Fiche de TD-TP no. 2

Évaluation et typage

Exercice 1. Exercice no. 2.6 de Amadio (évaluation et typage).

Exercice 2. Montrez qu'ils existent des expressions du langage `lse` qui s'évaluent, mais qui ne peuvent pas être typés.

Expressions regulières

Exercice 3. Proposez des expression régulières (sur l'alphabet $\{a, b, c\}$) pour les langages suivants :

1. le langage des mots avec un seul b,
2. le langage des mots avec exactement n b,
3. le langage des mots avec au plus n b,
4. le langage des mots avec un seul b, et au moins un a,
5. le langage des mots dont toute lettre non finale est telle que :
   - si c'est a, alors elle est suivie par a ou par b,
   - si c'est b, alors elle est suivie par b ou par c,
   - si c'est c, alors elle est suivie par c ou par a.

Remarque : pour ce dernier langage, on construira d'abord un automate qui le reconnaît et ensuite on déduira l'existence d'une telle expression régulière. A votre discrétion, on construira cette expressions en utilisant des résultats connus de la théorie des langages.

Exercice 4. Proposez des AFN qui reconnaissent les langages de l'exercice 3.

Exercice 5. Expliquez la signification des symboles suivants dans le contexte d'une expression régulière Lex :

```
*   +   |   ?   [   ]   {   }   (   )   \   "   .   ^   $   /
```

Pour chaque symbole proposez un cas d'utilisation.

Exercice 6. Considérez les règles suivantes d'un fichier Lex :

```
la|le|une|un  printf("article\n");
[a-z A-Z]+    printf("mot\n");
```

Comment l'analyseur lexicale subdivise la chaîne

```
le chat aime dormir
```

Justifiez votre réponse.

Exercice 7. Considérez le fichier lex suivants :

```
%option main
entier    0|[1-9][0-9]+
rationnel {entier}(\/{entier})?
%%
{rationnel}          printf("rationnel\n");
{entier}             printf("entier\n");
%%
```

La compilation du fichier produit un avertissement :

```
$ flex ce_fichier.lex
2.lex:6: warning, la règle ne peut être pairée
```

Comment vous allez modifier le code ? Justifiez votre réponse.

En TP

Lex

Exercice 8 : *Extraction des références d'un fichier html.* Dans un fichier html, les références distantes sont les valeurs des attributs href des balises `<a ...>` et les valeurs des attributs src des balise `<img ...>`. Notons d'abord qu'une balise peut avoir plusieurs attributs, séparés par des (suites d') espaces. Un espace pouvant être, un espace, un saut à la ligne ou même une tabulation... Ensuite, les attributs qui nous intéressent sont de la forme `NOM_ATTR=arg`, où `arg` est normalement n'importe quoi (ou presque !) entre doubles quotes `"` ou simples quotes `'`. Le presque veut dire que l'on peut mettre des simples quotes dans un argument cité entre doubles quotes et des doubles quotes dans dans un argument cité entre simples quotes (ouf !).

Question 1 : Écrire le fichier de description `url.lex` permettant de récupérer l'ensemble des références distantes d'un fichier html, et uniquement elles.

Par exemple, à partir d'un fichier dont le contenu serait :

```
<html
  xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
<head><title>Enseignements </title></head>
<body><div id="entete"> <img
src="../photo.jpg" alt="photo" width="131" height="121" />
<a HREF="http://www.lif.univ-mrs.fr">
<img id="lif" SRC="../shared/lif.gif" alt="Logo LIF" /> </a>
</div><h1> Teaching</h1><div id="corps"> <h2> 3<sup>ième</sup>
année de licence </h2>
<p> <A href="https://tele.ctes.univ-provence.fr/moodle/course/view.php?id=179
  Compilation </A> </p> <a
href="http://www.lif.univ-mrs.fr/~reyraud/">Retours</a> </div>
</body> </html>
```

votre fichier doit aboutir au résultat :

```
../photo.jpg http://www.lif.univ-mrs.fr ../shared/lif.gif
https://tele.ctes.univ-provence.fr/moodle/course/view.php?id=179
http://www.lif.univ-mrs.fr/~reyraud/
```

Interpréter les références distantes. Une fois récupérée les références distantes, on remarque qu'elle ont des aspects assez variables. Par exemple, dans l'url complète `http ://www.lif.univ-mrs.fr/ reyraud/`, on note un protocole `http` (qui pourrait être autre chose, par exemple, `ftp`), un nom de domaine `www.lif.univ-mrs.fr` et ensuite un chemin d'accès `/ reyraud/`. De manière très simplifiée, protocole et nom de domaine peuvent être omis, s'ils sont les mêmes que ceux du document. Par exemple l'url minimale / renvoie normalement à la racine du serveur.

Question 2 : Il vous est demandé de compléter votre analyseur url.lex, afin de déstructurer les url.

Exercice 9. Dans cet exercice nous nous proposons d'écrire un assembleur pour la machine virtuelle.
Modifiez votre exercice 11 afin que la machine virtuelle lise un fichier binaire (codez les instructions de la machine par des entiers).
Utilisez l'outil lex pour générer un traducteur du langage symbolique de la machine virtuelle vers son codage par entiers.

D'autres exercices (en TP)

Exercice 10 : *Syntaxe abstraite.* Écrivez des structures de donnés en C, aptes à représenter la syntaxe abstraite du langage `lse`.

Exercice 11. Écrivez un programme `virt.c` qui soit une implémentation de la machine virtuelle présentée en cours. La commande `virt fichier.lse` exécutera la séquence d'instructions `lse` contenue dans `fichier`. Après avoir exécuté une instruction `return`, elle affichera à l'écran le contenu de la case $M[0]$.

Exercice 12 : *Prise en main de yacc-bison.* Modifiez le fichier `lexeur.lex` de l'exercice 8 (TP 1) et intégrez-le avec un fichier `parseur.y` qui fasse l'analyse syntaxique d'un fichier contenant un programme `lse`. On produira un programme `lseparseur` qui affichera à l'écran si un programme est syntaxiquement correcte ou non.

Exercice 13. Ajoutez des actions au fichier `parseur.y` de l'exercice 12 pour générer votre premier compilateur. Le programme affichera sur la sortie standard une traduction d'un programme `lse` dans le langage de la machine virtuelle.