newton, Gauss, Fermat, Kronecker, Hensel et d'autres. Le premier algorithme de complexité polynomiale qui factorise des polynômes univariés à coefficients rationnels est réalisé en 1982 par Lenstra, Lenstra et Lovasz (LLL), celui-ci est basé sur le calcul d'un vecteur minimal des réseaux. En se basant sur cet algorithme et sur le lemme de Hensel, Grigoriev a fait en collaboration avec Chistov un algorithme polynomial de factorisation des polynômes à plusieurs variables à coefficients dans un corps qui est une extension primitive d'une extension purement transcendante de son corps premier (1982). Dans la même année, Kaltofen a décrit un algorithme polynomial déterministe de réduction de la factorisation des polynômes à plusieurs variables à celle des polynômes univariés. Berlekamp (1967) a établit un algorithme de factorisation des polynômes univariés à coefficients dans un corps fini. D'autres mathématiciens ont travaillé sur ce problème, citons par exemple Zassenhaus, Niederreiter, Von zur Gathen, Lenstra, Shoup, Lecerf.

Etant donné un polynôme paramétré $f \in F[u_1, \ldots, u_r, X_0, \ldots, X_n]$, l'algorithme partage l'espace des paramètres en un nombre fini d'ensembles constructibles. Sur chacun d'eux le nombre des facteurs absolument irréductibles ainsi que leur vecteur des multiplicités sont constants. Ces facteurs sont des polynômes paramétrés (à coefficients rationnels

en les paramètres) calculés par l'algorithme. En plus, l'algorithme calcule l'extension primitive de F qui contient tous les coefficients de ces facteurs. Le nombre d'éléments de cette partition ainsi que la complexité de l'algorithme sont exponentiels en n, r et d où d est une borne supérieure de degré de f par rapport à X. Cet algorithme repose sur une version paramétrisée du lemme de Hensel et sur un algorithme d'élimination de quantificateurs dans la théorie des corps algébriquement clos pour réduire le problème de la représentation des facteurs de f à celui de la représentation des solutions d'un certain système polynomial paramétré zéro-dimensionnel. Cet algorithme était le sujet d'une publication réalisée en 2004 [6].

Exemple 4.2 *Considérons le polynôme paramétré suivant :*

$$f = (u^2 + v)x^2 + uxy + vx + uy + v$$

x et y sont deux variables, u et v sont deux paramètres. L'algorithme décompose l'espace des paramètres sous la forme :

$$\mathbb{C}^2 = U_1 \sqcup U_2 \sqcup U_3$$

où $U_1 = \{u^2 + v = 0\}$. Pour tout $(a, b) \in U_1$ on obtient la factorisation absolue :

$$f^{(a,b)} = (x + 1)(ay + b).$$

$U_2 = \{u^2 + v \neq 0, u \neq 0\}$. Pour tout $(a, b) \in U_2$, $f^{(a,b)}$ est absolument irréductible. $U_3 = \{u = 0, v \neq 0\}$. Pour tout $(a, b) \in U_3$, il existe c racine cubique primitive de l'unité (dans ce cas $\chi = C^3 - 1$) qui vérifie

$$f^{(a,b)} = b(x - c)(x - c^2).$$

4.4 Systèmes algébriques paramétrés de dimensions positives

Le troisième algorithme décompose les variétés algébriques définies par des systèmes algébriques paramétrés de dimensions positives en composantes absolument irréductibles d'une manière uniforme sur les valeurs des paramètres. Plus précisement, l'algorithme partage l'espace des paramètres en un nombre fini d'ensembles constructibles W. Pour chaque W, le nombre des composantes absolument irréductibles est constant, chacune d'elles est donnée par un système représentatif paramétrique (i.e., une famille finie de polynômes paramétés tel que pour tout $a \in W$, les spécialisations de ces polynômes en a définissent la composante) et par un point générique efficace paramétrique (i.e., une représentation univariée rationnelle paramétrique des éléments de la composante). Le nombre des éléments de cette partition ainsi que la complexité totale de l'algorithme sont doublement exponentiels en n et simplement exponentiels en r et d. (voir [4, 5] pour les détails). Cette partition se fait par induction sur la codimension des composantes absolument irréductibles.

5 Présentation analytique de mes travaux de recherche après la thèse

Après la thèse, j'ai continué à travailler avec mon directeur de thèse Dimitry Grigoryev en tant qu'ATER à l'université de Rennes 1 (2006-2007). Tout en restant dans mon sujet de thèse, je me suis intéressé tout d'abord à la résolution des systèmes algébriques paramétrés de dimensions positives par les techniques des bases de Gröbner paramétriques. Ensuite aussi en restant toujours dans le domaine de calcul formel et complexité, j'ai étudié la complexité de la factorisation des opérateurs différentiels linéaires ordinaires (c'est l'analogue différentiel du problème de factorisation des polynômes multivariés étudié ci-dessus).

5.1 Bases de Gröbner paramétriques

Pour un système polynomial paramétré $f_1, \ldots, f_k \in F[u_1, \ldots, u_r][X_1, \ldots, X_n]$ (avec les mêmes notations ci-dessus), on cherche à construire une partition de l'espace des paramètres $\overline{F}^r$ en un nombre fini d'ensembles constructibles tel que pour chaque ensemble constructible W parmi eux, on calcule de polynômes $G_1, \ldots, G_s \in F(u_1, \ldots, u_r)[X_1, \ldots, X_n]$ vérifiant les propriétés suivantes :

- Les coefficients de $G_1, \ldots, G_s$ dans $F(u_1, \ldots, u_r)$ sont bien définis sur W.

- Pour tout $a \in W$, la famille $G_1^{(a)}, \ldots, G_s^{(a)} \in \overline{F}[X_1, \ldots, X_n]$ est une base de Gröbner réduite de l'idéal engendré par $f_1^{(a)}, \ldots, f_k^{(a)}$ dans $\overline{F}[X_1, \ldots, X_n]$ en respectant un certain ordre monomial fixé sur $X_1, \ldots, X_n$.

La famille $(G_1, \ldots, G_s)$ est appelée une base de Gröbner paramétrique (ou base de Gröbner compréhensive [31, 32]). La complexité du calcul des bases de Gröbner paramétriques est simplement exponentielle en n pour des idéaux zéro-dimensionnels [19, 24] et elle n'est pas étudiée dans [31, 32] pour des idéaux de dimensions positives. Il est bien connu que cette complexité est doublement exponentielle en n dans le cas non paramétrique pour des idéaux de dimensions positives et il n'y a pas de bornes de complexité dans le cas paramétrique.

5.2 D-modules

Dans le domaine de D-modules, je me suis intéressé à la factorisation des opérateurs différentiels linéaires ordinaires, une question liée d'une certaine manière à son analogue polynomial que j'ai largement étudié dans ma thèse. La factorisation constitut une étape importante dans la recherche des solutions des équations différentielles linéaires ordinaires. Elle permet de réduire ces équations à des équations d'ordres plus petits.

En 1990, Dmitry Grigoryev [20] a décrit un algorithme qui factorise des opérateurs

différentiels linéaires ordinaires avec une complexité triplement exponentielle en l'ordre n de l'opérateur à factoriser. Plus précisement, pour un opérateur différentiel linéaire ordinaire

$$L = \sum_{0 \leq j \leq n} \Big(\frac{f_j}{f}\Big) \frac{d^j}{d^j X} \in \mathbb{Q}(X)\Big[\frac{d}{dX}\Big]$$

où $f_j, f \in \mathbb{Q}[X]$ de degrés bornés par D. Soient M une borne supérieure de la taille binaire de L et N une borne supérieure sur les degrés des coefficients de tous les facteurs de L, alors

$$N \leq e^{(MD2^n)^{2^n}}$$

En plus, cet algorithme factorise L avec une complexité $(NM)^{O(n^4)}$. On voudrait améliorer cette borne de complexité en la rendant doublement exponentielle en n. Pour celà, j'ai commencé à montrer que le calcul des solutions de l'équation différentille $L(y) = 0$ sous forme de séries de Puiseux peut se faire dans un temps polynomial en n en construisant un arbre dont ses feuilles représentent les solutions (voir [22, 11]). La construction de cet arbre est basée sur les polygones de Newton différentiels qui sont bien détaillés dans mes papiers [7, 8]. Notons que la meilleure complexité connue à ce jour pour ce calcul est simplement exponentiel en n [20].

Dans [7], j'ai décrit un algorithme qui calcule les series de Puiseux solutions des équations différentielles polynomiales ordinaires avec une complexité binaire simplement exponentielle en le nombre de termes des series. Une telle borne de complexité binaire a été réalisée dans [20] mais juste pour l'équation différentielle polynomiale de *Riccatti* associée à L.

Dans [8], j'ai décrit la théorie des équations de *Riccatti* associées à des opérateurs différentiels linéaires ordinaires (les polygones de Newton différentiels de toutes leurs dérivées).

Le domaine de D-modules est un domaine large contenant plusieurs questions ouvertes, comme par exemple l'inégalité de Bézout différentielle (voir [21] pour avoir une idée de la complexité de ce problème). A noter que c'est l'analogue différentielle de l'inégalité de Bézout pour les idéaux polynomiaux que j'ai étudié et utilisé dans ma thèse.

5.3 Transformation de diffusion (Scattering transform)

Pendant mon année d'ATER à l'université de Rennes 2 (2007-2008), j'ai étudié les applications de mes travaux de recherche (calcul symbolique) sur la résolution des équations différentielles aux dérivées partielles non-linéaires par la théorie de la transformation de diffusion inverse (calcul numérique). Pour cela, j'ai travaillé avec M. Qinghua Zhang, directeur de recherche à l'INRIA (équipe SISYPHE de l'IRISA de Rennes) sur la représentation des signaux par un petit nombre de paramètres en se basant sur les

techniques de la théorie de diffusion (Scattering theory).

Cette transformation est analogue à celles de Fourier et Laplace qui permettent de calculer de solutions analytiques des équations différentielles linéaires. Elle a été introduite dans la litérature pour la première fois par Gardner et ses partenaires dans leurs papiers des années 70 (voir [16]). Dans [16], les auteurs ont étudié l'équation de Korteweg et de Vries (KdV) :

$$u_t + 6uu_x + u_{xxx} = 0$$

qui est une équation d'évolution non-linéaire décrivant le mouvement d'une vague le long d'un canal. Leur méthode est divisée en deux parties : La transformation de diffusion directe (TDD) et inverse (TDI). La TDD transforme le problème de résolution du KdV à celui de la recherche des valeurs et des vecteurs propres de l'équation de Schrödinger linéaire $L\psi = \lambda\psi$ où

$$L = -\frac{d^2}{dx^2} + u$$

est l'opérateur de Schrödinger linéaire. La TDI reconstruit une solution du KdV (i.e., le potentiel u) à partir des données de diffusion, i.e., les valeurs et les vecteurs propres. Parmi les solutions du KdV, on trouve les solitons [13], des ondes solitaires qui se propagent sans se déformer dans un milieu non-linéaire et dispersif.

Cette technique permet aussi de résoudre d'autres équations d'évolution non-linéaires comme la KdV modifiée, l'équation de Schrödinger non-linéaire, l'équation de Sine-Gordon, etc ..... (voir [13]).

5.4 Représentation des signaux

La représentation des signaux ou des données numériques avec de modèles mathématiques est un problème classique. Un tel modèle est caractérisé par un nombre de paramètres qui représentent le signal modélisé. Les modèles mathématiques les plus connus sont l'interpolation polynomial et les series de Fourier.

Plus précisement, soit f un signal, i.e., une fonction réelle d'une variable réelle x (x représente le temps). Une représentation linéaire du signal f est une écriture de f sous la forme :

$$f(x) = \sum_{n \in S} a_n f_n(x)$$

où $f_n : \mathbb{R} \longrightarrow \mathbb{R}$ sont des fonctions prises d'une base de fonctions, les paramètres linéaires a_n sont de réels et S est un ensemble fini d'indices. Une représentation non-linéaire de f est une écriture de f sous la forme :

$$f(x) = \sum_{n \in S} a_n g_n(x, b_n)$$

où g_n sont de fonctions non-linéaires en les vecteurs réels b_n des paramètres.

La représentation des signaux permet entre autres d'analyser, filtrer, extraire et de compresser un signal donné. On s'intéresse à établir des représentations non-linéaires d'un signal f par les techniques de la transformation de diffusion (TDD et TDI). On considère l'équation de Schrödinger linéaire $L = -\frac{d^2}{dx^2} + f$ associée à f et on applique les procédés TDD et TDI pour reconstruire le potentiel f à partir des données de diffusion du problème propre. Ceci permet de représenter f par de fonctions g_n et un petit nombre de paramètres a_n et b_n. Dans ce cas les g_n sont de fonctions rationnelles en les fonctions exponentielles.

Références

[1] A. Ayad and Claude Marché, *Behavioral Properties of Floating-Point Programs*. Soumis. Voir http://ali.ayad.free.fr/main.pdf.

[2] A. Ayad, *On formal methods for certifying floating-point C programs*. Rapport de recherche 2009, Projet CerPAN, ProVal, INRIA. Voir http://ali.ayad.free.fr/floats.pdf.

[3] A. Ayad, *Vérification déductive de programmes flottants dans Frama-C*. Dans MajecSTIC2009, MAnifestation des JEunes Chercheurs en Sciences et Technologies de l'Information et de la Communication, Avignon, France, 16-18 Novembre 2009. Voir http://ali.ayad.free.fr/avignon_09.pdf.

[4] A. Ayad, *Complexity of solving parametric polynomial systems*, soumis.

[5] A. Ayad, *Complexité de la résolution des systèmes algébriques paramétrés*, Thèse doctorat, Université de Rennes 1, France, Octobre 2006. Voir http://ali.ayad.free.fr/Thse_Ayad.pdf.

[6] A. Ayad, *Complexity bound of absolute factoring of parametric polynomials*, Zap. Nauchn. Sem. S.-Peterburg. Otdel. Mat. Inst. Steklov. (POMI) 316, Teor. Slozhn. Vychisl. 9, 224 (2004), p. 5-29. Translation in J. Math. Sci. (N. Y.) 134 (2006), no. 5, 2325-2339. Voir http://ali.ayad.free.fr/ayad_04.pdf.

[7] A. Ayad, *On the complexity of solving ordinary differential equations in terms of Puiseux series*, Prépublication IRMAR, Rennes, France, Mai 2007. Voir http://arxiv.org/abs/0705.2127

[8] A. Ayad, *A note on the computation of Puiseux series solutions of the Riccatti equation associated with a homogeneous linear ordinary differential equation*. Publication 2008. Voir http://arxiv.org/abs/0802.2625.

[9] P. Baudin, J-C Filliâtre, C. Marché, B. Monate, Y. Moy and V. Prevosto. *ACSL : ANSI/ISO C Specification Language*, manual reference, 2008, http://frama-c.cea.fr/acsl.html.

[10] S. Boldo and J-C Filliâtre, *Formal Verification of Floating-Point Programs*, 18th IEEE International Symposium on Computer Arithmetic, ARITH 2007, Montpellier, France, p. 187-194.

[11] J. Cano, *The Newton Polygon Method for Differential Equations*, Computer Algebra and Geometric Algebra with Applications, 2005, p. 18-30.

[12] X. Dahan, E. Schost, *Sharp estimates for triangular sets*, Proceedings ISSAC 2004.

[13] P. G. Drazin and R.S. Johnson, *Solitons : an introduction*, Cambridge University Press, second edition, 1989.

[14] Frama-C, a verification tool for C programs. http://www.frama-c.cea.fr/.

[15] X-S. Gao, S-C. Chou, *Solving parametric algebraic systems*, ISSAC 1992, California USA, p. 335 - 341.

[16] C. S. Gardner, J. M. Greene, M. D. Kruskal, and R. M. Miura, *Korteweg-deVries equation and generalizations. VI. Methods for exact solution*, Communications on Pure and Applied Mathematics, volume 27, (1974), p. 97-133.

[17] M.-J. Gonzalez-Lopez, T. Recio, *The ROMIN inverse geometric model and the dynamic evaluation method*, In "Computer Algebra in Industry, Problem Solving in Practice", Edited by Arjeh M. Cohen, Wiley, 1991, p. 117- 141.

[18] D. Grigoriev, *Factorization of polynomials over a finite field and the solution of systems of algebraic equations*, J. Sov. Math., 34(1986), No.4 p. 1762-1803.

[19] D. Grigoriev, N. Vorobjov, *Bounds on numbers of vectors of multiplicities for polynomials which are easy to compute*, Proc. ACM Intern. Conf. Symb and Algebraic Computations, Scotland, 2000, p. 137-145.

[20] D. Grigoriev, *Complexity of factoring and GCD calculating of ordinary linear differential operators*, J. Symp. Comput., 1990, vol.10, N 1, p. 7-37.

[21] D. Grigoriev, *Weak Bezout inequality for D-modules*, J. Complexity, 2005, vol. 21, p. 532–542.

[22] D. Grigoriev, M. Singer, *Solving ordinary differential equations in terms of series with real exponents*, Trans. AMS, 1991, vol. 327, N 1, p. 329-351.

[23] D. Lazard, F. Rouillier, *Solving parametric polynomial systems*, Rapport de recherche, INRIA, Projet SALSA, Octobre 2004.

[24] A. Montes, *A new algorithm for discussing Gröbner basis with parameters*, J. of Symb. Comp., 33, 2002, p. 183-208.

[25] Y. Moy, *Automatic Modular Static Safety Checking for C Programs*, PhD thesis, University Paris XI, 2009.

[26] Y. Moy, *Jessie Plugin Tutorial*, 2008, http://frama-c.cea.fr/jessie.html.

[27] E. Rodríguez-Carbonell and D. Kapur. *Automatic generation of polynomial loop invariants : Algebraic foundations*. In ACM ISSAC'04, volume 505040, p. 266273. ACM Press, 2004. http://www.cs.unm.edu/~kapur/myabstracts/issac04enric.html.

[28] E. Schost, *Computing Parametric Geometric Resolutions*, Applicable Algebra in Engineering, Communication and Computing 13(5), 2003, p. 349 - 393.

[29] E. Schost, *Sur la résolution des systèmes polynomiaux à paramètres*, Thèse de doctorat, École polytechnique, décembre 2000.

[30] D. Wang, *Elimination Practice Software Tools and Applications*, World Scientific Pub Co Inc, 2004.

[31] V. Weispfenning, *Comprehensive Gröbner bases*, J. Symbolic Computation, 14 (1991), p. 1-29.

[32] V. Weispfenning, *Solving parametric polynomial equations and inequalities by symbolic algorithms*, MIP-9504, Universitat Passau, Januar 1995, in Proc. of the workshop "Computer Algebra in Science and Engineering", Bielefed, August 1994, World Scientific, 1995, p. 163-179.

[33] J-C. Filliâtre, *The Why verification tool*, 2002, http://why.lri.fr/.

[34] V. E. Zakharov and A. B. Shabat, *Exact theory of two dimensional self-focusing and one dimensional self-modulation of waves in non linear media*, Sov. Phys. 1972, J. E. T. P. 34, p. 62-69.