Examen de rattrapage d'Approches formelles pour la vérification de programmes

Master CNS

jeudi 24 juin 2021

Durée : 3h.
Documents autorisés sauf livres. Aucun appareil électronique n'est autorisé.
Ce sujet comporte 4 exercices indépendants, qui peuvent être traités dans l'ordre voulu. Il contient 3 pages.
Les questions précédées par le symbole ($\star$) sont des questions bonus qui pourront être traitées à la fin.
L'exercice 1 doit être rédigé sur une copie séparée des autres exercices.

Exercice 1 : EACL - Vérification à runtime

On s'intéresse à la fonction C find dont l'entête est la suivante :

```
int sorted_find(int *a, int n, int val)
```

1. Ecrire le contrat en ACSL associé à la spécification informelle suivante : *la fonction prend en entrée un tableau a trié dans l'ordre croissant , sa longueur n, et un int val à rechercher. Elle retourne l'index d'une cellule qui contient la valeur recherchée. Elle retourne -1 si la valeur val n'est pas présente dans le tableau.*
2. Regardons maintenant l'implémentation de la fonction `sorted_find` donnée ci-dessous.

```
int sorted_find(int *a, int n, int val){
   for (int i = 0; i < n-1; i++) {
     if (a[i] == val)
     return i;
     if (a[i] > val) return (-1);
   }
   return (-1);
}
```

   Soit le programme de test ci-dessous. Que se passe-t-il si on exécute cette fonction (sans utiliser le plugin EASCL) ?

```
int main(void) {
    int tab[4] = {2, 1, 0, 0};
    printf("%i\n", sorted_find(tab, 4, 0));
    return 0;
}
```

3. Que se passe-t-il si on exécute le programme de test précédent sans utiliser le plugin EACSL de Frama-C ?
4. Que se passe-t-il si on exécute le programme de test précédent en utilisant le plugin EACSL et le contrat que vous avez écrit à la première question ?

Exercice 2 : Preuve

Soient les formules suivantes :

(a) $(a \Rightarrow (b \Rightarrow c)) \Rightarrow (b \Rightarrow (a \Rightarrow c))$

(b) $(a \wedge b) \Rightarrow (b \wedge a)$

(c) $(\exists X.\ p(X)) \Rightarrow (\exists X.\ p(X))$

(d) $((\forall X.\ p(X)) \vee (\forall X.\ q(X))) \Rightarrow (\forall X.\ (p(X) \vee q(X)))$

Par chacune d'entre elles :

1. Calculer les clauses correspondant à la négation de la formule.
2. Donner une preuve par résolution. (Cf. fig. 2 page 3.)
3. Donner une preuve en déduction naturelle. (Cf. fig. 1 page 3.)

Par ailleurs,

4. ($\star$) Donner deux programmes OCaml dont le type est celui associé respectivement à la formule (a) et à la formule (b) via la correspondance de Curry–Howard–De Bruijn.

Exercice 3 : Conditions de vérification

1. Calculer les conditions de vérification pour les programmes et post-conditions suivantes :
   a) VC(`{a = 3 * b; b = 2 * a - b;}`, $2a = b$)
   b) VC(`if (a == b) b = 0; else a = a + b;`, $a \neq b$)
2. Montrez que les pré-conditions suivantes sont valides pour les programmes et post-conditions suivantes :
   a) Pré-condition : $x \neq y$ Programme `{ x = 3 * y; y = x - y; x = x - y; }` Post-condition $y = 2x$
   b) Pré-condition : $x \geq 0$ Programme `if (x > 127) x = 255; else x = 2 * x;` Post-condition $x \leq 255$

Exercice 4 : Preuve de programme

On considère le programme suivant :

```
int double(int n) {
  int x = 0;
  int y = x;
  while (x != n) {
    x = x + 1;
    y = y - 1;
  }
  return x - y;
}
```

On cherche à montrer que si n est positif ou nul, alors `double(`n`)` retourne le double de n.

1. Donner les spécifications en ACSL de la fonction.

$$\text{ax}\ \frac{}{\Gamma, A \vdash A} \qquad \bot\text{-e}\ \frac{\Gamma \vdash \bot}{\Gamma \vdash A}$$

$$\wedge\text{-e}_g\ \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \qquad \wedge\text{-e}_d\ \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B} \qquad \wedge\text{-i}\ \frac{A \qquad B}{A \wedge B}$$

$$\Rightarrow\text{-e}\ \frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B} \qquad \Rightarrow\text{-i}\ \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$$

$$\forall\text{-e}\ \frac{\Gamma \vdash \forall x.\ A[x]}{\Gamma \vdash A[t]} \qquad \forall\text{-i}\ \frac{\Gamma \vdash A[x]}{\Gamma \vdash \forall x.\ A[x]}\ x \text{ non libre dans } \Gamma$$

$$\neg\text{-e}\ \frac{\Gamma \vdash \neg A \qquad \Gamma \vdash A}{\Gamma \vdash \bot} \qquad \neg\text{-i}\ \frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}$$

$$\vee\text{-e}\ \frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C} \qquad \vee\text{-i}_g\ \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \qquad \vee\text{-i}_d\ \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$$

$$\exists\text{-e}\ \frac{\Gamma \vdash \exists x.\ A[x] \qquad \Gamma, A[x] \vdash C}{\Gamma \vdash C}\ x \text{ non libre dans } \Gamma, C \qquad \exists\text{-i}\ \frac{\Gamma \vdash A[t]}{\Gamma \vdash \exists x.\ A[x]}$$

Vous pouvez également utiliser ces deux règles dérivées :

$$\vee\text{-d}\ \frac{\Gamma, A \vee B, A \vdash C \qquad \Gamma, A \vee B, B \vdash C}{\Gamma, A \vee B \vdash C} \qquad \exists\text{-d}\ \frac{\Gamma, \exists x.\ A[x], A[x] \vdash C}{\Gamma, \exists x.\ A[x] \vdash C}\ x \text{ non libre dans } \Gamma, C$$

FIGURE 1 – Règles de la déduction naturelle

$$\text{Resolution}\ \frac{P \vee C \qquad \neg Q \vee D}{\sigma(C \vee D)} \qquad \text{Factoring}\ \frac{P \vee Q \vee C}{\sigma(P \vee C)}$$

σ est l'unificateur le plus général de P et Q

FIGURE 2 – Règles de la résolution

2. Calculer la condition de vérification pour le corps de la boucle `{ x = x + 1; y = y - 1; }` avec la post-condition $x + y = 0$.
3. Quelles sont les variables modifiées par le corps de la boucle ?
4. En déduire la condition de vérification de la boucle en entier, avec comme invariant de boucle $x + y = 0$ et comme post-condition $x - y = 2n$.
5. En déduire la condition de vérification du corps de la fonction en entier, avec la même post-condition.
6. Quelle famille de prouveurs automatiques serait a priori capable de la démontrer ?
7. Déduire de la condition de vérification qu'avec la précondition $n \geq 0$, le programme est correct.
8. A-t-on besoin de la précondition ?