Aix*Marseille université

Domaine Sciences et Technologies
L3 Informatique (maths-info)

Compilation : Projet
Code UE : SIN5U16

Année 21-22

Générateur d'analyseur lexical

Ce projet s'adresse aux étudiants de L3 informatique, parcours maths/info. Ce **projet obligatoire** (c'est-à-dire que l'absence de rendu vous rendra défaillant à l'UE), sera pris en compte dans l'évaluation de l'UE seulement s'il permet de faire monter votre note de l'examen de "Compilation". **Il comptera pour 1/5 de la note finale dans ce cas**, les quatre autres cinquièmes étant réservés à la note de l'examen terminal de fin de semestre.

1 Objectif général

Ce projet a pour objectif de vous faire travailler sur les éléments majeurs du processus de compilation qui ont trait à l'analyse lexicale. Plus précisément, vous aurez l'occasion par ce projet de mettre en œuvre l'ensemble des notions vues en cours à ce sujet, depuis la création d'un arbre de syntaxe d'une expression rationnelle jusqu'à l'obtention de l'automate fini déterministe minimal associé à cette expression.

2 Précisions

Votre objectif général est de procéder au développement d'un outil informatique qui consiste en **un générateur d'analyseur lexical** (nommé Gal ci-dessous afin de simplifier la lecture). Autrement dit, ce qu'il vous est demandé ici est de créer une nouvelle version, simplifiée bien entendu, du logiciel Flex que vous êtes en train d'apprendre manipuler. Plus précisément, étant donnée n'importe quelle expression rationnelle en entrée, l'outil développé doit créer *in fine* l'analyseur lexical qui lui est associé, *i.e.* l'automate fini déterministe minimal permettant de reconnaître n'importe quel mot appartement au langage rationnel défini par l'expression rationnelle fournie.

Ainsi, vous devrez développer un Gal en C. Celui-ci prend une *expression rationnelle simple* en entrée, définie dans un fichier. Pour être tout à fait clair, une expression rationnelle simple est une expression rationnelle définie uniquement à partir de la répétition de Kleene (représentée par l'opérateur unaire `*`, *e.g* `a*` signifiant « l'expression `a` zéro ou plusieurs fois »), la concaténation (représentée par deux sous-expressions rationnelles qui se suivent, *e.g.* `ab` signifiant « l'expression `a` suivie de l'expression `b` ») et la disjonction (représentée par l'opérateur binaire `|`, *e.g.* `a | b` signifiant « l'expression `a` ou l'expression `b` », le « ou » étant ici à comprendre dans son sens exclusif). Le fichier contenant la définition de l'expression rationnelle sur laquelle le Gal doit être créé est de la forme :

> *La première ligne du fichier doit contenir la définition de l'expression rationnelle simple qui doit forcément être suivie d'un caractère* newline*, représenté ici par le carré noir.*
> `(a|(bc))*(ba(ca|ba))`
> ■

Depuis l'expression rationnelle e donnée en entrée, votre Gal devra construire séquentiellement :
- **l'arbre de syntaxe abstraite** $\mathcal{T}$ de e (construit de la gauche vers la droite comme cela a été vu en cours et en TD) ;
- **l'automate fini non déterministe** $\mathcal{N}$ associé à $\mathcal{T}$ contenant potentiellement des ϵ-transitions, par l'application de l'algorithme de McNaughton, Yamada et Thompson ;
- **l'automate fini déterministe** $\mathcal{D}$ associé à $\mathcal{N}$ par application de l'algorithme de construction de sa table de transition *Dtrans* vu en cours ;
- **l'automate fini déterministe minimal** $\mathcal{D}'$ de e.

Vous devrez également fournir **une représentation graphique** de chacun des « objets » $\mathcal{T}$, $\mathcal{N}$, $\mathcal{D}$ et $\mathcal{D}'$. La création de telles représentations devra être automatisée à l'aide d'un logiciel de représentation de graphe tel que graphviz.

À partir de l'automate $\mathcal{D}'$, votre Gal devra automatiquement engendrer le code d'un programme C dont l'exécutable obtenu par compilation avec `gcc` implantera **l'analyseur lexical** de e. En prenant un mot m en entrée, cet analyseur lexical devra alors répondre :

— "m `appartient au langage rationnel défini par` e." si m est reconnu par $\mathcal{D}'$;

— "m `n'appartient pas au langage rationnel défini par` e." sinon.

Information importante : l'implémentation logicielle que vous allez produire pour créer votre Gal doit se fonder uniquement sur les librairies de base du langage C (*e.g* `stdlib.h`, `stdio.h`, `unistd.h`, `string.h`...). Je vous invite donc à ne pas utiliser d'environnement de développement intégré (IDE) pour produire votre code mais plutôt d'utiliser un éditeur de code basique / standard, du type VI ou Emacs sous Linux par exemple. En effet, les IDE font parfois appel implicitement à des librairies non-standard, que je n'installerai pas sur ma machine au moment de tester vos projets. Au passage, bien que cela me semble évident, je préfère le dire : je m'attends bien sûr à ce que vous réalisiez ce travail sous un système d'exploitation de type UNIX. Votre code devra aussi intégrer un fichier Makefile permettant de la compiler simplement et intégralement par l'appel de la commande make dans le terminal.

3 Rendu (consignes)

Le code développé devra être envoyé par mail de sujet « Compilation – rendu » aux adresses `sylvain.sene@univ-amu.fr` et `sylvain.sene@lis-lab.fr` sous la forme d'une archive `projet_compil_<nom>_<prenom>.tar.gz`. Cette archive devra contenir un répertoire `compil_<nom>_<prenom>` contenant l'ensemble de vos fichiers composant votre Gal ainsi qu'un Makefile permettant de le compiler, dans un répertoire `src`, ainsi que votre rapport au format pdf de 10 pages maximum dans un répertoire `doc`.

La date de rendu (date butoir ferme) est le vendredi 23 avril 2022 à 23h59. Tout élément qui me parviendra après cette date ne sera pas pris en compte et la note donnée au projet sera donc 0/20.

Veillez bien à respecter les consignes données et la date. Je transmettrai par retour de mail un accusé de réception. En conséquence, si vous ne recevez pas cet accusé de réception alors que vous m'avez bien transmis votre projet, inquiétez-vous en me contactant directement par email.