

# Correction TD 5: Mise en pratique!

Où l'on aime les ordinateurs.

Dans ce TD, on va utiliser un outil codé par votre enseignant pour programmer quelques algorithmes dans un langage fait maison  $M^I \nearrow L$ , ensuite calculer le graphe de contrôle de flot, puis et proposer des tests, et voir si ces tests couvrent les critères «tout les nœuds», «tout les arcs», «couverture par chemins indépendants».

Si l'outil plante sans laisser de message, veuillez demander de l'aide à votre encadrant.

Pour ne pas avoir trop à lire, vous êtes invités à expérimenter les choses, même si ce n'est pas dans le TD, tout n'est pas exprimé avec précision, si vous avez une erreur, essayez de comprendre pourquoi.

Exercice 1 | Première mise en main

Le langage  $M^I \mathcal{N} L$  permet de définir des fonctions manipulant des entiers. Toutes les variables et opérations arithmétiques traitent des entiers strictement positifs. Ainsi si a < b l'opération a - b donne 0.

```
1
  fun condition(a,b)
2
      local res;
      if a>b then
3
4
         res:=a
5
      else
6
         res:=b
7
      end:
8
      return res
9
  end
```

```
1 fun condition2(a,b)
2 if a>b then
3 return a
4 else
5 return b
6 end
7 end
```

```
1
   fun descente(a,b)
2
       local c;
3
       c:=a;
4
       local d1:=b ;
       while (d1>0\&\&a<b) do
5
6
          print d1;
7
          d1 := d1 - c
8
       end;
9
       return 0
10
   end
```

Voilà trois exemples de fonctions. Commencez par les recopier dans un fichier exo1.mini. Il s'agit d'un programme d'études, donc on va créer un deuxième fichier exo1.study

```
from("exo1.mini")
1
2
   study firstStudy with(descente)_extendedCFG
3
     drawCFG("descenteCFG.dot")_ps
4
     interpret (1,0)
5
6
     interpret (2,10)
7
     test(1,0)=0
8
     showPaths("descenteCFGPaths.dot")_pdf
9
10
   study simpleCondition with(condition)
11
     test(1,2)=2
12
     test(6,0) = 6
13
     drawCFG("cond.dot")_pdf
14
15
   end
16
17
   study conditionWithManyReturns with(condition2)
18
     test(1,2)=2
19
     test(6,0) = 6
     drawCFG("condRet.dot")_pdf
20
21
   end
```

Assurez vous que la commande dot soit disponible, si ce n'est pas le cas, installez graphviz. Si dot n'est toujours pas disponible, n'utilisez pas les suffixes \_pdf ou \_ps mais ouvrez les fichiers .dot avec l'interface graphique de graphviz.



Vous voyez ici trois études relatives aux trois fonction définies précédemment. Tout d'abord le fichier commence en indiquant où trouver ces fonctions from("file.mini"). Ce fichier doit toujours commencer par cette ligne. Ensuite le fichier liste les études study nameOfStudy with(functionName) ... end.

N'attendez plus longtemps et exécutez le logiciel en ligne de commande ./MiniLStudy exo1.study

Observez la sortie et commentez la. Au besoin modifiez les fichiers pour mieux comprendre

Correction : La sortie est constituée de lignes " $d1 = \dots$ " représentant les instructions print. Sinon la sortie est scindée en blocs commençant par le nom de l'étude, puis par les nombres de test passés et la couverture par nœus, arcs et complexité cyclomatique (le nombre de chemins indépendants testé est indiqué ainsi que la complexité cyclomatique)

Dans ce TD on ne s'intéresse pas vraiment aux résultats des tests, seulement à la couverture du graphe. Vous avez surement remarqué de nouveaux fichiers dans le répertoire. Remarque : les fichiers .dot sont des formats intermédiaires, ils sont ensuite convertis en pdf ou en ps.

C2 Ouvrez ces fichiers, essayez de comprendre à quoi ils correspondent. Entre autre, à quoi le suffixe extendedCFG correspond-il? Vous aurez besoin d'expérimenter pour bien répondre à la question

Correction : Les nouveaux fichiers sont des graphes, ils correspondent aux CFG générés avec l'instruction drawCFG. Le suffixe \_extendedCFG permet de laisser le graphe tel quel, sinon il est compacté (les nœuds vides sont supprimés si possible, les nœuds de commandes sont fusionnés quand possible). L'instruction showPaths permet en complément d'indiquer les nœuds et arcs empruntés par le flot dans les tests.



### Exercice 2 Et maintenant on pratique

Petite aide : les opérateurs suivants sont disponibles : + - \* / < > <= >= != == && || !. Il n'y a pas de conversion d'entier en booléen (N'écrivez pas if a then... mais if a>0 then...)

On se propose de reprendre l'algorithme d'Euclide et la multiplication rapide du TD 1.

Implémentez ces fonctions dans un fichier exo2.mini Remarque : l'opération modulo n'existe pas, mais vous trouverez bien un moyen, non?

Correction :

```
fun pgcd(a,b)
2
      while a!=b do
3
        if (a>b) then
           a := a - b
4
5
        else
           b := b - a
6
7
        end;
        a:=a
8
9
      end;
10
      return a
11
   end
```

```
fun multrap(a, b)
2
      local r:=0;
3
      while a>0 do
        if (a/2)*2 == a then
5
           a := a/2
6
        else
7
           r:=r+b;
           a := (a-1)/2
8
9
        end:
10
        b := 2 * b
11
      end;
12
      return r
13
   end
```

Pour la fonction pgcd, trouvez des DT pour couvrir les trois critères de couverture (par nœud, par arc, par chemins indépendants). Vous avez un outil pour expérimenter, votre justification peut juste se réduire au fichier exo2.study, non? Correction :

```
1 study pgcd with (pgcd)
2 test(5,3)=1
3 test(2,2)=2
```

Raphaël Charrondière

raphael.charrondière@inria.fr

Semaine 8 - échéance S9

```
4 test(3,5)=1
5 end
```

# (3) Idem pour la multiplication rapide Correction:

```
1  study multrap with (multrap)
2  drawCFG( "multrap.dot" )_pdf
3  test(2,3)=6
4  test(0,5)=0
5  test(5,1)=5
6  end
```

## **★ \$ >**

# Exercice 3 Entraînement pour l'indépendance : mise en pratique

Reprenez l'exo 2 du TD 4.

Créez une fonction qui traduit le graphe donné dans un fichier exo3.mini

Vérifiez que la base de chemin que vous aviez trouvé est bien une base de chemins indépendants. Évidement que je veux votre fichier exo3.study



#### Exercice 4 | Saurez-vous trouver la voie?

Est-ce que l'outil peut valider une couverture par nœuds et non par arcs? Ceux qui ne justifient pas devrait lire Tintin (C'est une blague.)

Reprenez le graphe du parti (TD 4), pouvez-vous trouver une fonction dont il est le GCF? Dans le cas échéant un graphe similaire? Et sinon pourquoi?

