Corrigé de l'examen d'Approches formelles pour la vérification de programmes

Master CNS

jeudi 5 novembre 2020

Exercice 2 : Preuve

Soient les formules suivantes :

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

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

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

(d) $(\exists X.\ (p(X) \wedge q(X))) \Rightarrow ((\exists X.\ p(X)) \wedge (\exists X.\ q(X)))$

Par chacune d'entre elles :

1. Calculer les clauses correspondant à la négation de la formule.

   (a)

$$\begin{aligned}
&(\neg(a \Rightarrow (b \Rightarrow c)) \Rightarrow ((a \Rightarrow b) \Rightarrow (a \Rightarrow c))\\
\Leftrightarrow\ &(a \Rightarrow (b \Rightarrow c)) \wedge (a \Rightarrow b) \wedge a \wedge \neg c\\
\Leftrightarrow\ &(\neg a \vee \neg b \vee c) \wedge (\neg a \vee b) \wedge a \wedge \neg c
\end{aligned}$$

   $\{\neg a \vee \neg b \vee c; \neg a \vee b; a; \neg c\}$

   (b)

$$\begin{aligned}
&\neg((a \vee b) \Rightarrow (b \vee a))\\
\Leftrightarrow\ &(a \vee b) \wedge \neg b \wedge \neg a
\end{aligned}$$

   $\{a \vee b; \neg b; \neg a\}$

   (c)

$$\begin{aligned}
&\neg((\forall X.\ p(X)) \Rightarrow (\forall X.\ p(X)))\\
\Leftrightarrow\ &(\forall X.\ p(X)) \wedge (\exists X.\ \neg p(X))\\
\Leftrightarrow\ &\forall X.\ (p(X) \wedge (\exists X.\ \neg p(X)))\\
\Leftrightarrow\ &\forall X.\ (p(X) \wedge (\exists Y.\ \neg p(Y)))\\
\Leftrightarrow\ &\exists Y.\ \forall X.\ (p(X) \wedge \neg p(Y))\\
\text{Skolem :}\quad &\forall X.\ (p(X) \wedge \neg p(c))
\end{aligned}$$

   $\{p(X); \neg p(c)\}$

(d)

$$\begin{aligned}
&\neg((\exists X.\ (p(X) \wedge q(X))) \Rightarrow ((\exists X.\ p(X)) \wedge (\exists X.\ q(X))))\\
\Leftrightarrow\ &(\exists X.\ (p(X) \wedge q(X))) \wedge ((\forall X.\ \neg p(X)) \vee (\forall X.\ \neg q(X)))\\
\Leftrightarrow\ &(\exists X.\ (p(X) \wedge q(X))) \wedge ((\forall Y.\ \neg p(Y)) \vee (\forall Z.\ \neg q(Z)))\\
\Leftrightarrow\ &\exists X.\ \forall Y.\ \forall Z.\ ((p(X) \wedge q(X)) \wedge (\neg p(Y) \vee \neg q(Z)))\\
\text{Skolem :}\quad &\forall Y.\ \forall Z.\ ((p(c) \wedge q(c)) \wedge (\neg p(Y) \vee \neg q(Z)))
\end{aligned}$$

$\{p(c); q(c); \neg p(Y) \vee \neg q(Z)\}$

2. Donner une preuve par résolution. (Cf. fig. 2 page 7.)

(a)
$$\text{Resolution}\ \dfrac{\text{Resolution}\ \dfrac{\text{Resolution}\ \dfrac{\neg a \vee \neg b \vee c \qquad a}{\neg b \vee c} \qquad \text{Resolution}\ \dfrac{\neg a \vee b \qquad a}{b}}{c} \qquad \neg c}{\square}$$

(b)
$$\text{Resolution}\ \dfrac{\text{Resolution}\ \dfrac{a \vee b \qquad \neg a}{b} \qquad \neg b}{\square}$$

(c)
$$\text{Resolution}\ \dfrac{p(X) \qquad \neg p(c)}{\square}\ \sigma = \{X \mapsto c\}$$

(d)
$$\text{Resolution}\ \dfrac{\text{Resolution}\ \dfrac{\neg p(Y) \vee \neg q(Z) \qquad p(c)}{\neg q(Z)}\ \sigma = \{Y \mapsto c\} \qquad q(c)}{\square}\ \sigma = \{Z \mapsto c\}$$

3. (a) On pose $\Gamma = a \Rightarrow b \Rightarrow c, a \Rightarrow b, a$

$$\Rightarrow\text{-i}\ \dfrac{\Rightarrow\text{-i}\ \dfrac{\Rightarrow\text{-i}\ \dfrac{\Rightarrow\text{-e}\ \dfrac{\Rightarrow\text{-e}\ \dfrac{\widehat{\vdash}\ \dfrac{}{\Gamma \vdash a \Rightarrow b \Rightarrow c} \qquad \widehat{\vdash}\ \dfrac{}{\Gamma \vdash a}}{\Gamma \vdash b \Rightarrow c} \qquad \Rightarrow\text{-e}\ \dfrac{\widehat{\vdash}\ \dfrac{}{\Gamma \vdash a \Rightarrow b} \qquad \widehat{\vdash}\ \dfrac{}{\Gamma \vdash a}}{\Gamma \vdash b}}{\Gamma \vdash c}}{a \Rightarrow b \Rightarrow c, a \Rightarrow b \vdash a \Rightarrow c}}{a \Rightarrow b \Rightarrow c \vdash (a \Rightarrow b) \Rightarrow a \Rightarrow c}}{\vdash (a \Rightarrow b \Rightarrow c) \Rightarrow (a \Rightarrow b) \Rightarrow a \Rightarrow c}$$

(b)
$$\Rightarrow\text{-i}\ \dfrac{\vee\text{-d}\ \dfrac{\vee\text{-i}_d\ \dfrac{\widehat{\vdash}\ \dfrac{}{a \vee b, a \vdash a}}{a \vee b, a \vdash b \vee a} \qquad \vee\text{-i}_g\ \dfrac{\widehat{\vdash}\ \dfrac{}{a \vee b, b \vdash b}}{a \vee b, b \vdash b \vee a}}{a \vee b \vdash b \vee a}}{\vdash (a \vee b) \Rightarrow (b \vee a)}$$

(c)
$$\Rightarrow\text{-i}\ \dfrac{\widehat{\vdash}\ \dfrac{}{\forall X.\ p(X) \vdash \forall X.\ p(X)}}{\vdash (\forall X.\ p(X)) \Rightarrow (\forall X.\ p(X))}$$

(Autre solution :
$$\Rightarrow\text{-i}\ \dfrac{\forall\text{-i}\ \dfrac{\forall\text{-e}\ \dfrac{\widehat{\vdash}\ \dfrac{}{\forall X.\ p(X) \vdash \forall X.\ p(X)}}{\forall X.\ p(X) \vdash p(X)}\ \text{en utilisant le terme } X}{\forall X.\ p(X) \vdash \forall X.\ p(X)}\ X \text{ non libre dans } \forall X.\ p(X)}{\vdash (\forall X.\ p(X)) \Rightarrow (\forall X.\ p(X))}$$
)

(d) En posant $\Gamma = \exists X.\ (p(X) \wedge q(X)), p(X) \wedge q(X)$

$$
\Rightarrow\text{-i}\ \dfrac{
  \exists\text{-d}\ \dfrac{
    \wedge\text{-i}\ \dfrac{
      \exists\text{-i}\ \dfrac{
        \wedge\text{-e}_g\ \dfrac{
          \widehat{\vdash}\ \dfrac{}{\Gamma \vdash p(X) \wedge q(X)}
        }{\Gamma \vdash p(X)}
      }{\Gamma \vdash \exists X.\ p(X)}\ \text{avec le terme } X
      \qquad
      \exists\text{-i}\ \dfrac{
        \wedge\text{-e}_d\ \dfrac{
          \widehat{\vdash}\ \dfrac{}{\Gamma \vdash p(X) \wedge q(X)}
        }{\Gamma \vdash q(X)}
      }{\Gamma \vdash \exists X.\ q(X)}\ \text{avec le terme } X
    }{\Gamma \vdash (\exists X.\ p(X)) \wedge (\exists X.\ q(X))}
  }{\exists X.\ (p(X) \wedge q(X)) \vdash (\exists X.\ p(X)) \wedge (\exists X.\ q(X))}\ X \text{ non libre dans } (\exists X.\ p(X)) \wedge (\exists X.\ q(X))
}{\vdash (\exists X.\ (p(X) \wedge q(X))) \Rightarrow ((\exists X.\ p(X)) \wedge (\exists X.\ q(X)))}
$$

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.

   Le type associé à (a) est `('a -> 'b -> 'c) -> ('a -> 'b) -> 'a -> 'c`.

   On peut considérer le programme

   ```
   let p = fun x -> fun y -> fun z -> x z (y z)
   ```

   qui correspond à la preuve en déduction naturelle.

   Le type associé à (b) est `('a, 'b) or -> ('b, 'a) or` avec

   ```
   type ('a, 'b) or = Left of 'a | Right of 'b
   ```

   On peut considérer le programme

   ```
   let p = fun x -> match x with Left a -> Right a | Right b -> Left b
   ```

   qui correspond à la preuve en déduction naturelle.

Exercice 3 : Conditions de vérification

1. Calculer les conditions de vérification pour les programmes et post-conditions suivantes :

   a) VC(`{a = 42; b = a - b; a = b - a;}`, $a = b$)
   = VC(`a = 42;`, VC(`b = a - b;`, VC(`a = b - a;`, $a = b$))))
   = VC(`a = 42;`, VC(`b = a - b;`, $b - a = b$))
   = VC(`a = 42;`, $(a - b) - a = a - b$)
   = $(42 - b) - 42 = 42 - b$
   qui peut être simplifiée en $0 = 42$.

   b)

   $$
   \begin{aligned}
   &\mathrm{VC}(\texttt{if (x > 100) x = 100; else ;},\ x > 42) \\
   &= (x > 100 \Rightarrow \mathrm{VC}(\texttt{x = 100;},\ x > 42)) \wedge (x \leq 100 \Rightarrow \mathrm{VC}(\texttt{;},\ x > 42)) \\
   &= (x > 100 \Rightarrow 100 > 42) \wedge (x \leq 100 \Rightarrow x > 42)
   \end{aligned}
   $$

   qui est logiquement équivalente à $x > 42$.

2. Montrez que les pré-conditions suivantes sont valides pour les programmes et post-conditions suivantes :

a) Pré-condition : $a > 0$ Programme `b = a + 2;` Post-condition : $b \geq 0$
On calcule VC(`b = a + 2;`, $b \geq 0$) $= a + 2 \geq 0$.
Comme $a > 0$ implique $a + 2 \geq 0$, le triplet est valide.

b) Pré-condition : $a = b$ Programme `if (a > b) m = a; else m = b;` Post-condition $m = b$
On calcule VC(`if (a > b) m = a; else m = b;`, $m = b$)
$= (a > b \Rightarrow$ VC(`m = a;`, $m = b$)$) \wedge (a \leq b \Rightarrow$ VC(`m = a;`, $m = b$)$)$
$= (a > b \Rightarrow a = b) \wedge (a \leq b \Rightarrow b = b)$.
Éventuellement, on peut simplifier cette proposition en une proposition logiquement équivalente $a \leq b$.
Comme $a = b$ implique bien cette proposition, le triplet est valide.

Exercice 4 : Preuve de programme

On considère le programme suivant :

```
int sum(int n) {
  int i, s;
  i = 0;
  s = 0;
  while (i != n) {
    s = s + i;
    i = i + 1;
  }
  return s;
}
```

On cherche à montrer que si n est positif ou nul, alors `sum`(n) retourne $\frac{n(n-1)}{2}$.

1. Donner les spécifications en ACSL de la fonction.
```
/*@ requires n >= 0;
  assigns \nothing;
  ensures \result == n * (n - 1) / 2; */
```
2. Calculer la condition de vérification pour le corps de la boucle `while` avec la post-condition $s = \frac{i(i-1)}{2}$.
VC(`{ s = s + i; i = i + 1; }`, $s = \frac{i(i-1)}{2}$)
= VC(`s = s + i;`, VC(`i = i + 1;`, $s = \frac{i(i-1)}{2}$))
= VC(`s = s + i;`, $s = \frac{(i+1)i}{2}$)
$= s + i = \frac{(i+1)i}{2}$
qu'on peut simplifier en $s = \frac{i(i-1)}{2}$.
3. Quelles sont les variables modifiées par le corps de la boucle ?
`s` et `i`.

4. En déduire la condition de vérification de la boucle en entier, avec comme invariant de boucle $s = \frac{i(i-1)}{2}$ et comme post-condition $s = \frac{n(n-1)}{2}$.

   VC(`while (i != n) { ... }`, $s = \frac{n(n-1)}{2}$
   $= \mathit{Inv} \wedge \mathit{Preserv} \wedge \mathit{Final}$

$$= \left(\begin{array}{l} s = \dfrac{i(i-1)}{2} \\ \wedge \left(\forall s.\ \forall i.\ \left(i \neq n \wedge s = \dfrac{i(i-1)}{2}\right) \Rightarrow \mathrm{VC}(\texttt{\{ s = s + i; i = i + 1; \}}, {\scriptstyle s=\frac{i(i-1)}{2}})\right) \\ \wedge \left(\forall s.\ \forall i.\ \left(i = n \wedge s = \dfrac{i(i-1)}{2}\right) \Rightarrow s = \dfrac{n(n-1)}{2}\right) \end{array}\right)$$

$$= \left(\begin{array}{l} s = \dfrac{i(i-1)}{2} \\ \wedge \left(\forall s.\ \forall i.\ \left(i \neq n \wedge s = \dfrac{i(i-1)}{2}\right) \Rightarrow s = \dfrac{i(i-1)}{2}\right) \\ \wedge \left(\forall s.\ \forall i.\ \left(i = n \wedge s = \dfrac{i(i-1)}{2}\right) \Rightarrow s = \dfrac{n(n-1)}{2}\right) \end{array}\right)$$

5. En déduire la condition de vérification du corps de la fonction en entier, avec la même post-condition.

   VC(`i = 0; s = 0; while ...`, $s = \frac{n(n-1)}{2}$)

$$= \left(\begin{array}{l} 0 = \dfrac{0(0-1)}{2} \\ \wedge \left(\forall s.\ \forall i.\ \left(i \neq n \wedge s = \dfrac{i(i-1)}{2}\right) \Rightarrow s = \dfrac{i(i-1)}{2}\right) \\ \wedge \left(\forall s.\ \forall i.\ \left(i = n \wedge s = \dfrac{i(i-1)}{2}\right) \Rightarrow s = \dfrac{n(n-1)}{2}\right) \end{array}\right)$$

   qu'on peut simplifier en $\left(\forall s.\ \forall i.\ \left(i = n \wedge s = \frac{i(i-1)}{2}\right) \Rightarrow s = \frac{n(n-1)}{2}\right)$

6. Quelle famille de prouveurs automatiques serait a priori capable de la démontrer ?
   La famille des prouveurs SMT (par exemple Z3 ou alt-ergo) qui sont capables de faire des preuves modulo l'arithmétique non-linéaire.

7. Déduire de la condition de vérification qu'avec la précondition $n \geq 0$, le programme est correct.

   Même sans précondition, la condition de vérification est toujours vraie.
   (Soit i et n quelconques.
   Supposons $i = n$ et $s = \frac{i(i-1)}{2}$.
   Puisque $i = n$, en remplaçant i par n dans $s = \frac{i(i-1)}{2}$ on obtient $s = \frac{n(n-1)}{2}$.)
   NB : la précondition sert en fait à montrer que la boucle termine.

8. ($\star$) Que se passe-t-il si on change la troisième ligne en `i = 1;` ? Peut-on toujours prouver la correction du programme ? On étudiera en particulier le cas où $n = 0$.

   Si on change cette ligne, dans le cas où $n = 0$ la boucle `while` et donc le programme ne terminent jamais.

   La preuve de correction fonctionne toujours : seule la première des trois partie de la condition de vérification est modifiée en $0 = \frac{1(1-1)}{2}$, mais elle est toujours vraie. On

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

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

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

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

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

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

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

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

a en fait montré la correction partielle du programme : si le programme termine, alors la spécification est vérifiée. Prouver la correction totale oblige à montrer la terminaison du programme, ce qui n'est pas le cas ici pour la précondition $n \geq 0$.

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

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

FIGURE 2 – Règles de la résolution