

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université M'hamed BOUGARA de BOUMERDES



Faculté des Sciences
Département d'Informatique

MEMOIRE DE MAGISTER

Spécialité : Systèmes informatiques et génie des logiciels

Option : Spécification de Logiciels et Traitement de l'Information

Ecole Doctorale

Présenté par :

LOUNAS Razika

Thème

**Preuve en Coq de propriétés de programmes numériques
partant du code en C**

Devant le jury de soutenance composé de:

Mr. Mohammed AHMED NACER	Professeur	USTHB	Président
Mr. Mohammed MEZGHICHE	Professeur	UMBB	Rapporteur
Mme. Thouraya TEBIBEL	Maître de conférences	ESI	Examinateur
Mr. Ahmed AIT BOUZIAD	Maître de conférences	UMBB	Examinateur

Année Universitaire : 2008/2009

Résumé

L'utilisation des programmes informatiques dans des applications critiques nécessite l'utilisation des méthodes formelles basées sur la rigueur mathématique pour établir leur correction conformément à leurs spécifications.

La méthode formelle Why permet de générer, à partir d'un programme C spécifié avec Caduceus, un ensemble d'obligations de preuves qu'il faut prouver à l'aide d'un assistant de preuve pour établir la correction du programme.

Le calcul matriciel est intensivement utilisé dans les programmes scientifiques. Ceci a engendré le développement de plusieurs bibliothèques dont BLAS (Basic Linear Algebra Subroutines), pour permettre une écriture rapide et efficace des programmes de calcul matriciel.

Dans notre travail, nous avons utilisé la méthode Why pour prouver deux programmes issus des BLAS : le produit matriciel et la résolution des systèmes.

Nous avons utilisé l'assistant de preuve Coq pour décharger les obligations de preuves. Pour mener les preuves, nous avons proposé une nouvelle définition du type matrice qui peut être utilisé pour prouver d'autres programmes.

Mots clés : Preuves des programmes, l'outil Caduceus, la méthode Why, l'assistant de preuve Coq, la bibliothèque BLAS.

Table des Matières

Introduction générale	05
Chapitre 1 - Introduction aux preuves des programmes	07
1. Introduction	08
2. Les méthodes formelles	08
3. La spécification formelle	09
3.1. Les paradigmes de spécification	10
3.1.1 Le paradigme fonctionnel	10
3.1.2 Spécification à base d'états	10
3.1.3 Le paradigme algébrique	10
3.1.4 Le paradigme état- transition	10
3.1.5 Le paradigme logique	10
4. Les preuves des programmes	11
4.1. La synthèse des programmes	11
4.1.1 L'extraction	11
4.1.2 Le raffinement	12
4.2. La vérification des programmes	13
4.2.1 L'annotation des programmes	13
4.2.2 La génération de spécification	14
5. Les systèmes de preuves	14
5.1. Définition	14
5.2. Critères de classification	15
5.2.1 Critère de De Bruijn	15
5.2.2 Logique traitée	15
5.2.3 Automatisation/expressivité	15
5.2.4 Style d'interaction	15
Chapitre 2 – Preuves des programmes impératifs et des programmes numériques	17
1. Preuves des programmes impératifs	18
1.1. La logique de Hoare	18
1.1.1 Correction d'un triplet de Hoare	19
1.1.2 Les règles d'inférence	19
1.1.2.1. L'instruction Skip	19
1.1.2.2. L'affectation	19
1.1.2.3. La composition séquentielle	19
1.1.2.4. La conditionnelle	19
1.1.2.5. La boucle	20
1.1.2.6. Les règles logiques	21
1.1.3 Quelques travaux sur la logique de Hoare	22
1.2. La logique de Dijkstra	23
1.2.1 Définitions	23
1.2.2 Propriétés du calcul WP	23
1.2.3 Règles du calcul WP	24
1.2.3.1. L'instruction Skip	24
1.2.3.2. L'affectation	24
1.2.3.3. La composition séquentielle	24
1.2.3.4. La conditionnelle	24
1.2.3.5. La boucle	24

1.2.4 Utilisation du calcul WP pour les preuves des programmes.....	25
1.3. Les techniques de plongement.....	26
1.3.1 Formalisation par plongement superficiel.....	26
1.3.2 Formalisation par plongement profond.....	26
1.4. Les techniques de raisonnement sur la mémoire.....	26
1.4.1 Le modèle concret.....	26
1.4.2 Le modèle de Burstall-Bornat.....	27
1.4.3 Logique de fragmentation.....	27
2. Preuves des programmes de calcul numérique.....	28
2.1. Formalisation de l'arithmétique des ordinateurs.....	28
2.1.1 Arithmétique flottante IEEE-754.....	28
2.1.2 Formalisation de l'arithmétique flottante.....	29
2.2. Formalisation de l'arithmétique d'intervalles.....	31
2.3. Formalisation des nombres réels.....	32
2.3.1 Les erreurs numériques.....	32
2.3.2 Formalisation des nombres réels dans les systèmes de preuves.....	33
2.3.2.1. Construction versus axiomatisation.....	33
2.3.2.2. Formalisme intuitionniste versus formalisme classique.....	34
2.3.3 Quelques exemples de formalisation des nombres réels.....	35
Chapitre 3 – L'assistant de preuves Coq.....	36
1. Introduction.....	37
1.1. Le lambda-calcul.....	37
1.2. La logique constructive.....	37
1.3. La sémantique de Heyting et Kolmogorov.....	38
2. Le système Coq.....	38
3. Le langage de spécification : Gallina.....	38
3.1. Les types inductifs.....	39
3.1.1 Les types inductifs sans récursivité.....	39
3.1.2 Les types inductifs avec récursivité.....	39
3.1.3 Vecteurs et matrices.....	40
3.2. Les fonctions récursives.....	41
3.2.1 Les fonctions structurellement récursives.....	41
3.2.2 Définir des fonctions par récurrence bien fondée.....	41
3.3. Les prédicats inductifs.....	42
4. Manipulation des preuves.....	43
4.1. La tactique.....	43
4.2. Quelques tactiques de bases.....	43
4.3. Les tactiques numériques.....	47
5. Formalisation des réels.....	49
6. Preuves des programmes dans Coq.....	49
Chapitre 4 – Les outils Caduceus et Why.....	50
1. Introduction.....	51
2. L'outil Caduceus.....	51
2.1. Annotations et spécifications.....	52
2.2. La traduction Caduceus.....	53
2.2.1 Le modèle mémoire dans Caduceus.....	53
2.2.1.1. Le type « pointer ».....	54
2.2.1.2. Les états mémoire.....	54
2.2.1.3. La théorie associée au modèle.....	55
2.2.2 Le calcul d'effets dans Caduceus.....	55

2.2.3 La traduction du code C dans Caduceus	56
3. L'outil Why	56
3.1. Introduction	56
3.2. Le langage Why	57
3.3. La méthode Why	58
3.3.1 La traduction fonctionnelle	58
3.3.2 Méthodologie de preuves	60
3.3.3 Calcul de la plus faible précondition	61
4. Exemple d'utilisation	63
Chapitre 5 – Contribution. Application de la méthode Why à la certification	
De programmes d'algèbre linéaire	65
1. Introduction	66
1.1. Les bibliothèques numériques	66
1.2. La bibliothèque BLAS.....	66
1.2.1 Le produit matriciel.....	67
1.2.2 La résolution des systèmes.....	68
2. Preuve de correction de DGEMM_PLUS et DTRSM.....	70
2.1. Description du problème	70
2.2. Annotation et extraction des obligations de preuves des programmes DGEMM_PLUS et DTRSM.....	71
2.2.1 Annotation du programme DGEMM_PLUS	71
2.2.2 Annotation du programme DTRSM	74
2.2.3 Application de Caduceus et Why.....	76
2.2.4 Définition des objets logiques dans Coq	76
2.2.5 Les obligations de preuves	78
2.3. Preuve dans Coq de la relation $DGEMM_PLUS(A, DTRSM(A, X), 0) = X$	79
2.3.1 Présentation du type matrice.....	79
2.3.1.1. Définition	79
2.3.1.2. Fonctions d'accès et de modification	79
2.3.1.3. Passage vers le type pointer	80
2.3.2 Définition de la fonction <code>dgemm_plus</code> dans Coq	81
2.3.3 Démonstration que <code>dgemm_plus</code> réalise DGEMM_PLUS	82
2.3.4 Démonstration de la propriété $DGEMM_PLUS(A, DTRSM(A, X), 0) = X$	83
Conclusion et perspectives	87
Annexes	90
Annexe A Le fichier <code>dgemm_spec_why.v</code>	91
Annexe B Le fichier <code>dtrsm_spec_why.v</code>	95
Bibliographie	98