Conception Formelle
DM : Un peu de théorie et de pratique.

METTEZ VOS NOMS ICI

2023-2024

Ce travail est à réaliser en binôme. La date de rendu est fixée au 11 mai 2024.

Les exercices sont par difficulté croissante, ils sont assez guidés et les questions sont relativement bien détaillées, mais faites bien attention à ne pas oublier des éléments dans les questions (certaines contiennent des sous-questions). Une attention particulière sera portée à la clarté des justification et au fait qu'elles montrent la compréhension de l'exercice ou non.

Les trois exercices devraient avoir un poids à peu près similaire dans la notation finale. Ce barème n'est qu'indicatif, mais le message de celui-ci est de traiter les questions dans l'ordre.

Structure du devoir et rendu Le devoir est constitué du présent document, ainsi que de fichiers de code à compléter en parallèle des questions posées. Le code fourni est découpé ainsi : un fichier .c pour chaque fonction. Ces fichiers sont à remplir au fur et à mesure.

Pour répondre aux questions, vous avez deux choix :

— Soit compléter le présent document .tex en plaçant vos réponses dans les blocs solutionorbox prévues à cet effet. Pour cela, référez-vous aux macros définies dans les sujets de TD. Si vous avez besoin de compiler avec la police OpenDyslexic, faites-le avec la ligne suivante :

```
latexmk -xelatex -usepretex='\def\dyslexic{}' TD-TP.tex
```

— Soit produire un document pdf par d'autres moyens (autre logiciel, scan), tant que c'est lisible, ça me conviendra.

Vous devrez rendre (sur la page moodle du cours) :

— Le présent document de réponse (ou votre version).

— Le dossier `code` complété. Dans les fichiers qui le composent, vous pourrez si besoin rajouter des commentaires expliquant comment prouver certaines spécifications si ce n'est pas immédiat, ou expliquer ce qui vous bloque.

Rappels de logique : On rappelle que :

— $p \Rightarrow q \equiv \neg p \vee q$

— $(p \wedge q) \Rightarrow r \equiv p \Rightarrow q \Rightarrow r$.

Vous aurez à manipuler des expressions comprenant des $\forall$. Pour ces expressions-là, lorsque vous appliquez un pas de WLP, grossièrement il vous faut découper votre formule entre partie modifiée et partie non-modifiée, et appliquer le calcul. Plus concrètement ici, ce que vous devriez avoir, c'est quelque chose du genre :

$$\begin{aligned}
&\mathbf{WLP}(t[i] = x, \forall j; 0 \leq j \leq i ==> \varphi(t[j])) \\
\equiv &\mathbf{WLP}(t[i] = x, (\forall j; 0 \leq j < i ==> \varphi(t[j])) \wedge \varphi(t[i])) \\
\equiv &(\forall j; 0 \leq j < i ==> \varphi(t[j])) \wedge \varphi(x)
\end{aligned}$$

La règle la plus générale d'où vient cela est la suivante : pour φ et ψ des formules quelconques :

$$\forall j; \varphi(j) \equiv \forall j; \psi(j) ==> \varphi(j) \wedge \forall j; \neg\psi(j) ==> \varphi(j);$$

Un autre point, c'est «soyez paresseux» : si vous arrivez à un moment sur un calcul équivalent à $\bot \Rightarrow \mathbf{WLP}(i-j, \phi)$, il est inutile de calculer $\mathbf{WLP}(i-j, \phi)$ pour déterminer que l'implication est vraie.

On attendra que vos calculs de WP soient suffisamment détaillés, mais vous pouvez sauter quelques étapes si elles sont faciles (comme par exemples, appliquer plusieurs substitutions d'un coup). Évidemment, tant que cela est correct.

Comme dit souvent, pour les justification de vérité de formule, on n'attendra pas de preuves formelles, mais des justifications convaincantes (i.e., qui n'oublient pas de cas, mais on restera tolérant sur la forme). En gros, quand vous aurez une implication, une possibilité sera de dire un truc du genre «l'implication est vraie car telle et telle partie de la partie gauche impliquent bien la partie droite». Un «ben oui» (ou plutôt un remplacement par $\top$) sera admissible pour des propriétés du genre $\bot \Rightarrow \varphi$, $\varphi \Rightarrow \top$ ou $a < b \Rightarrow a \leq b$. Cependant, pour faire cela de manière convaincante, vous auriez intérêt à simplifier vos formules avant.

Dans le présent sujet, on donne une version normalisée du code pour vous faciliter preuves (if then else développés, un seul return, que des while). Évidemment, vous pouvez coder autrement, mais il sera plus aisé de faire ces restrictions sur papier.

Attention : on manipule beaucoup de unsigned int. En pratique, dans vos formules, *quantifiez sur des integer, et non des int*.

Un mot sur les preuves : Certaines des preuves peuvent être difficiles pour les solveurs, aussi faites bien les trois points suivants avant de désespérer de vos spécifications :

— Vérifiez que le solveur Z3 4.12.2 (counter-examples) est activé (il parvient à démontré des propriétés où alt-ergo et les autres variantes de Z3 échouent – ne me demandez pas pourquoi). Attention, il y a trois variantes de Z3 installée au CREMI, et c'est bien celle avec counter-examples qui fonctionne dans certains cas. Pensez à

cliquer sur «filter» dans l'onglet Provers de frama-c si vous ne le voyez pas.
- Si certaines propriétés ne sont pas prouvées, retentez la preuve : quand les solveurs tentent trop de preuves en même temps, il arrive que certaines timeout pour de mauvaises raisons. Les relancer peut régler le problème.
- Prouvez d'abord le contrat de fonction, avant d'ajoutez les gardes RTE, puis prouvez ces dernières. Dans certains cas, tenter de tout prouver d'un coup peut ne pas fonctionner.
- Si cela ne marche toujours pas (et que vous avez confiance en la propriété), cliquez sur le nom du but, puis sur la tactique «filter». Cela peut parfois régler le problème (aucune garantie bien sûr).

Si rien de tout cela ne marche, vous avez probablement oublié de spécifier certaines hypothèses (ou votre propriété est fausse), donc reprenez votre stylo.

Exercice 1 : Mid

On considère la fonction suivante, `mid`. On la considère pour l'instant comme s'exécutant uniquement sur des entiers positifs.

```
1 /*@ requires a >= 0;
2 requires b >= 0;*/
3 int mid(int a, int b)
4 {
5     int r;
6     r = (a % 2 + b % 2) / 2;
7     return (a / 2 + b / 2 + r);
8 }
```

(a) Si ce n'est pas déjà fait et que vous rendrez le présent .tex, mettez vos noms dans la balise author situé en haut de ce document (celle où il y a écrit un message assez explicite). (0 points)

(b) Que fait cette fonction ? Donnez-en une post-condition de la forme `\result == X`, et qui n'est pas une paraphrase du code, mais une formule beaucoup plus simple (si vous ne voyez vraiment pas, testez-la sur plusieurs exemples). Décrivez également en français cette post-condition.

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(c) Calculez $\mathrm{WP}(5-7,\psi)$ où ψ est la post-condition donnée précédemment.

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(d) Pourquoi la formule obtenue est-elle toujours vraie (justifiez en raisonnant sur les valeurs possibles de `a%2` et `b%2`) ?

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(e) Les gardes RTE sont-elles toujours satisfaites ? Pourquoi ?

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(f) Quelle est la clause `\assigns` la plus restrictive vérifiée par cette fonction ? Ajoutez-là dans le fichier `.c`.

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(g) Considérez maintenant la seconde fonction du fichier, `mid_full`, qui est pour le moment la même (sauf qu'on a plus comme précondition que les arguments sont positifs). Les gardes RTE sont-elles toujours satisfaites ? Pourquoi ?

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(h) La même post-condition n'est par contre pas satisfaite. En utilisant frama-c, explorez les quatre cas possibles qui découlent du signe de a et de b et dites pour chacun dans lesquels ce code satisfait la postcondition déterminée plus haut.

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(i) Déduisez-en une modification simple (qui consiste en l'ajout d'instruction traitant les cas manquants et pas en la modification de ce qui est déjà là) de la fonction `mid_full` qui permet à la fonction de satisfaire la même postcondition, et ce tout en ayant les gardes RTE satisfaites avec pour seule précondition $\top$ (donc sans clause requires). Vous placerez le code directement dans le fichier `mid.c`.

Pourquoi (informellement) votre solution ne peut-elle pas provoquer d'overflow ?

Réponse :

METTEZ VOTRE RÉPONSE ICI.

Exercice 2 : Shift_pos

On considère maintenant la fonction suivante :

```
 1  void shift_pos(int *tab, int size)
 2  {
 3      int i = 0;
 4      while (i < size)
 5      {
 6          tab[i] += i;
 7          i++;
 8      }
 9      return;
10  }
```

(a) Commencez par donner, dans un format de documentation (type doxygen), une spécification de cette fonction. Vous pouvez l'écrire en français ou en anglais, mais soyez précis.

> **Réponse :**
>
> METTEZ VOTRE RÉPONSE ICI.

(b) Donnez une formule logique équivalente à la post-condition de votre spécification. On appellera dans la suite cette formule ψ.

> **Réponse :**
>
> METTEZ VOTRE RÉPONSE ICI.

(c) Déterminez un invariant de boucle $I1$ permettant de déduire que `i == size` est vrai en fin de boucle. Vous justifierez qu'on a bien $I1 \wedge \neg(i < size) \Rightarrow i == size$.

> **Réponse :**
>
> METTEZ VOTRE RÉPONSE ICI.

(d) Déterminez un invariant de boucle I_ψ permettant de déduire ψ en fin de boucle. Vous justifierez qu'on a bien $I1 \wedge I_\psi \wedge \neg(i < size) \Rightarrow \psi$.

> **Réponse :**
>
> METTEZ VOTRE RÉPONSE ICI.

Bon, les deux invariants précédents s'obtiennent bien avec les techniques génériques pour deviner des invariants. Néanmoins, si vous demandez son avis à Frama-c, vous devriez voir qu'il accepte de déduire votre post-condition avec ces deux invariants, mais pas de prouver que I_ψ est un invariant, et il refuse également que $I1$ soit établi. On va essayer de régler ça dans la suite de l'exercice.

(e) Calculez $\mathsf{WP}(6, \varphi)$ pour une formule φ quelconque.

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(f) Déduisez-en $\mathsf{WP}(6-7, I_\psi)$. Quelle hypothèse vous manque pour pouvoir prouver que $i < size \wedge I_\psi \Rightarrow \mathsf{WP}(6-7, I_\psi)$?

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(g) Donnez un invariant I_{same} qui permet de prouver que I_ψ est bien invariant. Vous justifierez que cela implique bien l'hypothèse déterminée ci-dessus (indication : on a déjà croisé ce cas dans les TD machine).

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(h) Calculez maintenant $\mathsf{WP}(3, I1 \wedge I_\psi \wedge I_{\mathsf{same}})$. Après simplification, quelle est la précondition nécessaire pour que, avec les invariants précédent, la fonction `shift_pos` soit correcte ?

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(i) Pour que Frama-C accepte la spécification du point précédent, il faut fournir à la boucle une clause loop assigns. Quelle est cette clause ?

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(j) Exhibez un variant qui démontre la terminaison de cette fonction, et, en faisant explicitement les calculs de WP nécessaires, démontrez que ce variant en est bien un.

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(k) Complétez le fichier `shift_pos.c` de manière à ce que Frama-C valide votre spécification et démontre que toutes les gardes RTE soient satisfaites. Vous aurez besoin de clauses requires supplémentaire. Vous n'oublierez pas de spécifier que la fonction termine, ainsi que d'ajouter une clause assigns à votre fonction.

(l) (bonus) En réalité, il est possible de modifier les invariants afin que la précondition déterminée à la question (h) soit inutile. Cela ne demande que de modifier l'invariant $I1$. Voyez-vous quoi faire ?

> **Réponse :**
>
> METTEZ VOTRE RÉPONSE ICI.

Exercice 3 : find_last

On considère la fonction suivante :

```
1  /*@ ensures Rightmost: \forall integer k; \result < k < size
2  ==> tab[k] != value;
3  ensures Value: tab[\result] == value || \result == size;
4  */
5  unsigned int find_last(int *tab, int value,
6  unsigned int size)
7  {
8      unsigned int i = 0;
9      unsigned int pos = size;
10     while (i < size)
11     {
12         if (tab[i] == value)
13             pos = i;
14         i++;
15     }
16     return pos;
17 }
```

Cette fonction renvoie la position la plus à droite d'un tableau égale à une valeur fixée en paramètre s'il y en a au moins une, et renvoie sa taille s'il n'y en a pas.

(a) Les postconditions données ici ne sont en fait pas assez précises. En effet, la fonction qui se contente de renvoyer `size` quoiqu'il arrive (nommée `find_last_wrong` dans le fichier de code) satisfait ces postconditions. Démontrez, en calculant $\mathrm{WP}(\texttt{find\_last\_wrong}, \psi)$ pour chaque post-condition que `find_last_wrong` satisfait bien ces deux post-conditions avec pour seule précondition, $\top$.

> **Réponse :**
>
> METTEZ VOTRE RÉPONSE ICI.

(b) La coupable est évidemment la postcondition Value. Proposez des postconditions à la place de Value qui décriront correctement le comportement de `find_last` tout en n'étant pas satisfaites par `find_last_wrong`. Indication : cela revient à préciser dans quels cas se produisent les cas décrits par Value.

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(c) Calculez WP$(16, \psi)$ pour chacune des postconditions (Rightmost et celles qui remplacent Value).

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(d) Calculez WP$(12 - 14, \varphi)$ pour une formule φ quelconque.

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(e) Déterminez, grâce aux deux questions ci-dessus des invariants de la boucle 10-15 permettant de démontrer les postconditions. Pour chaque invariant, explicitez (en français) ce qu'il signifie et pourquoi il est vrai (pour ce point, vous pouvez utiliser le calcul de la question précédente).

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(f) En calculant WP$(8 - 9, I)$ pour les invariants, déduisez-en des préconditions qui rendent vos postconditions vraies.

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(g) Exhibez un variant qui démontre la terminaison de cette fonction, et, en faisant explicitement les calculs de WP nécessaires, démontrez que ce variant en est bien un.

Réponse :

METTEZ VOTRE RÉPONSE ICI.

(h) Complétez le fichier `find_last.c` de manière à ce que frama-c certifie que votre fonction et correcte, termine et ne peut causer d'erreur à l'exécution. Vous n'oublierez pas d'inclure une clause assigns à votre fonction.

(i) Finalement, donnez une spécification précise au format doxygen de cette fonction en vous servant évidemment de tout ce qu'on vient de déterminer ici.

Réponse :

METTEZ VOTRE RÉPONSE ICI.