Validation d'algorithmes
L3 MIAGE, UGA
2020/2021

Raphaël Charrondière
raphael.charrondiere@inria.fr
Semaine 8 - échéance S9

Feuille de 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 MINIL , 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 MINIL 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.

```
fun condition(a,b)
    local res;
    if a>b then
       res:=a
    else
       res:=b
    end;
    return res
end
```

```
fun condition2(a,b)
    if a>b then
       return a
    else
       return b
    end
end
```

```
fun descente(a,b)
    local c;
    c:=a;
    local d1:=b ;
    while(d1>0&&a<b) do
       print d1;
       d1:=d1-c
    end;
    return 0
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")

study firstStudy with(descente)_extendedCFG
  drawCFG("descenteCFG.dot")_ps
  interpret(1,0)
  interpret(2,10)
  test(1,0)=0
  showPaths("descenteCFGPaths.dot")_pdf
end

study simpleCondition with(condition)
  test(1,2)=2
  test(6,0)=6
  drawCFG("cond.dot")_pdf
end

study conditionWithManyReturns with(condition2)
  test(1,2)=2
  test(6,0)=6
  drawCFG("condRet.dot")_pdf
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.

Raphaël Charrondière
raphael.charrondiere@inria.fr

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`

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

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.*

2➤ 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*

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.

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

2➤ 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 ?

3➤ Idem pour la multiplication rapide

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

Reprenez l'exo 2 du TD 4.

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

2➤ 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 ?

1➤ 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.)*

2➤ 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 ?
