LOG3430 – Méthodes de test et de validation du logiciel

Laboratoire 4

Tests basés sur les états

Département de génie informatique et de génie logiciel

École Polytechnique de Montréal

Hiver 2020

1. Mise en contexte théorique

L'approche des tests basés sur les états de Chow consiste à générer un arbre des transitions à partir du diagramme d'état d'un programme ou d'une composante sous test. Les chemins qu'on peut trouver dans cet arbre représentent les transitions et les nœuds représentent les états. Chaque transition dans l'arbre est un chemin qui part d'un état S et revient à ce même état. Le retour à l'état S marque la fin de l'arc dans l'arbre. Donc, il ne peut pas y avoir des répétitions de chemins dans l'arbre. Cette méthode de test nous permet de modéliser les transitions simples, qui ne contiennent pas de boucles, afin de générer les cas de tests correspondants. Pour plus de détails et d'exemples, voir les notes de cours sur les Tests d'états.

Exemple :

L'exemple dans la Fig1. montre comment fonctionnent les transitions. Chaque transition change d'état. Cependant, pas tous les états sont accessibles à partir de tous les états. Par exemple, vous ne pouvez pas faire dodo à l'école. RÉVEILLEZ-VOUS!

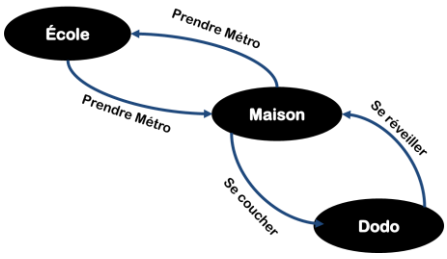


Fig1. Exemple d'états et de transition

Dans ce travail, nous souhaitons tester le fonctionnement d'un micro-parseur syntaxique Parcer_FSM.py. Il s'agit d'écrire un analyseur de langage pour une syntaxe minimale avec utilisation de règles pour détecter des instructions à exécuter (interprétateur ou compilateur). Il existe bien sûr des méthodes traditionnelles d'analyse syntaxique d'un langage (ex., lex / yacc[1]).
Exemple :
La figure 2 suivante montre un exemple simple du parseur.

[1] https://books.google.ca/books/about/Lex_Yacc.html?id=YrzpxNYegEkC&source=kp_book_description&redir_esc=y

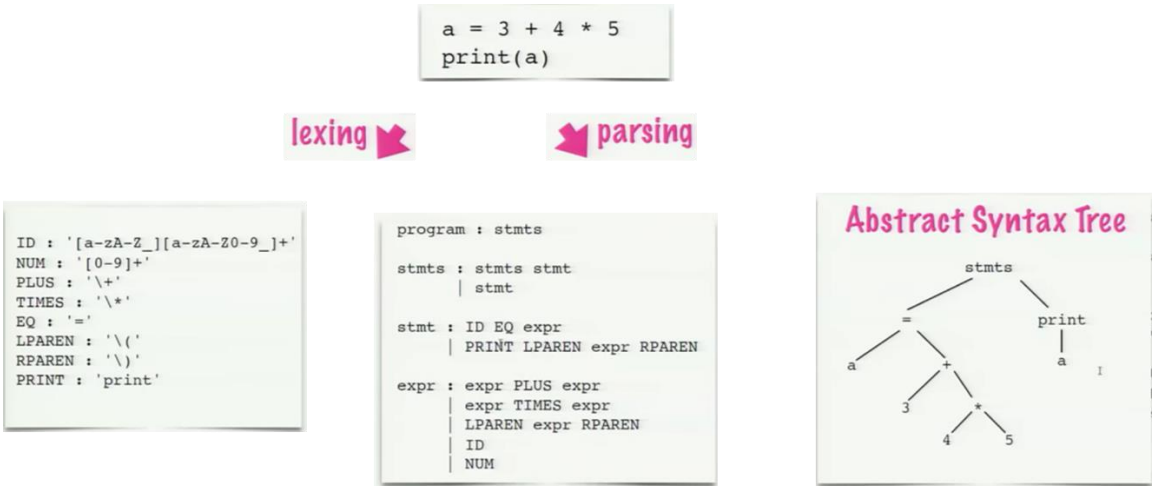


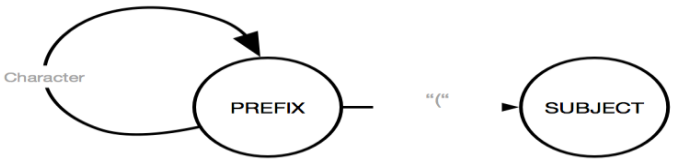
Fig2. Exemple de lexing, parsing et construction du graph[2]

2. Objectifs

- Représenter la classe de l'interpréteur « *Parcer_FSM.py* » avec un diagramme d'états
- Générer l'arbre des transitions de Chow à partir du diagramme d'états.
- Identifier les cas de tests et les implémenter à l'aide de Unittest.

3. Mise en contexte pratique

Pour réaliser ce travail, il faut commencer par construire le diagramme d'états du parseur (interpréteur) « *Parcer_FSM.py* ». Ci-dessous la première transition comme indication.



Ensuite, il faut générer l'arbre des transitions correspondant en spécifiant les états et les conditions de garde pour chaque transition.

Finalement, il faut identifier les cas de test à partir de l'arbre trouvé et implémenter les séquences trouvées avec Unittest. Voir des exemples indicatifs dans la présentation.

4. Travail à effectuer

[2] Réf. David Beazley - Reinventing the Parser Generator, 2018.

4.1. Construire le diagramme d'états de l'interprétateur « *Parcer_FSM.py* » et donner à chaque transition un nom, à chaque état également.

4.2. Construire l'arbre des transitions de l'interprétateur « *Parcer_FSM.py* ». Créer une table de transition des changements d'état en explorant le code et proposez de nouvelles transitions possibles.

4.3. Identifiez tous les cas de tests avec les conditions à partir de l'arbre trouvé.

4.4. À l'aide de Unittest, écrire une classe de test unitaire pour tester les cas de test identifiés dans la question précédente.

4.5. À l'aide de l'outil Coverage.py, évaluez la couverture de l'interprétateur « *Parcer_FSM.py* » selon la couverture des branches et identifiez les branches non couvertes, s'il y en a.

5. Livrables attendus

Les livrables suivants sont attendus :

- Un rapport pour le laboratoire avec des réponses sur les questions.
- Le dossier COMPLET contenant le projet.

Le tout à remettre dans une seule archive zip avec pour nom matricule1_matricule2_lab1.zip à téléverser sur Moodle.

Le rapport doit contenir le titre et numéro du laboratoire, les noms et matricules des coéquipiers ainsi que le numéro du groupe.

Consultez le site Moodle du cours pour la date et l'heure limite de remise des fichiers. Un retard de]0,24h] sera pénalisé de 10%, de]24h, 48h] de 20% et de plus de 48h de 50%.