Sémantique des blocs inline assembleurs x86 dans du code C

Jérôme FERET
jerome.feret@ens.fr

Assité par
Marc CHEVALIER
marc.chevalier@ens.fr

1 Contexte scientifique

Dans certains domaines, les bugs informatiques peuvent causer de lourdes pertes humaines, économiques ou environnementales. On parle alors de logiciel critique, pour lesquels il faut obtenir de fortes garanties tant au niveau de la sûreté (le programme se comporte bien) que sécurité (le programme est protégé des intrusions extérieures). Les méthodes formelles, en particulier l'interprétation abstraite [CC77] permettent de construire des algorithmes qui prouvent ces propriétés au niveau sémantique.

L'équipe ANTIQUE a créé un analyseur de C, ASTRÉE [Cou+a ; Bla+03], spécialisé dans les programmes embarqués. Cet analyseur a plusieurs succès industriels tels que la vérification des commandes de vol des Airbus A340 et A380. ASTRÉE a déjà été étendu pour supporter l'analyse de programmes asynchrones [Cou+b ; Min14]. À l'heure actuel, ASTRÉE est capable de prouver des propriétés de sûreté dans du code industriel réel.

2 Sujet

Pour garantir le bon fonctionnement d'un système critique, il faut en certifier chaque couche. On s'intéresse souvent au logiciel final. C'est un premier pas indispensable, mais c'est incomplet. En effet, il faut s'assurer du bon fonctionnement du système d'exploitation (le cas échéant), du matériel etc.. Souvent, on est réduit à faire confiance au matériel, faute d'implémentation à certifier, mais on peut tout à fait s'intéresser à l'OS.

Le cas d'étude est l'analyse d'un OS embarqué. En particulier, on veut prouver l'isolation de la mémoire.

La première étape en ce sens est le support dans ASTRÉE de programmes C comportant des blocs de code écrit en assembleur x86.

Le C a pour but d'être portable, donc indépendant de la plateforme. L'assembleur est au contraire, éminemment dépendant du matériel. Un OS ne peut donc pas se passer

d'assembleur pour utiliser les fonctionnalités dépendantes du matérielles et qui sont hors du cadre du C. Tout n'a pas besoin d'être en assembleur. La majorité des opérations et du contrôle de flot se formule aussi bien en C et reste plus clair et moins sujet à erreurs. C'est pourquoi, l'assembleur n'est présent que sous forme de petits bouts de code épars insérés au sein du code C, dont l'abstraction est confortable.

Pour être capable de faire une analyse sur ces codes, il faut avant tout avoir une sémantique de ces langages.

La sémantique du C est décrite par sa norme, et celle de l'assembleur par la documentation des processeurs. Mais il n'existe pas de description précise des interactions entre le C et l'assembleur. C'est en effet un sujet délicat puisqu'il faut faire des hypothèses supplémentaire sur le C pour qu'il s'interface correctement. Il faut par exemple connaître (plus ou moins précisément) la position des variables dans la mémoire, ou encore, connaître la structure de la pile. Certaines de ces hypothèses existent déjà sous forme d'ABI (notamment les conventions d'appel). D'autres dépendent énormément de la compilation, et fonctionnent en particulier dans le cas d'une compilation simple (style dirigé par la syntaxe), peu optimisante. D'autres part, une description aussi précise de ces hypothèses que l'ABI n'est pas toujours nécessaire. Parfois, une abstraction de celles-ci peut suffire à garantir que le code s'exécute correctement.

Les interactions peuvent prendre différentes formes. La plus simple consiste à modifier les variables du C dans un bloc assembleur. Les interactions les plus complexes agissent sur le flot de contrôle. Par exemple, avec un retour précoce de la fonction ou, pire, en modifiant l'adresse de retour.

Le but du stage est de décrire cette sémantique, trouver les hypothèses raisonnables et automatiser cette déduction dans la mesure de ce qui apparaîtra possible.

3 Mission

Ce sujet pose un certain nombre de questions peu explorées. Ce qui signifie qu'il est possible de choisir un sujet très théorique ou au contraire plein d'implémentations subtiles ou encore un compromis entre les deux.

Les missions d'un stage équilibré entre théorie et implémentation peuvent être :

— choisir un modèle pour décrire l'état mémoire lors de l'exécution d'un programme C avec de l'assembleur inline ;
— comprendre la sémantique des blocs inline et trouver les hypothèses nécessaire de compilation au bon fonctionnement du code cible ;
— implémenter un interpréteur d'une partie (très) restreinte du C qui gère les blocs assembleurs, notamment le flot de contrôle de l'assembleur (sans hypothèses sur les adresses absolues concrètes) ;
— inférer automatiquement les règles nécessaires à l'exécution d'un programme.

Comme on est toujours dans le cadre de l'embarqué, il y a d'autres questions adjacentes qui sont tout aussi intéressantes. Par exemple, comment prouver que la pile

est convenablement vidé pour ne pas laisser de données inutiles alors que le flot de contrôle du C est détourné par de l'assembleur. Ou... les possibilités sont colossales !

4 Prérequis

Il n'est rien attendu de particulier en plus que ce qu'on fait en L3, si ce n'est de l'enthousiasme.

5 Équipe d'accueil

Le stage aura lieu dans l'équipe projet INRIA ANTIQUE qui se trouve au département d'informatique de l'ENS. L'équipe ANTIQUE, anciennement nommée ABSTRACTION est le chef de file mondial en interprétation abstraite. L'interprétation abstraite est une théorie unifiante pour l'approximation de structures mathématiques [CC77] qui a été inventé en 1977 par Patrick COUSOT (fondateur de l'équipe ABSTRACTION). L'interprétation abstraite est basée sur les idées que 1) le comportement des systèmes dynamiques peut être observé à différents niveaux d'abstraction et que 2) la comparaison entre deux niveaux d'abstraction peut être formellement définie par le truchement d'objets algébriques (comme des relations, des correspondances de GALOIS, des opérateurs de clôture, des idéaux etc.). L'interprétation abstraite peut être utilisée pour fournir un cadre à différentes méthodes d'analyses statiques ou pour dériver de nouveaux analyseurs statiques directement de la définition d'une relation d'abstraction. L'équipe ABSTRACTION, puis l'équipe ANTIQUE développe depuis 2001 l'analyseur statique ASTRÉE. ASTRÉE [Bla+03] est un analyseur statique permettant de prouver automatiquement l'absence d'erreurs d'exécution dans des logiciels embarqués. ASTRÉE est utilisé par l'industrie des domaines de l'avionique, du nucléaire, automobile et aérospatial, principalement en Europe et Asie.

Références

[Bla+03] Bruno BLANCHET et al. "A static analyzer for large safety-critical software". In : ACM SIGPLAN Notices. T. 38. 5. ACM. 2003, p. 196-207.

[CC77] Patrick COUSOT et Radhia COUSOT. "Abstract interpretation : a unified lattice model for static analysis of programs by construction or approximation of fixpoints". In : Proceedings of the 4th POPL. ACM. 1977, p. 238-252.

[Cou+a] Patrick COUSOT et al. The Astrée Static Analyzer. URL : `http://www.astree.ens.fr/`.

[Cou+b] Patrick COUSOT et al. The AstréeA Static Analyzer. URL : `http://www.astreea.ens.fr/`.

[Min14] Antoine MINÉ. "Relational thread-modular static value analysis by abstract interpretation". In : VMCAI. Springer. 2014, p. 39-58.