Département IMA / 4A (S8)
Informatique Fondamentale 2010/2011

Laure Gonnord et Julien Forget
http://laure.gonnord.org/pro/

Projet Tutoré

Réalisation d'un compilateur de langage de patchworks

Introduction

Source (avec l'aimable autorisation des auteurs) :
http://www.enseignement.polytechnique.fr/profs/informatique/Philippe.Chassignet/02-03/INF_431/dm_1.html

Il s'agit d'écrire un compilateur de langage de patchworks[1], en Java. Un patchwork sera donné sous forme d'un fichier texte, et votre programme sera capable de l'interpréter et de l'afficher dans une fenêtre. Par exemple si le fichier `patchwork.txt` contient une description de patchwork :

```
./launch.sh patchwork.txt
```

fournira l'affichage :

1 Plantons le décor !

Le langage patchwork

- Un patchwork est un assemblage réalisé à partir des 4 dessins *primitifs* (ou patchs) suivants : 'a'  , 'b'  , 'c'  et 'd'  ;
- 3 opérations permettent ensuite de réaliser des assemblages de patchworks (qu'ils soient primitifs ou eux mêmes des assemblages) :
  - la *concaténation* permet de juxtaposer deux patchworks l'un à côté de l'autre horizontalement : a+b+c+d =  ;
  - la *rotation* permet de faire tourner un patchwork de 90 degrés dans le sens des aiguilles d'une montre : a+rot a+rot rot a+rot rot rot a =  ;
  - la *répétition* permet de copier le même patchwork plusieurs fois : (b+d)[3] =  .

Exercice 1 *Un exemple de patchwork est donné Figure 1. Représentez (au brouillon pour l'instant) les différentes étapes de la construction de ce patchwork (ie les patchworks intermédiaires correspondant aux sous-expressions). Vous fournirez cette explication pas-à-pas dans votre rapport.*

1. `http://fr.wikipedia.org/wiki/Patchwork`

FIGURE 1 – Le patchwork pour (rot (b+rot rot rot b)+ rot (rot b+rot rot b))[3]

Syntaxe concrète textuelle et sémantique des programmes Nous définissons maintenant le langage textuel complet sous forme de grammaire BNF :

```
PROG ::= | INSTRUCTION
         | INSTRUCTION PROG

INSTRUCTION ::= | DEFINITION
                | SHOW
                | SIZE

DEFINITION ::= def IDENTIFICATEUR = PATCHWORK ;

SHOW ::= show PATCHWORK ;

SIZE ::= size PATCHWORK ;

PATCHWORK ::= PRIMITIVE
            | IDENTIFICATEUR
            | ( PATCHWORK )
            | PATCHWORK + PATCHWORK
            | rot PATCHWORK
            | PATCHWORK[ENTIER]
```

Explications :

- un programme (`PROG`) est donc une suite d'instructions ;
- une `INSTRUCTION` est soit une `DEFINITION` (cf ci-dessous), soit de la forme `show PATCHWORK` (affichage dans une fenêtre), soit de la forme `size PATCHWORK` (affichage de la taille du patchwork dans le terminal, ie le nombre de lignes et de colonnes) ;
- une `DEFINITION` permet de sauver un patchwork dans une variable. Par exemple :
  ```
  def x = rot a;
  show x;
  ```
  définit un nouveau patchwork nommé `x` dont la valeur vaut `rot a`. On peut utiliser `x` ensuite comme un patchwork (ici on demande à afficher le patchwork correspondant à `x`).

Un patchwork peut être construit des différentes manières suivantes :

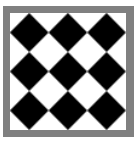
- une `PRIMITIVE`, valant 'a', 'b', 'c' ou 'd' (les carrés de base) ;
- un `IDENTIFICATEUR` (autre que 'a', 'b', 'c' ou 'd') faisant référence à une variable précédemment définie ;
- une concaténation de deux patchworks (+) ;
- une rotation (rot) ;
- une répétition d'une valeur entière d'un autre patchwork.

Priorités : `[]` est prioritaire sur `rot` qui est prioritaire sur `+` (concaténation).

EXERCICE 2 *Un exemple de programme complet est décrit Figure 2. Détaillez le sens de chacune des instructions du programme (au brouillon). Vous fournirez cette explication pas-à-pas dans votre rapport.*

```
def losange = rot (b+rot rot rot b)
              +rot (rot b+rot rot b);
def ligne = losange[3];
def damier = (rot ligne)[3];
show damier;
```

(a) Définition



(b) Interprétation

FIGURE 2 – Exemple de programme avec son interprétation

2 Travail demandé

2.1 Séance 1 - Analyse syntaxique

Préliminaires Vous devez tout d'abord récupérer les outils `jflex` et `java cup` permettant de compiler votre grammaire, ainsi que les fichiers `Makefile` et `launch.sh` écrits par nos soins.

1. Récupérer l'archive SVN créé pour le Tutorat sur le serveur svn de l'école [2] :

   `svn checkout svn+ssh://synthe/ima4/ProjetIF`

2. Installer JFlex dans un répertoire de votre compte (copier, puis décompresser `jflex-1.4.3.tar.gz` qui se trouve dans le répertoire Tools) ;
3. Copier `java-cup-11a.jar` dans ce même répertoire d'installation ;
4. Faire un premier `checkout` de votre projet :

   `svn checkout svn+ssh://synthe/ima4/<nombinome>`

5. Copier dans le répertoire <nombinome> `Makefile` et `launch.sh` (compilation, exécution de vos programmes)
6. Ajouter les droits d'exécution au fichier `launch.sh` ;
7. Éditer ces 2 fichiers pour indiquer les bons chemins vers `jflex` et `java-cup`.
8. `svn add`, `svn commit` :-)

REMARQUE 1 *JFlex a été téléchargé à l'adresse http://jflex.de/jflex-1.4.3.tar.gz et JavaCup à l'adresse http://www2.cs.tum.edu/projects/cup/java-cup-11a.jar.*

La grammaire patchwork

1. Lire attentivement la description du langage et faire les exercices 1 et 2.
2. Dans un répertoire nommé `quilt` de votre archive SVN, écrire le fichier `flex` et le fichier `cup` qui correspondent à la grammaire de patchworks. On se contentera pour l'instant de faire uniquement l'analyse syntaxique, sans actions associées aux règles de la grammaire ; (ne pas oublier de commiter)
3. Tester sur les programmes patchworks fournis (dans `ProjetIF/Progs`). Au début de la séance 2, cette analyse syntaxique devra fonctionner (nous tiendrons compte de tout retard pour la note finale du tutorat).

2.2 Séance 2 - AST

1. Créer la hiérarchie de classes Java permettant de représenter l'AST d'un patchwork ;
2. Ajouter dans le fichier `cup` les actions permettant de construire l'AST correspondant à un patchwork contenu dans un fichier texte.

2. De l'extérieur, c'est plus compliqué, mais expliqué au semestre dernier (cf cours de PA)

2.3 Séance 3 - Interprétation des patchworks

Nous vous renvoyons au site web cité en début d'énoncé pour la manière d'interpréter un patchwork (voir Section 4).

API d'affichage de patchwork dans une fenêtre Copiez les fichiers `PatchWork.java` et `PatchWorkInterface.java` de l'archive SVN "ProjetIF" (répertoire `patchwork`) et placez les dans le répertoire contenant les sources de votre projet.

Ces fichiers définissent une API, *que vous ne devez pas modifier*, contenant les fonctions de base permettant d'afficher des patchworks dans une fenêtre. Toutes les fonctions dont vous avez besoin sont décrites dans l'interface, excepté le constructeur `PatchWork()` qui permet de créer un nouveau patchwork vide.

Interprétation des instructions Il s'agit maintenant de donner un sens à un AST représentant un patchwork. Vous devez donc réaliser l'interprétation des trois instructions apparaissant dans un AST de patchwork. Il est conseillé de les traiter dans l'ordre ci-dessous :

1. `size` : Laissez pour l'instant de côté le cas des patchworks contenant des variables.
2. `show` : cf section 4.2 de la page web citée en intro. On notera que :
   - L'affichage d'un patchwork nécessite à plusieurs endroits de connaître sa taille ;
   - L'évaluation de la répétition d'un patchwork est très similaire à celle de la concaténation (donc mettre au point l'une des deux complètement avant de coder la deuxième) ;
   - Laissez pour l'instant de côté le cas des patchworks contenant des variables.
3. `def` : Cette instruction a pour effet de rajouter une définition de variable dans un *environnement de variables*. Cet environnement de variables devra être rajouté en paramètre de la fonction d'évaluation (section 4.3 de la page web citée en intro) afin de pouvoir compléter l'interprétation des instructions `size` et `show` avec la gestion des variables.

Pour aller plus loin Regarder les extensions proposées par le site initial.

3 Mise en oeuvre et consignes pour le rendu

Mise en oeuvre Expliquée plus haut (séance 1).

Rendu Nous récupérerons **le 10 mai 2011 20h** (5 points par jour de retard) vos projets dans vos dépôts qui devront avoir la structure suivante :

- un fichier `Readme.txt` contiendra une description rapide de votre logiciel.
- un répertoire `Code` (avec Makefile+launch). Pour faciliter la correction, la classe principale s'appellera `Ifdessin`.
- un répertoire `Progs` qui comprendra au moins les dessins fournis par l'énoncé
- éventuellement, un répertoire old contenant du code inutile.
- un fichier `nomdubinome.pdf` contiendra votre rapport. Le rapport ne comprendra pas plus de 5 pages, devra être clair et précis et notamment comporter les limitations de votre outil.

Attention ! votre dépot SVN ne comprendra ni `jflex` ni `javacup` !

Nous fournirons un script python qui permettra de vérifier les consignes. Des points seront enlevés aux binômes pour lesquels le script renvoie `Fail`.