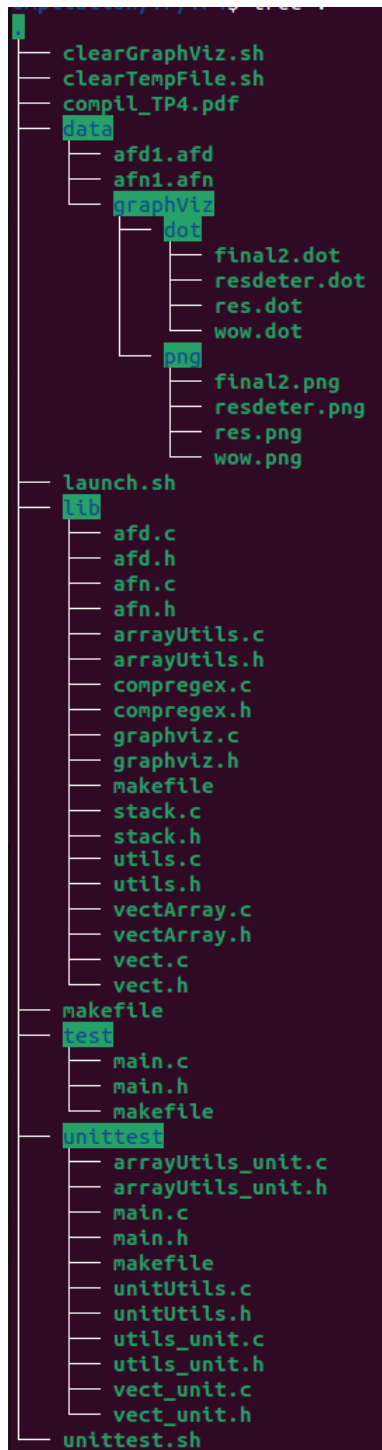

I53 - DM 2

Adam Chareyre

Table des matières

| | |
|----------------------------------|---|
| Arborescence du projet | 2 |
| Contenu de l'archive | 2 |
| Problèmes et bugs | 3 |
| Exemples | 3 |
| Grammaire | 4 |
| Conversion d'AFN et AFD en image | 4 |

Arborescence du projet



Contenu de l'archive

Vous trouverez plusieurs fichiers .sh :

- cleaGraphViz.sh : permet de supprimer tous les fichiers .dot et .png du dossier data/graphViz
- clearTempFile.sh : permet de supprimer tous les fichiers .a, .o et executables du projet

- `launch.sh` : permet de compiler le projet
- `unittest.sh` : permet de lancer les tests unitaires du projet ou des tests particuliers

Vous trouverez également le fichier `makefile` à partir duquel le correcteur pourra lancer la commande `make` pour compiler le projet (execute le fichier `launch.sh`).

Le dossier `unittest` contient les tests unitaires du projet mais peut également servir de zone de test pour le correcteur.

Le dossier `data` contiendra des `afn` et `afd` sous la forme de fichier. Le dossier contiendra également des images correspondant au graph des `afn` et `afd` (leur fichier `dot` sera stocké en parallèle dans le dossier `dot`).

Le dossier `lib` contient la majeure partie du code source du projet.

Le dossier `test` contient le fichier à partir duquel l'exécutable `mygrep` est construit.

Problèmes et bugs

Un unique bug à été trouvé, dans un cas très particulier, l'étoile de Kleene ne fonctionne pas comme attendue.

Si l'expression régulière est de la forme `"E*"` où `E` est une expression régulière et que la chaîne donnée en entrée vaut la chaîne vide, alors, la chaîne sera dite non-acceptée alors qu'elle devrait l'être.

Cependant pour des expressions régulières de la forme `"E1 op E2* op E3"` avec `E1` et `E2` des expressions régulières, la chaîne acceptera bien une chaîne de caractère de la forme `"E1E3"`.

Par exemple :

- `./grep "a*" "a"` fonctionnera comme attendu
- `./grep "a*" "aaaa"` fonctionnera comme attendu
- `./grep "a*" ""` ne fonctionnera pas comme attendu
- `./grep "ba*" "baaa"` fonctionnera comme attendu
- `./grep "ba*" "b"` fonctionnera comme attendu

Sauf oubli, aucune fuite de mémoire n'a été trouvée.

Exemples

Une série d'exemple se trouveront dans le fichier `makefile`, vous pourrez les lancer à partir de la commande `makefile test`.

Grammaire

$E1 \rightarrow C1E2$

$E2 \rightarrow +C1E2 \mid \varepsilon$

$C1 \rightarrow KC2$

$C2 \rightarrow \text{if } ((c \geq 'a' \ \&\& \ c \leq 'z') \parallel c == '(' \parallel c == '' \parallel c == '[') \text{ KC2}$

`addToTextStack('.')`;

$K \rightarrow P$ if $c == '*'$ `addToTextStack('*')` else if $(c == '')$ recopie du dernier et ajout d'un point ;

$R \rightarrow \text{letter}$ `addToTextStack(c)` ; `addToTextStack('')` ; $\mid \varepsilon$

$P \rightarrow E1$ si $c == '('$, letter R sinon si $c == '['$ sinon letter

letter $\rightarrow [a-z]$

Conversion d'AFN et AFD en image

Dans un soucis de facilitation de correction pour le correcteur, un module de visualisation des AFD et AFN a été développé. Vous pourrez créer l'image correspondant à l'un des deux avec un `(afn/afd)_toPng(afd/afn, nomfichier)`.

Vous trouverez le fichier image généré dans le dossier

`./data/graphViz/png/nomfichier.png`.

Attention ! l'outil `graphViz` est nécessaire. S'il n'est pas installé sur votre machine, vous pourrez l'installer avec la commande `"sudo apt install graphviz"`.

Par exemple, pour l'afn correspondant à l'expression régulière `"(a+b)c"` on aura le graph suivant d'affiché :

