LOG3210 - Élément de langages et compilateur
TP4 : Variables Vives

Ettore Merlo – Professeur
Doriane Olewicki – Chargée de cours et de laboratoire

Hiver 2020

1 Objectifs

- Identifier les sucesseurs et prédécesseurs de chaque statement d'un bloc de code;
- Identifier les variables référencées et déclarées de ces statements;
- Extraire les variables vives d'un bloc de code.

2 Travail à faire

Dans ce TP, vous implémenterez l'algorithme d'extraction des variables vives. Celui-ci est explicité dans un document ("variables vives (v2)") sur moodle. Le pseudo code de cet algorithme est le suivant:

```
forall (node in nodeSet) {
   IN[node] = {}
   OUT[node] = {}
}

workList = {}
workList.push(stop_node);

while(!workList.empty()) {
   node = workList.pop();

   for (succNode in successors(node)) {
      OUT[node] = OUT[node] union IN[succNode];
   }

   OLD_IN = IN[node];
   IN[node] = (OUT[node] - DEF[node]) union REF[node];

   if(IN[node] != OLD_IN) {
      for(predNode in predecessors(node)) {
         workList.push(predNode);
      }
   }
}
```

Nous vous demandons de gérer les trois cas suivants :

- Statement basique (assignation);
- IfStatement (avec et sans else);
- WhileStatement.

Les prédécesseurs et successeurs de Statements basiques et IfStatement sont déjà implémentés, on vous demande de gérer vous même le WhileStatement.

Je vous encourage à commencer par générer pour chaque step les sets REF et DEF. **Ensuite**, SUCC et PRED lors du parcours de l'arbre pour les WhileStmt. Ensuite, quand le code remonte au Noeud Programme, vous exécutez l'algorithme Variables Vives (Code sources: `compute_IN_OUT()`). Donc, l'algorithme de variables vives est appliqué APRES le parcours de l'arbre.

Pas mal d'outil sont déjà mis en place dans le code source pour vous aider, notamment:

- La classe `StepStatus` (pour représenter les sets à chaque step);
- La Map `allSteps` (pour représenter tous les steps;
- `previous_step` (pour représenter les derniers noeuds visités, utile pour déterminer les PRED à chaque step et aussi à la fin pour connaitre les noeud-STOP).

Vous êtes libres de ne pas utiliser ces outils.

3 Barème

Le TP est évalué sur 20 points, les points étant distribué comme suit :

- REF/DEF correct : 6 points;
- PRED/SUCC corrects pour les WhileStattement : 7 points;
- Algorithm variables vives correct (IN/OUT corrects) : 7 points.

La moitié des points de chaque catégorie seront attribués si REF, DEF, SUCC et PRED sont corrects (voir code source et tests).

L'ensemble des tests donnés sous Ant doivent passer au vert pour que le laboratoire soit réussi. La qualité du code sera aussi vérifiée et prise en compte pour la correction.

4 Remise

Le devoir doit être fait en **binôme**. Remettez sur Moodle une archive nommée *log3210-tp4-matricule1-matricule2.zip* avec uniquement le fichier `VariablesVivesVisitor` ainsi qu'un fichier `README.md` (si vous le désirez) contenant tous commentaires concernant le projet.

L'échéance pour la remise est le **21 Mars 2020 à 23h55** .

Une pénalité de 10 points (50%) s'appliquera par jour de retard. Une pénalité de 4 points (20%) s'appliquera si la remise n'est pas conforme aux exigences (nom du fichier de remise, fichiers à rendre).

Si vous avez des questions, veuillez me contacter sur moodle ou sur mon courriel : doriane.olewicki@gmail.com.