Spécifier des propriétés de haut niveau pour apprivoiser un programme C avec MetAcsl

Virgile Robles

CEA List, Laboratoire de Sûreté et Sécurité des Logiciels

18 décembre 2020

Méthodes formelles

Ensemble de méthodes pour obtenir des logiciels plus corrects, applicables durant tout le cycle de développement.

Verification déductive : permet de **prouver** qu'un programme est correct vis-à-vis de sa spécification sur toutes ses exécutions possibles.

Pourquoi tout le monde ne l'utilise pas ?

- la spécification doit être formalisée :
  - pas évident
  - si la spec est mal formalisée, la preuve ne vaut rien
- tout n'est pas automatique : la preuve est parfois ardue

Le cas du C

Frama-C permet d'exprimer des propriétés de programme et de les vérifier

Son language de spécification [1] est idéal pour prouver des propriétés sur des **petites unités de code**, via les *contrats*.

Un contrat spécifie les propriétés...

- qui doivent être vraies avant le bloc de code
- qui doivent être vraies à l'issue du bloc de code

Il spécifie aussi quelles parties de la mémoire peuvent changer

1. ACSL : **A**NSI/ISO **C** **S**pecification **L**anguage

Cependant

La spécification informelle d'un logiciel contient souvent des exigences **globales**, notamment les propriétés de **sécurité**.

Exemple

Le programme ne doit pas permettre à un utilisateur d'accéder aux données sensibles sans les privilèges requis.

Comment exprimer des exigences globales avec un langage local ?

Problème de...

Maintenance

Si exprimable, le même contrat est copié/collé partout

- Modification : tout doit changer
- Fastidieux, risque d'erreur

Problème de...

Traçabilité

Si plusieurs propriétés globales sont exprimées :

- Mélange des contrats
- Pas de lien entre haut et bas niveau
- Pas de recul : quelles propriétés sont prouvées ?

Problème...

D'expressivité

Certaines propriétés ne sont pas facilement exprimables avec des contrats

- Encodage parfois possible mais compliqué
- Les deux problèmes précédents sont amplifiés
- La preuve est ardue

La thèse doit amener à...

1. Identifier formellement une classe de propriétés globales intéressantes
2. Développer un moyen de les **spécifier** qui soit :
   - traçable
   - maintenable
   - expressif
3. Développer un moyen de **vérifier** ces propriétés

En pratique...

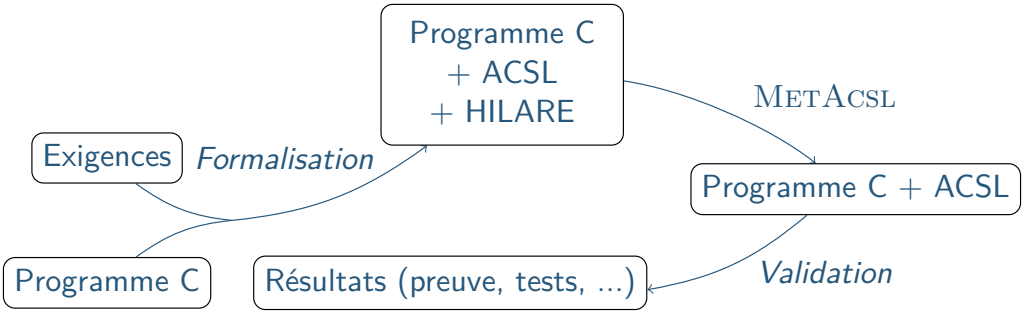
En s'appuyant sur des exemples de propriétés fournies par des partenaires (DGA) :

1. Toute propriété formulée comme un invariant ou une condition sur l'écriture/lecture de la mémoire : HILARE[2]
2. Développement d'une **extension** d'ACSL pour les écrire
3. Développement du greffon MetAcsl pour Frama-C qui :
   - lit les HILARE
   - les transforme en annotations équivalentes dans le code

   Permet d'utiliser les outils existants pour la vérification

2. **H**igh-**l**evel **A**CSL **R**equirement (anciennement méta-propriété)

Visuellement...



Application à WOOKEY

Le projet WOOKEY

Prototypage d'une clé USB sécurisée chiffrante avec authentification forte.
Complètement open source et open hardware.

En concertation avec l'ANSSI, analyse de plusieurs propriétés de sécurité sur le *bootloader* du firmware :

- il viole pas la structure d'automate spécifiée
- il ne revient jamais sur les choix qu'il fait
- il ne lit/n'écrit pas plus de mémoire que nécessaire

Conclusion

- Il pourrait être souhaitable de rendre le language HILARE de plus haut niveau...
- Une bonne partie de l'objectif initial est atteint

Rendre plus accessible la spécification/vérification d'exigences globales

MetAcsl

- le greffon est utilisé industriellement
- il est disponible publiquement [a]

a. https://git.frama-c.com/pub/meta

Un aperçu d'HILARE

```
1  meta \prop,
2    \name(confidential_read),
3    // Fonctions concernées : toutes
4    \targets(\ALL),
5    // Tout accès à la mémoire ...
6    \context(\reading),
7      \forall_sensitive(s,
8        //...qui lit une donnée sensible...
9        \read == s.data
10       //...doit se faire dans des conditions correctes !
11       ==> user_privilege >= s.needed_privilege
12     );
```