Examen final
d'Approches formelles pour la vérification de programmes
Partie preuve de programmes

Master CNS

vendredi 8 novembre 2024

Durée : 1h30.
Documents autorisés sauf livres. Aucun appareil électronique n'est autorisé.
Ce sujet comporte 3 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.

Exercice 1 : Preuve

Soient les formules suivantes :

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

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

(c) $(\forall X.\ \forall Y.\ p(X, Y)) \Rightarrow (\neg \exists Z.\ \neg p(Z, Z))$

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, pour les formules (a) et (b),

4. Donner l'entrée au format DIMACS correspondant à la formule.
5. ($\star$) Donner un programmes OCaml dont le type est celui associé à la formule via la correspondance de Curry–Howard–De Bruijn.

Exercice 2 : Conditions de vérification

1. Pour chacun des programmes `p` et postcondition Q, calculer la condition de vérification $VC(\texttt{p}, Q)$.

	`p`	Q
a)	`x = 42; y = y - x; x = 2 * y`	$x > 0$
b)	`if (x == 2) y = x; else y = 2;`	$y = x$

2. Montrez que la pré-condition suivante est valide pour les programme et post-condition suivantes :
   Pré-condition : $x - y = a$
   Programme `if (x == y) ; else { x = 2 * (x + y); y = x - 2 * y; x = x - y;}`
   Post-condition $x - y = -2a$

Exercice 3 : Preuve de programme

On considère le programme suivant :

```
int length_of_positive_prefix(int *a, int s) {
  int i = 0;
  while (i != s && a[i] >= 0)
    i += 1;
  return i;
}
```

On cherche à montrer qu'il retourne la taille du plus grand préfixe du tableau `a` qui contient uniquement des valeurs positives ou nulles. Autrement dit :

— l'entier retourné est égal à `s`, ou alors on retourne l'indice d'une case de `a` où la valeur est strictement négative ;

— pour tous les entiers j entre 0 et l'entier retourné (strictement), la valeur de `a` en j est positive ou nulle.

1. Donner les spécifications en ACSL de la fonction.
   On mettra comme précondition que `a` est un tableaux de taille `s` positive ou nulle.
2. Calculer la condition de vérification pour le corps de la boucle `while` avec la post-condition
$$\forall j,\ 0 \leq j < i \Rightarrow \mathtt{a}[j] \geq 0 \tag{1}$$
3. Quelles sont la ou les variable(s) modifiée(s) par le corps de la boucle ?
4. En déduire la condition de vérification de la boucle en entier, avec comme invariant de boucle la formule (1) et comme post-condition
$$(i = s \vee a[i] < 0) \wedge \forall j.\ 0 \leq j < i \Rightarrow a[j] \geq 0 \tag{2}$$
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 $s \geq 0$, le programme est correct.

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

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

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

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

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

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

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

Vous pouvez également utiliser ces deux règles dérivées :

$$\vee\text{-d}\ \frac{\Gamma, A \vee B, A \vdash C \qquad \Gamma, A \vee B, B \vdash C}{\Gamma, A \vee B \vdash C}$$

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

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