



N° d'ordre :

UNIVERSITE DE M'SILA
FACULTE DES MATHÉMATIQUES ET DE L'INFORMATIQUE
Département d'Informatique

MEMOIRE

Présenté pour l'obtention du diplôme de Master

Domaine : Mathématiques et Informatique

Filière : Informatique

Spécialité : Système d'information Avancés

Par :

BOUKHALAT Youssef

SUJET

**Un Translateur C vers Promela
pour la Vérification formelle des programmes C
avec Spin Model-checker**

Soutenu publiquement le : 28/06/2012 devant le jury composé de :

Mr. MEHANNI Tahar

Université de M'sila

Président

Mr. BOUHOUTTA GUERMECH Salah Eddine

Université de M'sila

Rapporteur

Mr. DEBBI Imad

Université de M'sila

Examineur

Promotion : 2011 /2012

Table des matières

Introduction générale

I	Introduction	I
2	Objectif	2
3	Motivation	2
4	Les travaux existants	2
5	Structure du mémoire.....	3

Chapitre I : Vérification & Model Cheking

I.1	Introduction	4
I.2	La vérification software	4
I.2.1	La Vérification	5
I.2.1.1	L'examen	5
I.2.1.2	Le test	5
I.2.2	Détecter des erreurs logicielles	6
I.3	Vérification formelle	7
I.4	Vérification à base de modèles	7
I.5	Le Model Cheking	9
I.5.1	L'approche du Model Cheking	9
I.5.2	Les Caractéristique du Model Checking	10
I.5.2.1	Le processus du Model Checking	10
I.5.2.1.1	Phase de modélisation	10
I.5.2.1.2	Phase d'exécution	10
I.5.2.1.3	Phase d'analyse	10
I.5.3	La Modélisation	11
I.5.3.1	Les propriétés pertinentes	11
I.5.3.2	La Validation	12
I.5.4	L'exécution du Modèle-checker	12
I.5.5	L'analyse	12
I.5.6	Organisation de vérification	14
I.5.7	Le succès du Model Checking	14
I.5.8	Les points forts et faibles	14
I.5.8.1	Les points forts du Model Checking	14
I.5.8.2	Les faiblesses du Model Checking.....	15
I.5.9	Outils de Model Checking	17
I.5.10	SPIN Model Checker	18
I.6	Conclusion.....	18

Chapitre 2 : La Vérification formelle des programmes

2.1	La vérification des programmes.....	19
2.2	Le principe de vérification des programmes	20
2.3	La Traduction dirigée par la syntaxe (syntax directed translation)	21
2.3.1	Le principe de traduction dirigé par la syntaxe	22
2.3.1.1	Définition de la syntaxe d'un langage	22
2.3.1.2	Les composants d'une grammaire à libre contexte	22
2.3.1.3	L'arbre d'analyse	23
2.3.1.4	L'Action sémantique.	23
2.3.1.5	Table des symboles	24
2.3.1.6	Le principe	24
2.4	Conclusion	25

Chapitre 3 : Le principe de Translation du C vers Promela

3.1	Introduction	26
3.2	Le Langage C	26
3.2.1	Introduction	26
3.2.2	Sous-C proposé	27
3.2.2.1	Mots Clés	28
3.2.2.2	Types de données	28
3.2.2.3	Opérateurs et expressions	29
3.2.2.4	Contrôles d'exécution	30
3.2.2.5	Les Fonctions	30
3.3	Promela	31
3.3.1	Mots Clés	31
3.3.2	Types de données	31
3.3.3	Opérateurs et expressions	33
3.3.4	Contrôles d'exécution	33
3.3.5	Processus	34
3.4	Le principe de la translation C vers Promela	35
3.4.1	Les chemins grammaticaux	36
3.4.2	La sélection	36
3.5	La translation de C vers Promela	37
3.5.1	Les types de données.....	37
3.5.1.1	Types de données primitives	37
3.5.1.2	Les types de données composées	37
3.5.2	Expressions et opérations	39
3.5.3	Contrôles d'exécution	39
3.5.4	La notion d'atomicité	41
3.5.5	Les Fonctions	42
3.6	Conclusion	43

Chapitre 4 : Translateur C vers Promela

4.1	Le Translateur C vers Promela (CProl)	44
4.1.1	Les Outils de Développement	44
4.2	La modélisation du translateur CProI	45
	Diagramme de Classes	45
4.2.1	Description des Classes	46
4.2.2	L'interface principale	46
4.3	Exemples	48
4.4	Conclusion	51

Conclusion générale	52
---------------------------	----

Références	45
------------------	----

Annexes

Annexe A: Grammaire de Langage C	56
Annexe B: Grammaire de Promela	58
Annexe C: Exemple de traduction avec les chemins grammaticaux sélectionnés	62

1 Introduction

Ces derniers temps, nous développons de plus en plus souvent des systèmes critiques et complexes, comme les applications réparties, tels que les protocoles de communication et les systèmes distribués et toutes sortes de systèmes embarqués tels que les cartes à puce, les ordinateurs de poche, téléphones mobiles et téléviseurs haut de gamme. Ces systèmes et logiciels sont massivement empiéter sur la vie quotidienne, Ces systèmes doivent répondre aux spécifications et aux besoins des futurs utilisateurs, en plus, il faut qu'ils soient corrects. Par conséquent, la fiabilité du logiciel devient un problème critique pour le monde entier. Autrement dit, ces systèmes doivent être vérifiés et valides avant leur implémentation.

C est l'un des langages de programmation les plus populaires et est extrêmement utilisé pour développer tous les types de logiciels. Par conséquent, la sécurité des logiciels écrits en C représente une proportion importante de la fiabilité des logiciels.

Les techniques de vérification ont été développées dans un premier temps par des équipes de chercheurs et sont de plus en plus utilisées dans le milieu industriel. Elles sont utilisées pour l'analyse et la vérification d'une grande variété de systèmes (systèmes, logiciels, systèmes réactifs, systèmes temps réel). Ces techniques sont efficaces et sont fréquemment utilisées pour détecter les problèmes et les bogues dans les systèmes. De plus, de nombreuses études sont en cours pour élargir leurs champs d'utilisation et améliorer leur efficacité.

Le model checking est une des techniques de vérification qui nécessite un modèle simplifié du système. Elle permettra de tester automatiquement si ce modèle peut accueillir les spécifications du système donné ou non. Un grand nombre d'outils le model checking ont été proposés comme les CHESS, UPPAAL and SLAM.

Un des outils les plus efficaces pour la vérification automatique est le vérificateur de modèle de SPIN (SPIN model checker). Ce vérificateur de modèle [-] est un outil général pour vérifier l'exactitude des modèles de logiciels de façon rigoureuse et surtout automatisé. La langue d'entrée de SPIN est Promela.

Transformer un langage de programmation dans un langage pris en charge par un outil de vérification est un moyen largement adopté de faire la vérification formelle. Il permet la réutilisation des langues et des outils existants. Dans ce travail, nous proposons une méthode médiate de transformation correcte de C à Promela afin de faire la vérification formelle pour trouver les problèmes potentiels dans les programmes à l'aide de Spin.

2 Objectif :

Nous présentons une façon indirecte de vérifier les programmes C avec le modèle checker par traduire le code C vers Promela, qui est le langage input pour le Spin. Ce qui nous permet d'utiliser le Spin pour vérifier les programmes C. nous avons choisi un sous-ensemble du langage C.

3 Motivation

C est l'un des plus populaires langages de programmation sur la planète, et il est intensivement utilisé pour développer tout type de systèmes, allant des programmes embraqués à des applications software. Plus le C est utilisé, plus on aura besoin de trouver les méthodes appropriées pour vérifier la fiabilité des programmes C, et localiser les erreurs potentielles le plutôt possible.

Spin model-checker est l'un des model-checkers les plus populaires au monde. L'outil a été appliqué à toute sorte de vérification, allant de la vérification des logiciels de traitement des appels complexes utilisés dans les échanges téléphoniques, jusqu'à la validation des logiciels délicats de contrôle pour les missions spéciales. Cependant, il ne peut pas vérifier ces programmes directement, pour remédier à cette situation, nous avons à trouver une méthode pour utiliser ce puissant outil. Dans ce mémoire nous générons le code Promela qui représente les modèles des programmes C par la translation du langage C vers la langue Promela, alors nous pouvons analyser le modèle généré utilisant le Spin model-checker.

A cause de la haute flexibilité et réutilisabilité il était impossible de considérer tout le langage C, c'est pourquoi nous avons choisi un sous-ensemble C dans le cadre de notre projet. Le programme cible écrit en C est limité dans le sous-ensemble C contenant les types de données de bases et composés, les statements, et les fonctions.

4 Les travaux existants :

Modex

Modex est un outil développé par Gerard Holzmann. Il est écrit en ANSI-C, avec l'objectif d'extraire automatiquement des modèles de haut-niveau pour la vérification à partir des codes ANSI-C. il est destiné pour les applications incluant les logiciels de switching de la téléphonie, les systèmes d'exploitations distribués, l'implémentation des protocoles, les méthodes de contrôle concurrentielles, et les applications client-serveurs [1].

Il génère le Promela des codes C. mais il reste limité, il utilise Promela d'une façon restreinte, on peut remarquer qu'il ne manipule pas les pointeurs, et ne gère pas les appels des procédures. Qui prive le projet de précisément simuler certains codes C originaux.

JPF (Java PathFinder)

Le JPF est originellement implémenté en 1999, il a été développé comme un traducteur d'un sous-ensemble de Java vers Promela, et utilise Spin pour la vérification à base de modèles de la translation. Le principe de translation s'articule sur l'essai de traduire Java en approchant les mêmes fonctionnalités dans Promela. JPF est capable de vérifier tout programme Java qui ne dépend pas des méthodes natives non-supportées [2].

Le Translateur proposé par Ke Jiang

Ce traducteur était développé spécialement pour prendre en charge la translation des algorithmes concurrentiels écrits en C. cependant il présente des limitations, telles qu'il ne supporte pas les pointeurs, et ne résout le problème de structure de données multi-tableaux dimensionnels non supporté par le Promela [3].

5 Structure du mémoire

Pour mener à bien notre projet et atteindre les objectifs fixés, nous avons structuré notre mémoire de la façon suivante :

Une Introduction générale introduisant le domaine d'intérêt, une description du projet en question en présentant ses objectifs argumentés avec une motivation suivie par les travaux existants et enfin la structure du mémoire.

Chapitre 01 est consacré à présenter les techniques de vérification du software, commençant par les techniques traditionnelles à la vérification formelle et arrivant au model checking où le Spin model-checker est décrit.

Chapitre 02 introduit la tendance de la communauté des méthodes formelles à se focaliser sur la vérification à base de modèles des programmes écrits dans les langages de programmation modernes. Suivie d'une discussion des méthodes de translation de ces langages vers le Promela qui est la langue input du Spin l'outil de vérification à base de modèles le plus sollicité.

Chapitre 03 présente le sous-ensemble C choisi, ainsi que la grammaire C et celle aussi du Promela. Le principe de la translation proposé est explicité et détaillé.

Chapitre 04 est dédié à l'implémentation du traducteur CProm, l'environnement du développement, les outils et langages utilisés sont présentés. Certains résultats représentatifs sont exposés.

Le mémoire est conclu avec une conclusion générale et des perspectives pour des travaux futurs.

Conclusion Générale

Les méthodes formelles sont de plus en plus appelées pour la vérification de programmes écrits dans des langages de programmations modernes. Cette philosophie originaire des idées du projet JPF (Java PathFinder) lancé par la NASA en 1999 pour analyser des programmes Java. L'analyser ce fait par l'application du Spin model-checker indirectement sur les programmes Java, en passant par une étape intermédiaire de translation vers le Promela, cette phase est nécessaire étant donné que le Spin accepte les inputs dans cette langue.

Ces mêmes idées ont été reprises par extension à d'autre langage de programmation. Vu que le langage C est l'un des langages de programmations les plus populaires sollicité dans le développement des programmes dans divers domaines. Plusieurs efforts ont été investi afin de mettre en œuvre des outils de translation complets du C vers Promela, néant moins tous les attentives souffres de limitations. Ces dernières sont dues en leur globalité à la richesse et la diversité du langage C comparant avec le Promela.

La translation proposée dans notre projet, suit le principe de la translation dirigée par la grammaire, où nous avons augmenté cette dernière avec des attributs pour mieux contrôler le processus et l'arbre d'analyse abstrait. Cependant, nous avons introduit une technique de sélection des chemins grammaticaux qui nous évite d'élaborer des chemins sans intérêt, et de ce fait ne génère que ce qui est sûre. Une fois on assure que le chemin est unique, la translation est engagée immédiatement vers le Promela. Cette façon de faire optimise considérablement le temps d'exécution du translateur baptisé CProl. Il assure la translation correcte de tout le sous-ensemble C choisi.

Dans notre travail nous avons essayé de surmonter certaines des limitations déjà évoquées, à savoir : une proposition d'implémenter une structure de données composées non supportée par le Promela qui sont les tableaux multidimensionnels pour rendre la translation des tableaux multidimensionnels naturellement supportés en C, facile à réaliser.

Au niveau des appels de fonctions récursives qui ne sont pas prises en charge d'une façon complète, surtout que le mécanisme du C et Promela sont différent, puisque en Promela les fonctions sont modélisées par des processus où la valeur retournée est envoyée et reçue par des canaux. Ce mécanisme limite notre souplesse de passage de paramètre si facile en C. pour surmonter cette difficulté nous avons tenté de manipuler soigneusement le mécanisme des canaux.

Nous avons profité de l'atomicité existante en Promela pour assurer que certaines séquences du code C s'exécutent d'une façon indivisible.

Le développement du Prototype CProl a été réalisé suivant une vision orienté objet avec le C #.

Perspectives et travaux futurs

Automatisation du lancement du test qui nécessite la translation du code, gérer l'initialisation suivi du lancement du Spin.

Les limitations principales sont les pointeurs du C, plusieurs actuellement sont engagés pour y trouver une solution concluante. Cependant aucune d'entre elle n'est complète.

Nous voudrions attiré l'attention sur les bibliothèque parallèles du C, spécialement la bibliothèque Wool et OpenMP(Open Multiprocessing).qui contiennent des fonctions assurant la concurrence, une notion sensible et délicate en programmation. Les algorithmes concurrentiels sont exposés à des erreurs pouvant ne pas être détectées au niveau design. Donc, il serait très intéressant de les tester avec le Spin surtout qu'il est dédié pour la vérification u parallélisme dans les modèles.

- [1] Gerard J. Holzmann and Margaret H. Smit. "Software model checking: Extracting verification models from source code. Formal Methods for Protocol Engineering and Distributed Systems", (Conference Proceedings FORTE/PSTV99) 1999.
- [2] K Havelund, T Pressburger. "Model Checking Java Programs Using Java PathFinder", International Journal on Software Tools for Technology Transfer (2). 2000
- [3] Ke Jiang, "Model Checking C Programs by Translating C to Promela", Memoir de Master, université de UPPSALA, 2009.
- [4] Christel Baier & Joost-Pieter Katoen, "Principles of Model Checking". 2008.
- [5] Joost-Pieter Katoen, "Concepts, Algorithms, and Tools for Model Checking" 1999.
- [6] A. Maleki, "Framework for Analyzing Highly Concurrent Algorithms In SPIN" April 2011.
- [7] Messaoud RAHIM, "Vérification distribuée des systèmes de grandes tailles". Mémoire de Magistère en Informatique(INI). 2003.
- [8] Model Checking, Wikipedia:
http://fr.wikipedia.org/wiki/Model_checking consulté en juin 2012
- [9] S. Chouali, « Contribution du raffinement à la vérification de systèmes sous hypothèses d'équité » thèse de doctorat. 2003.
- [10] Girish Keshav Palshikar, "An introduction to model checking". 2004
<http://www.eetimes.com/design/embedded/4024929/An-introduction-to-model-checking>
consulté en juin 2012
- [11] MoonWalker model checking .NET applications:
<http://fmt.cs.utwente.nl/tools/moonwalker/> consulté en juin 2012
- [12] UPPAAL : <http://www.uppaal.com/> consulté en juin 2012
- [13] PRISM Model checker : <http://www.prismmodelchecker.org/> consulté en juin 2012

Bibliographie

- [14] The SPIN Homepage. <http://www.spinroot.com>.
- [15] Wim H. Hesselink, "Introduction to the Model Checker Spin". October, 2008
- [16] G.J. Holzmann, "The SPIN model checker: Primer and reference manual". 2004.
<http://my.safaribooksonline.com/book/software-engineering-and-development/software-testing/0321228626> consulté en juin 2012
- [17] W. Visser, K. Havelund, G. Bart. "Model Cheking Programs" .2002.
- [18] Maggie Johnson. "Syntax Directed Translation", 2008.
- [19] TIOBE Programming Community Index:
<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html> consulté en juin 2012
- [20] Mordechai Ben-Ari, "Principles of the Spin Model Checker". janv. 2008
http://books.google.dz/books/about/Principles_of_the_Spin_Model_Checker.html?id=eVTN8UanIGcC&redir_esc=y

Abstract

Transforming a specification from language into a language supported by a verification tool is a widely adopted way of doing formal verification. It enables the reuse of existing languages and tools.

Model-checker is an approach for formal verification. It is used in the designing phase of a system. In model checking a model is made from an existing design and that model is then used to verify the design. Spin model-checker is such a model verifier. The model that Spin verified is written in Promela.

Nowadays, the cost of program errors is increasing from day to day, so software reliability becomes a critical problem to the whole world. C is a programming language widely used for developing all types of software. Hence, the security of software written in C accounts for important proportion of software reliability.

This thesis will describe a mediate method of model checking C codes to find potential problems using Spin. The translator we developed uses a technique inspired by the technique of translation syntax directed translation techniques for doing the translation from C to Promela. C# was the language we used for programming.

Keywords : Verification, Spin model-checker, Syntax-directed translation, C language, Promela, C#

Résumé

La transformation d'une spécification du langage vers un langage pris en charge par un outil de vérification est un moyen largement adopté en vérification formelle. Il permet la réutilisation des langages et des outils existants.

Le Model-checker est une approche pour la vérification formelle. Il est utilisé dans la phase de conception d'un système. Dans model-checking un modèle est construit à partir d'une conception existante et ce modèle est ensuite utilisé pour vérifier la conception. Spin model-checker est un vérificateur de modèle. Le modèle que Spin vérifié est écrit dans Promela.

Aujourd'hui, le coût des erreurs de programme est en augmentation de jour en jour, ce qui fait que la fiabilité du logiciel devient un problème critique pour le monde entier. C est un langage de programmation largement utilisé pour développer tous types de logiciels. Par conséquent, la sécurité des logiciels écrits en C représente la proportion importante de la fiabilité des logiciels.

Ce mémoire décrit une méthode médiate de model checking codes C pour trouver les problèmes potentiels à l'aide de Spin. Le traducteur que nous avons développé utilise une technique inspiré par la technique de traduction dirigée par la syntaxe pour faire la traduction de C à Promela. C # a été le langage utilisé pour l'implémentation.

Mots-clés : Vérification, Model-checker Spin, Translation dirigée par la syntaxe, Langage C, Promela, C#

ملخص

التحويل من لغة إلى لغة أخرى معتمدة من قبل وسيلة للتحقق هو نهج أعتمد على نطاق واسع للقيام بالتحقق الشكلي، يسمح بإعادة استعمال اللغات حوالأدوات الموجودة.

نموذج التحقق هو نهج للتحقق الشكلي. يمكن استخدامه في مرحلة تصميم نظام، في نموذج التحقق يبنى النموذج من خلال تصميم منجز، حيث يستعمل هذا النموذج للتحقق. نموذج (C) هو محقق يعمل على النماذج التي مكتوبة في لغة (Promla).

في الوقت الحاضر تكلفة أخطاء البرمجيات تتزايد من يوم إلى آخر، لذلك موثوقية البرمجيات أصبحت مسألة حساسة بالنسبة للعالم، C هي لغة برمجة مستعملة على نطاق واسع لتطوير جميع أنواع البرمجيات، وبالتالي فإن أمن البرمجيات المكتوبة بلغة C تمثل نسبة كبيرة من موثوقية البرمجيات.

في هذه المذكرة نصف طريقة لنموذج التحقق لفحص نموذج برنامج C من جل الكشف على المشاكل الكامنة بالاستعانة بأداة التحقق من النماذج SPIN. المترجم الذي قمنا بإنجازه يستعمل طريقة مستوحاة من تقنية الترجمة الموجهة بالنحو بهدف الترجمة من C إلى Promla. C# هو اللغة المستعملة في تطوير البرنامج.

كلمات مفتاحية: أداة التحقق من النماذج SPIN، الترجمة الموجهة بالنحو، لغة C، Promla، C#