INFO M227 Analyse de programmes pour la cybersécurité

Assistant : Gonzague Yernaux (gonzague.yernaux@unamur.be)

Année 2023-2024

Travail individuel : Interprétation abstraite

Aperçu général du travail

Dans ce travail, il vous est demandé de réaliser l'étude complète, ainsi que l'implémentation, d'une analyse statique de programmes, sur base d'une description de problématique informelle. Vous pouvez choisir parmi deux énoncés : votre analyse portera soit sur les valeurs d'un langage de programmation manipulant des points du plan cartésien, soit sur une extension du langage Small prenant en compte des éléments de concurrence.

Délivrables

Il est attendu de vous que vous délivriez sur WebCampus, dans l'espace prévu à cet effet, une archive en format standard (`.zip`, `.rar`, `.7z` ou autre) contenant :

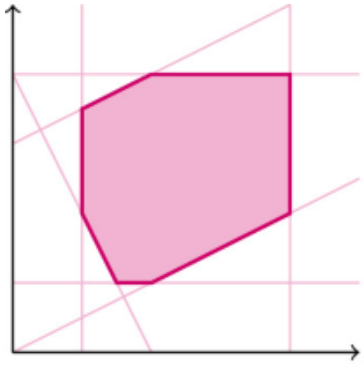
- d'une part, **un rapport au format PDF ou écrit à la main ;**
- d'autre part, **le code source de votre outil d'analyse.**

Choix du sujet

Pour ce travail, vous avez le choix entre deux sujets d'analyse.

Première possibilité

Le **Choix 1** consiste à établir une **analyse de polyèdres** pour un petit langage prototype dédié à la manipulation de points dans le plan cartésien. Vous devez définir la syntaxe et la sémantique du langage, puis développer une analyse permettant d'approximer les valeurs des variables en étudiant ce que l'on appelle une propriété *relationnelle* : plutôt que de retenir un intervalle de valeurs possibles pour chaque point, il sera possible de retenir des inégalités, telles que "le point A a une abscisse plus élevée que le point B" ou "le point A = (x,y) est tel que 2y-x <= 6" . Ce genre d'inégalités permettent de définir, pour chaque point, un polyèdre de possibilités. Votre analyseur doit être capable d'afficher le polyèdre calculé pour chaque variable (point) du programme, à chaque ligne du programme. Par exemple (tiré de Ziat, G., Maréchal, A., Pelleau, M., Miné, A., Truchet, C. (2020). Combination of Boxes and Polyhedra Abstractions for Constraint Solving. In: , *et al.* Formal Methods. FM 2019 International Workshops. FM 2019. Lecture Notes in Computer Science(), vol 12233. Springer, Cham. https://doi.org/10.1007/978-3-030-54997-8_8) :



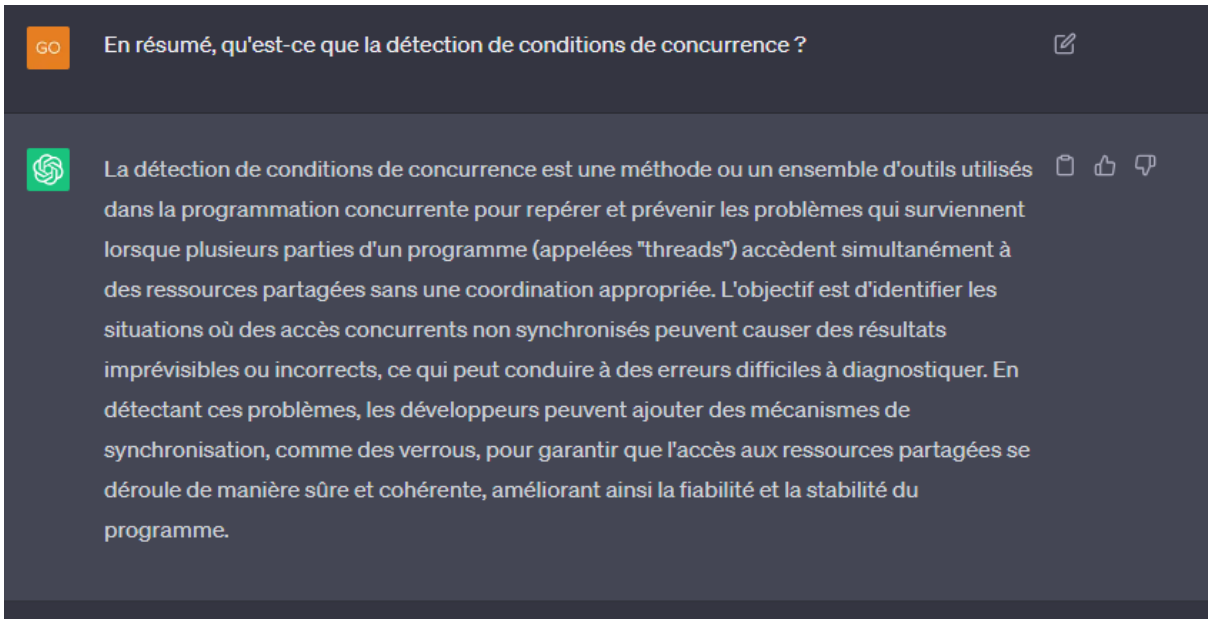
(a) A polyhedron.

$$
\begin{aligned}
x &\geq 1 \\
x &\leq 4 \\
y &\geq 1 \\
y &\leq 4 \\
2 \times y - x &\leq 6 \\
2 \times y - x &\geq 0 \\
2 \times x + y &\geq 4
\end{aligned}
$$

(b) Set of linear constraints.

Seconde possibilité

Le **Choix 2** consiste à étendre le langage Small pour le faire prendre en compte des aspects de *concurrence* ; c'est-à-dire que l'on pourra exécuter des portions de code en parallèle. À vous d'imaginer un jeu d'instructions permettant cela. Après en avoir défini la syntaxe et la sémantique, vous construirez une **détection de conditions de concurrence (*race detection* en anglais)**. Il s'agit d'identifier des cas où une même ressource (variable) est potentiellement requise par deux *threads* différents, sans que l'on puisse être certain de l'ordre d'exécution des deux instructions faisant usage de la ressource. Plus d'informations en ligne. Exemple (ChatGPT) :



Autre source intéressante : l'article "Static Application-Level Race Detection in STM Haskell using Contracts" disponible à l'adresse https://arxiv.org/abs/1312.2706. Cet article traite de la détection de conditions de concurrence en Haskell.

Imprécisions

Les deux choix exposés ci-dessus comprennent clairement des imprécisions : nous n'avons pas dit, par exemple, sur base de quels critères une condition de concurrence est considérée problématique. Un des objectifs du travail est que vous fassiez des recherches

sur votre sujet et preniez des décisions - qui forcément permettront de distinguer votre solution de celle de vos compères.

Consignes

Implémentation

Votre analyseur devra être implémenté dans l'un des langages suivants :

- Java ;
- Scala ;
- Python ;
- PHP ;
- Prolog
- JavaScript.

Le choix d'un autre langage peut et doit se justifier. Si vous souhaitez faire une telle demande, envoyez un mail à Gonzague en justifiant le langage que vous proposez.

Il vous est demandé que votre programme prenne, en entrée, (le nom d')un fichier contenant le programme à analyser. L'analyse doit ensuite tourner automatiquement, et afficher les résultats suivants : temps pris pour l'analyse, messages d'erreur, messages d'avertissements.

Pour le reste, ce travail est exploratoire de nature. Un grand nombre de libertés vous sont laissées.

- Vous êtes libres de choisir comment vous allez représenter exactement les programmes (ensemble de fonctions) qui sont les sujets de votre analyse, et comment vous allez définir quel est le point d'entrée (fonction « main ») de ces programmes. Dans tous les cas, **vous fournirez trois programmes d'entrée d'exemple,** de façon à ce qu'ils puissent être analysés rapidement pour tester votre analyse.
- Vous êtes libres de préciser les concepts restés flous dans l'énoncé**.** Il est normal que vous deviez explorer la matière en effectuant quelques recherches (en ligne, etc.) pour mieux comprendre le type d'analyse que vous allez implémenter. **Vous citerez vos sources externes dans le rapport**.
- En particulier, vous pouvez choisir d'*incorporer des notations spéciales, ou des annotations*, dans le langage des programmes à analyser. Si c'est votre choix, vous le documenterez et le justifierez dans votre rapport.

Rapport

Le rapport explicitera, de façon claire, formelle, précise et non-ambigüe, les points suivants.

1. Une description informelle de votre analyseur : quels sont ses objectifs, quelles sont ses capacités ?

2. La syntaxe et la sémantique concrète de votre langage (ou en tout cas des parties qui diffèrent du syllabus, ou des *collecting semantics* appropriés).
3. Le domaine abstrait et le treillis (*lattice*) sur lequel vous vous appuyez.
4. La fonction d'abstraction et la fonction de concrétisation.
5. Deux fonctions de flux (*flow functions*) que vous jugez être d'intérêt.
6. Votre analyse est-elle *sound* ? est-elle complète ? Si possible, prouvez, pour une de vos fonctions de flux du point précédent, la propriété de *local soundness*.
7. Une explication des différents cas d'erreur et d'avertissement que votre analyse détecte : sous quel conditions un tel cas survient-il ? Quels genres de messages l'analyse envoie-t-elle ?
8. Pouvez-vous donner une formulation de votre analyse, potentiellement un peu différente, en utilisant le framework GEN/KILL ?
9. Votre analyse est-elle *forward* ? *backwards* ? autre ?
10. Un exemple de programme d'entrée pour lequel vous expliquerez, étape par étape, les calculs effectués par votre analyse (suivant l'algorithme *worklist*).
11. La marche à suivre pour exécuter votre programme ;
12. Toute autre information que vous jugeriez utile de communiquer.

Cotation

Ce travail vaudra pour 50 % de la note du cours.

Les éléments principaux de cotation sont : la précision et l'adéquation du programme fourni, la qualité de contenu et de forme du rapport rendu avec le programme, la cohérence des choix de spécification et d'implémentation réalisés, l'élégance des solutions apportées, le taux d'investissement dans le travail (allant du minimum requis au dépassement des attentes).

Échéance

Le travail est à rendre pour le **10 novembre 2023 à 23h59**.

Des questions ?

Privilégiez le forum Questions-réponses sur WebCampus.

Bon travail !