

1 Introduction

Ces derniers temps, nous développons de plus en plus souvent des systèmes critiques et complexes, comme les applications reparties, 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 embarqués 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 plus tôt 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 Traducteur 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 pas le problème de structure de données multi-dimensionnelles 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.