SAP — TD10 — Pot pourri interprétation abstraite

28 avril 2010

1 Remplissage d'un tableau

On considère le programme suivant :

```
int t[20];

int main() {
  int i;
  for(i=0; i<20; i++) {
    t[i] = 1;
  }
}
```

Nous rappelons qu'en C, les variables globales sont initialisées à zéro.

1.1 Méthode B

1. Que vérifie le tableau t à la fin de la fonction main ?
2. Proposez un invariant de boucle qui permette de le démontrer.

1.2 Interprétation abstraite

On analyse ce programme par interprétation abstraite en avant sur le domaine des intervalles.

1. Dans un premier temps, on a mis une case abstraite pour représenter l'ensemble des cases du tableau. Faites tourner l'analyse sur main(). Qu'obtient-on comme information ?
2. Même question si on met une case abstraite par indice du tableau (donc une case pour t[0], ..., une pour t[19).
3. Que proposez-vous pour améliorer ce résultat ?

2 Mise à l'échelle d'un vecteur

On considère le code suivant :

```
int i;
for(i=0; i<20; i++) {
  t[i] = t[i] * 3;
}
```

Nous appelons ce programme sur un tableau de 20 cases, toutes contenant 1.

2.1 Méthode B

1. Que vérifie le tableau t à la fin de ce fragment de code ?
2. Proposez un invariant de boucle qui permette de le démontrer.

2.2 Interprétation abstraite

On analyse ce programme par interprétation abstraite en avant sur le domaine des intervalles.

1. Dans un premier temps, on a mis une case abstraite pour représenter l'ensemble des cases du tableau. Faites tourner l'analyse. Qu'obtient-on comme information ?
2. Même question si on met une case abstraite par indice du tableau (donc une case pour t[0], ..., une pour t[19).
3. Que proposez-vous pour améliorer ce résultat ?

3 Invariant de classe

```
class Compteur {
    int compte, maxi;

    Compteur(int maxi) throws Exception {
        this.maxi = maxi;
    }

    void incrementer() throws Exception {
        compte++;

        if (compte > maxi) {
            compte = maxi;
            throw new Exception("compteur trop grand");
        }
    }

    void decrementer() throws Exception {
        compte--;

        if (compte < 0) {
            compte = 0;
            throw new Exception("compteur negatif");
        }
```

```
    }
}
```

1. Proposez et justifiez un invariant de classe.
2. Ce programme a un bug, résultant en un invariant un peu curieux. Corrigez-le.

4 Verrous

```
int do_the_funk() {
  lock(structure);
  if (age < 18) {
    /* you cannot do the funk if under age */
    return 1;
  }
  really_do_the_funk();
  unlock(structure);
  return 0;
}
```

1. Trouvez le bug et proposez une correction.
2. Proposez une méthode d'analyse simple (pas forcément complète ou correcte) pour ce type de bugs.

5 Deux threads Java

En Java, la lecture ou l'écriture d'un mot de 32 bits est réputée atomique. Le type **long** désigne un entier de 64 bits.

```
class Toto {
  long compteur;
  void on() {
    compteur=-1;
  }
  void off() {
    compteur=0;
  }
}
```

Plusieurs threads appellent les fonctions on() et off().

1. Proposez un invariant de classe raisonnable.
2. Expliquez pourquoi ça risque de ne pas être un invariant correct en présence de plusieurs threads.
3. Proposez une correction.

6 Synchronisations

```
static String toto() {
  StringBuffer p = new StringBuffer();
  transformer(p);
  for(int i=0; i<1000; i++) {
    p.append("*" + i);
  }
  return p.toString();
}
```

Notre compilateur avancé a-t-il le droit de remplacer StringBuffer par StringBuilder ? À quelle condition ?