LOG3210 - Éléments de langages et compilateurs
TP5 : Expressions disponibles et Définitions valides

Chargée de cours
Doriane Olewicki

Chargé.e.s de laboratoire
Quentin Guidée, Raphaël Tremblay, Sara Beddouch

Hiver 2023

1 Objectifs

- Calculer les expressions disponibles IN et OUT pour chaque bloc de base;
- Calculer les définitions valides IN et OUT pour chaque instruction;
- Effectuer des optimisations de code pour améliorer l'efficacité et la performance du programme.

2 Travail à faire

Dans ce TP, l'objectif est d'implémenter des analyses de flux pour les expressions disponible et les définitions valides. Une fois ces analyses réalisées, vous devrez mettre en place une optimisation qui permettra l'élimination:

- `D'expressions communes;`
- `D'affectations multiples;`
- `De code mort.`

Langage de ce TP

Dans ce TP, nous travaillons à nouveau avec du code intermédiaire généré lors du TP3 (voir dossier `data`). Notre objectif cette fois est d'optimiser ce code (voir dossier `"expected"`) avant de le transformer en code machine (TP4). Nous allons donc nous concentrer uniquement sur des lignes de code intermédiaire, qui ne contiennent que des opérations arithmétiques et des assignations.

2.1 Expressions disponibles

Exercice 1

Dans ce premier exercice, l'objectif est d'implémenter le calcul des ensembles GEN et KILL des expressions disponibles. Pour ce faire, il est recommandé de se référer aux notes de cours afin de comprendre la logique d'implémentation.

Exercice 2

Dans ce deuxième exercice, il faut implémenter l'algorithme des expressions disponibles (voir annexe A).

Exercice 3

Dans ce troisième exercice, l'objectif est d'implémenter l'optimisation par élimination d'expression commune. Cette technique d'optimisation permet de repérer les expressions qui se répètent dans le code et de ne les évaluer qu'une seule fois, en remplaçant les occurrences répétées par la variable qui correspond à la même expression (voir dossier `"AvailableExpressionTest > expected"`). Pour l'implémenter, vous devrez utiliser les ensembles d'expressions disponibles IN et OUT.

2.2 Définitions valides

Exercice 1

L'objectif est d'implémenter le calcul des ensembles GEN et KILL pour l'analyse de type définitions valides. Pour ce faire, il est recommandé de se référer aux notes de cours afin de comprendre la logique d'implémentation.

Exercice 2

Dans ce deuxième exercice, il faut implémenter l'algorithme des définitions valides (voir annexe B).

Exercice 3

Dans cet exercice, vous devez mettre en œuvre deux techniques d'optimisation : l'élimination des affectations inutiles et l'élimination du code mort. L'élimination des affectations inutiles vise à identifier et supprimer les assignations qui ne sont pas nécessaires. De même, l'élimination du code mort consiste à détecter les portions de code qui ne contribuent pas à l'exécution du programme (par exemple, des variables inutilisées) et à les retirer. Ces approches permettent de réduire la taille du code et d'améliorer les performances en supprimant les éléments superflus. Pour mettre en œuvre ces optimisations, vous pouvez vous appuyer sur les résultats des tests situés dans le dossier `"ReachingDefinitionTest > expected"`. Vous devez ainsi implémenter les deux méthodes "computeSingleAssignment()" et "eliminateDeadCode()" en utilisant les tests pour valider votre code.

3 Barème

Le TP est évalué sur 20 points, distribués comme suit :

Fonctionnalités	
Expressions disponibles	/4
Définitions valides	/4
Optimisation d'expressions communes	/4
Optimisation d'affectations multiples	/4
Optimisation de code mort	/4
Pénalités	
Retard (par jour de retard)	-10
Non-respect des consignes de remise (nom du fichier,...)	-4
Code de mauvaise qualité	-2
Total	**/20**

4 Remise

L'échéance pour la remise du TP5 est le `18 Avril 2023 à 23:55`.

Remettez sur Moodle une archive nommée `log3210-tp5-matricule1-matricule2.zip` (tout en minuscule), contenant **uniquement** les 3 fichiers suivants:

- `AvailableExpressionVisitor.java`;
- `ReachingDefinitionsVisitor.java`;
- `README.md` (facultatif, à fournir si vous avez des commentaires à ajouter sur votre projet).

Le devoir doit être fait en **binôme**. Si vous avez des questions, veuillez nous contacter sur Discord via le canal de votre équipe (`#equipe-00`), sans nous mentionner! Les chargés vous répondront dès que possible, s'ils ont du temps libre en dehors des séances!

A Algorithme: Expressions disponibles

- `IN` := `Expr_IN`
- `OUT` := `Expr_OUT`

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

while(changes to any IN or OUT occur){
    for (i = 0; i < nodeList.size(); i++) {

        curr = nodeList[i]

        if (i > 0) {
            IN[curr] = OUT[nodeList[i - 1]]
        }

        OUT[curr] = GEN[curr] union (IN[curr] - KILL[curr])
    }
}
```

B Algorithme: Définitions valides

- `IN` := `ValDef_IN`
- `OUT` := `ValDef_OUT`

```
idem que "Expressions disponibles".
```