TP1 - AST
Control Flow Graph

À rendre avant le Jeudi 23 Janvier à 22h

1 Mise en place (avec Eclipse)

Téléchargez l'archive `tp1.tgz` sur `foad.univ-rennes1.fr` et extrayez-la (de préférence dans votre workspace Eclipse, ne pas hésiter à en créer un pour AST).

Méthode 1 Dans Eclipse, importer le projet : File > Open Projects from File System.... Pour l'Import source selectionnez le dossier (Directory...) de décompression `tp1`. Appuyez sur Finnish.

Méthode 2 Dans Eclipse, importez le projet : File > Import.. > Existing Projects into Workspace. Appuyez Next puis sur Select root directory > Browse..., et retrouver votre dossier de décompression `tp1`. Le projet devrait apparaître dans "Projects". Appuyez sur Finish.

Le projet devrait apparaître dans votre "Package Explorer".

L'analyse attend un fichier RTL en argument. Nous allons préciser cela dans la configuration de l'exécution de `MyTest.java` (la classe contenant un `main`).

Faites un clic droit sur `MyTest.java` > Run As ... > Run Configurations. Dans l'onglet Arguments (2ème onglet) > Program arguments : Variables... . Dans le type de variables choisissez `file_prompt` puis appuyez sur Ok. En appuyant sur Run vous pouvez tester le bon fonctionnement du projet.

Un explorateur de fichier devrait s'ouvrir à chaque execution, pour vous demander un fichier. Les fichiers de test RTL sont dans `/examples/rtl/`. Par exemple, le résultat dans la console pour `Factorial.rtl` devrait être :

```
TP1 ANALYSIS of func Main(a)

TP1 ANALYSIS of func Fac.ComputeFac(this num)
```

2 Travail demandé

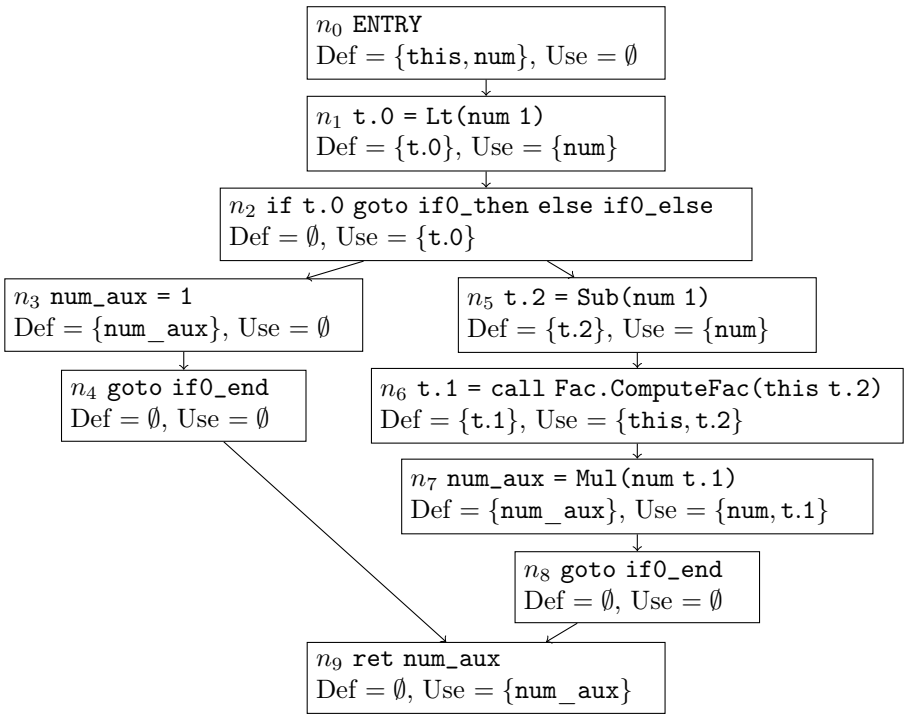
Le but de ce TP est d'implémenter une analyse pour construire le Control Flow Graph d'une fonction.

— Un nœud correspond à une instruction (normale ou terminale), ou au noeud spécial d'entrée de la fonction.

— Un arc relie une paire d'instructions consécutives dans l'exécution de la fonction.

— Chaque nœud contient l'ensemble des variables utilisées (*use*) et définies (*def*) par l'instruction.

Par exemple, le RtlFlowGraph de la fonction Fac.ComputeFac est présentée ci-dessous.



Le résultat dans la console sera alors :

```
TP1 ANALYSIS of func Fac.ComputeFac(this num)
  n0: [num, this] []; goto [n1]
  n1: [t.0] [num]; goto [n2]
  n2: [] [t.0]; goto [n3, n5]
  n3: [num_aux] []; goto [n4]
  n4: [] []; goto [n9]
  n5: [t.2] [num]; goto [n6]
  n6: [t.1] [this, t.2]; goto [n7]
  n7: [num_aux] [t.1, num]; goto [n8]
  n8: [] []; goto [n9]
  n9: [] [num_aux]; goto []
```

Une ligne donne dans l'ordre :

```
nom du nœud: [liste des defs] [liste des uses]; goto [liste des successeurs]
```

Structure du code

Votre code peut entièrement être rédigé dans le fichier `TP1RtlFlowGraph.java`.

Le constructeur `TP1RtlFlowGraph.(Function f)` doit construire le graphe associé à la fonction f passée en argument. Étudiez la classe `MyTest` pour comprendre l'utilisation de cette classe. Remarquez que `TP1RtlFlowGraph` hérite de la classe `FlowGraph`.

Le langage RTL est décrit dans le package `rtl`. Pour analyser une fonction dans ce langage, il vous faudra utiliser le *Visitor Pattern*. L'interface des *Visitor*s a déjà été implémenté pour les instructions normales (`InstrVisitor`), les instructions terminales (`EndInstrVisitor`) et les opérandes (`OperandVisitor`). Il n'est pas nécessaire d'implémenter de nouvelles interfaces pour les visiteurs, vos différents interpréteurs peuvent se contenter de celles-ci.

Une fois le graphe construit, la deuxième étape consiste à compléter les fonctions de consultations du graphe.

- `Object instr(Node n)` doit retourner l'instruction associée au nœud n. L'objet retourné est donc de type `Instr` ou `EndInstr`. Si le nœud n est le nœud d'entrée, renvoyer `null`.
- `Set<Ident> def(Node n)` Renvoie l'ensemble des identifiants (les variables) définis par l'instruction du nœud n.
- `Set<Ident> use(Node n)` renvoie l'ensemble des identifiants (les variables) utilisés par l'instruction du nœud n.
- `Set<Ident> entry()` renvoie le nœud correspondant à l'entrée de la fonction.

3 Rendu

Pour tester votre rendu, lancez le terminal dans le dossier `srctp1` (qui contient `tp1`). Exécutez-y les commandes suivantes :

```
chmod u+x test
./test examples/rtl/Factorial.rtl
```

Si la compilation se déroule sans erreur et le résultat est celui attendu, déposez votre fichier `TP1RtlFlowGraph.java` (et uniquement ce fichier !) sous Moodle. Cela ne vous dispense pas, bien sûr, de faire d'autres tests...