Composition d'informatique n°5

Sujet unique (Durée : 4 heures)

L'utilisation de la calculatrice **n'est pas autorisée** pour cette épreuve.

⋆ ⋆ ⋆

Automates à pile

On s'intéresse dans ce sujet à un modèle de calcul appelé automate à pile, permettant de représenter la même classe de langages que les grammaires hors-contexte. La première partie concerne des préliminaires sur les grammaires. La deuxième partie propose l'implémentation d'une liste en C. La troisième partie présente les automates à pile et quelques premiers résultats. La quatrième partie étudie en particulier les langages déterministes, très utiles pour leur analyse syntaxique simplifiée. Enfin, la cinquième et dernière partie concerne la preuve de l'équivalence avec les langages algébriques.

Consignes

Les questions de programmation doivent être traitées en langage C uniquement. On supposera que les bibliothèques `stdlib.h`, `stdbool.h`, `assert.h` et `string.h` ont été chargées.

Lorsque le candidat écrira une fonction, il pourra faire appel à des fonctions définies dans les questions précédentes, même si elles n'ont pas été traitées. Il pourra également définir des fonctions auxiliaires, mais devra préciser leurs rôles ainsi que les types et significations de leurs arguments. Les candidats sont encouragés à expliquer les choix d'implémentation de leurs fonctions lorsque ceux-ci ne découlent pas directement des spécifications de l'énoncé. Si les paramètres d'une fonction à coder sont supposés vérifier certaines hypothèses, il ne sera pas utile dans l'écriture de cette fonction de tester si les hypothèses sont bien satisfaites.

On identifiera une même grandeur écrite dans deux polices de caractères différentes, en italique du point de vue mathématique (par exemple n) et en Computer Modern à chasse fixe du point de vue informatique (par exemple `n`).

Sans précision supplémentaire, lorsqu'une question demande la complexité d'une fonction, il s'agira de la complexité temporelle dans le pire des cas. La complexité sera exprimée sous la forme $\mathcal{O}(f(n,m))$ où n et m sont les tailles des arguments de la fonction, et f une expression la plus simple possible. Les calculs de complexité seront justifiés succinctement.

1 Préliminaires sur les grammaires

On pose $L_= = \{u \in \{a,b\}^* \mid |u|_a = |u|_b\}$.

Question 1 Montrer que $L_=$ n'est pas rationnel.

On considère la grammaire G_0 définie par les règles $S \to aSb \mid bSa \mid SS \mid \varepsilon$.

Question 2 Représenter graphiquement un arbres de dérivation du mot $u = abbaba$ pour la grammaire G_0.

Question 3 Montrer que $L_= = L(G_0)$.

Question 4 La grammaire G_0 est-elle ambiguë ? Si oui, le prouver et déterminer sans justifier une grammaire non ambiguë qui lui est faiblement équivalente. Si non, le prouver rigoureusement.

2 Listes en C

On s'intéresse à une structure de liste chaînée d'entiers en C. On implémente une liste par le type suivant :

```
struct List {
    int hd;
    struct List* tl;
};

typedef struct List list;
```

Pour la suite, le terme « une liste » désignera en fait un pointeur d'un objet `list`, c'est-à-dire qu'une liste est de type `list*`.

Question 5 Écrire une fonction `list* cons(int hd, list* tl)` qui prend en argument une tête et une queue et renvoie une liste formée de cette tête et cette queue.

Question 6 Écrire une fonction `void list_free(list* lst)` qui libère la mémoire utilisée par une liste.

Question 7 Écrire une fonction `list* list_copy(list* lst)` qui copie une liste sans liaison de données.

Question 8 Écrire une fonction `void concat(list* l1, list* l2)` qui prend en argument deux listes et modifie la liste `l1` pour qu'après modification, la nouvelle liste `l1` contient les éléments de la concaténation entre `l1` et `l2`. On renverra une erreur si `l1` est le pointeur `NULL`.

Question 9 Déterminer la complexité temporelle de la fonction `concat`.

3 Automates à pile

L'ensemble des langages reconnaissables étant inclus strictement dans l'ensemble des langages algébriques, il n'est en général pas possible de trouver un automate fini qui reconnaît un langage algébrique. On propose dans cette partie d'étendre le modèle des automates pour reconnaître ces langages.

Le principe d'un automate à pile est similaire à celui d'un automate fini : on lit un mot donné en entrée, on change d'états au cours de la lecture, et si une certaine condition est remplie après avoir lu le mot, il est considéré comme accepté. La différence fondamentale est qu'au cours de la lecture, l'automate a accès à une **pile** de symboles, de taille non bornée, permettant de diriger les transitions.

Pour effectuer une transition, l'idée est la suivante :

- on lit une lettre du mot, ou le mot vide ε (comme dans une ε-transition), l'état courant et le symbole en haut de la pile, qu'on dépile ;
- en fonction de ces valeurs, on change d'état et on empile éventuellement de nouveaux symboles dans la pile.

Formellement, un **automate à pile** A est un septuplet $(Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F)$ tel que :

- Q est un ensemble fini d'**états** ;
- Σ est un alphabet fini. On notera $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$;
- Γ est un alphabet fini appelé **alphabet de pile** ;
- Δ est une **fonction de transition** de $Q \times \Sigma_\varepsilon \times \Gamma$ dans $\mathcal{P}(Q \times \Gamma^*)$. Si $(q, \beta) \in \Delta(p, a, Z)$, on note simplement $p \xrightarrow{a, Z|\beta} q$;
- $q_0 \in Q$ est **l'état initial** ;
- $Z_0 \in \Gamma$ est le **symbole de pile initial** ;
- $F \subseteq Q$ est un ensemble d'**états finaux**.

On utilise une représentation graphique similaire à celle des automates finis pour représenter un automate à pile. La figure 1 représente un automate à pile A_0 pour $\Sigma = \{a, b\}$ et $\Gamma = \{A, Z_0\}$.

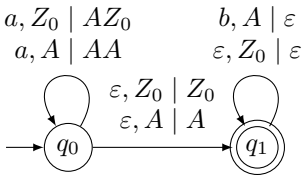

FIGURE 1 – L'automate à pile A_0.

Une **configuration** est un couple (q, γ) où $q \in Q$ est un état et $\gamma \in \Gamma^*$ représente le contenu de la pile (le haut de la pile étant à gauche de γ). Pour une transition $p \xrightarrow{a,Z|\beta} q$ et $\gamma \in \Gamma^*$, on dit qu'il existe une **transition de configuration** entre $(p, Z\gamma)$ et $(q, \beta\gamma)$ en lisant a. On notera $(p, Z\gamma) \xrightarrow{a} (q, \beta\gamma)$.

Un **calcul** d'un mot $u \in \Sigma^*$ est une suite de transitions de configuration $(p_0, \gamma_0) \xrightarrow{a_1} \dots \xrightarrow{a_k} (p_k, \gamma_k)$ telle que $u = a_1 \dots a_k$. On notera plus simplement $(p_0, \gamma_0) \xrightarrow{u}{}^* (q_k, \gamma_k)$.

Pour $u \in \Sigma^*$, s'il existe un calcul de u de la forme $(q_0, Z_0) \xrightarrow{u}{}^* (q, \gamma)$, on dit que u est accepté par A par :

- pile vide si $\gamma = \varepsilon$;
- état final si $q \in F$;
- pile vide et état final si $\gamma = \varepsilon$ et $q \in F$.

On note $L_P(A)$, $L_F(A)$ et $L_{PF}(A)$ l'ensemble des mots acceptés par A pour chacun des modes d'acceptation précédents.

Par exemple, $(q_0, Z_0) \xrightarrow{a} (q_0, AZ_0) \xrightarrow{a} (q_0, AAZ_0) \xrightarrow{\varepsilon} (q_1, AAZ_0) \xrightarrow{b} (q_1, AZ_0) \xrightarrow{b} (q_1, Z_0) \xrightarrow{\varepsilon} (q_1, \varepsilon)$ est un calcul acceptant le mot $u = aabb$ par pile vide et état final pour l'automate à pile A_0 de la figure 1. On peut montrer que $L_P(A_0) = L_{PF}(A_0) = \{a^n b^n \mid n \in \mathbb{N}\}$ et $L_F(A_0) = \{a^m b^n \mid m \geqslant n\}$.

Question 10 Donner, sans justifier, un calcul acceptant $aaab$ par état final pour l'automate A_0 de la figure 1.

Question 11 On considère l'automate A_1 de la figure 2 sur l'alphabet d'entrée $\Sigma = \{a, b\}$ et l'alphabet de pile $\Gamma = \{A, B, Z_0\}$.

Déterminer en justifiant succinctement les langages $L_P(A_1)$, $L_F(A_1)$ et $L_{PF}(A_1)$.

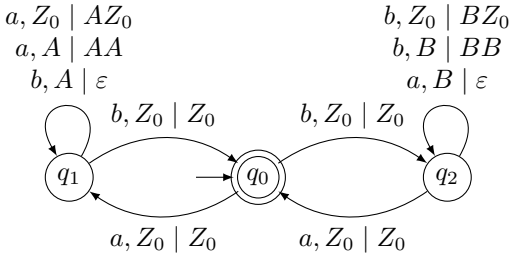

FIGURE 2 – L'automate à pile A_1.

Question 12 Représenter graphiquement sans justifier un automate à pile A_2 tel que $L_{PF}(A_2) = L_2 = \{u\overline{u} \mid u \in \{a, b\}^*\}$ où, si $u = a_1 \dots a_n \in \Sigma^*$, $\overline{u}$ est le mot miroir de u, $\overline{u} = a_n \dots a_1$.

Question 13 Soit $L \subseteq \Sigma^*$. Montrer qu'il y a équivalence entre les trois propriétés suivantes :

1. il existe un automate à pile A tel que $L_F(A) = L$;
2. il existe un automate à pile A tel que $L_P(A) = L$;
3. il existe un automate à pile A tel que $L_{PF}(A) = L$.

On donnera les constructions des automates à pile permettant de montrer les équivalences, et on justifiera succinctement leur correction.

4 Langages algébriques déterministes

4.1 Automate à pile déterministe

Soit $A = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F)$ un automate à pile. A est dit **déterministe** si et seulement si pour tout $q \in Q$, $a \in \Sigma$ et $Z \in \Gamma$:

$$|\Delta(q, a, Z)| + |\Delta(q, \varepsilon, Z)| \leqslant 1$$

Cela implique, en particulier, que pour tout mot u, il existe au plus un calcul de u.

Par exemple, l'automate à pile A_0 de la figure 1 n'est pas déterministe (car il y a deux transitions sortantes depuis l'état q_0 avec le symbole de haut de pile Z_0, l'une en lisant a, l'autre en lisant ε), et l'automate à pile A_1 de la figure 2 est déterministe.

Un langage L est dit **déterministe** s'il existe un automate à pile déterministe A tel que $L = L_F(A)$.

Question 14 Montrer que le langage $\{a^n b^n \mid n \in \mathbb{N}\}$ est déterministe.

Question 15 Montrer que tout langage rationnel est déterministe.

Soit $A = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F)$ un automate à pile déterministe. A est dit **complet** s'il peut lire tous les mots, c'est-à-dire si, pour $u \in \Sigma^*$, il existe une configuration (q, γ) telle que $(q_0, Z_0) \xrightarrow{u}{}^* (q, \gamma)$. En particulier, cela implique :

- la lecture d'un mot u ne peut pas emmener vers une configuration avec une pile vide ;
- pour tout $q \in Q$, $a \in \Sigma$ et $Z \in \Gamma$, $|\Delta(q, a, Z)| + |\Delta(q, \varepsilon, Z)| = 1$;
- il n'existe pas de cycle infini lisant ε.

Question 16 Montrer que si A est un automate à pile déterministe, alors il existe B un automate à pile déterministe complet tel que $L_F(A) = L_F(B)$.

On accepte une description informelle de la construction de l'automate B.

4.2 Implémentation

Dans toute cette partie, on suppose que les automates considérés sont complets. On suppose de plus que si q est un état final et $(q, \gamma) \xrightarrow{\varepsilon}{}^* (q', \gamma')$, alors q' est final.

On implémente un mot de Σ^* comme un élément de type `char*`, dont les caractères sont les 255 non nuls de la table ASCII. On admet qu'un caractère peut être utilisé comme un indice de tableau sans conversion. Un élément de Σ_ε sera représenté par un caractère différent de `'\0'` pour un élément de Σ et par le caractère `'\0'` pour ε.

On distinguera ainsi ε en tant qu'élément de Σ_ε (de type `char` et égal à `'\0'`) et ε comme élément de Σ^* (de type `char*` correspondant à un pointeur vers `'\0'`).

On implémente Γ comme l'ensemble d'entiers $\Gamma = [\![0, m-1]\!]$, et un mot de Γ^* comme une liste chaînée de ses lettres.

On implémente un automate à pile déterministe par le type `dpda` (*deterministic pushdown automaton*) suivant :

```
struct DPDA {
    int Q;
    int Gamma;
    int*** Dstate;
    list**** Dstack;
    bool* F;
};

typedef struct DPDA dpda
```

tel que si $A = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F)$ est un automate à pile déterministe représenté par un objet `A` de type `dpda`, alors :

- $Q = [\![0, n-1]\!]$, où $n = |Q| =$ `A.Q`, et $q_0 = 0$;
- $\Gamma = [\![0, m-1]\!]$, où $m = |\Gamma| =$ `A.Gamma`, et $Z_0 = 0$;
- `A.F` est un tableau de n booléens tel que pour $q \in Q$, `A.F[q]` vaut `true` si et seulement si $q \in F$;
- `A.Dstate` est un tableau de tableaux de tableaux d'entiers, de dimensions $n \times 256 \times m$ tel que pour $p \in Q$, $a \in \Sigma_\varepsilon$ et $Z \in \Gamma$, `A.Dstate[p][a][Z]` vaut q s'il existe une transition $p \xrightarrow{a,Z|\beta} q$ et -1 sinon ;
- `A.Dstack` est un tableau de tableaux de tableaux de , de dimensions $n \times 256 \times m$ tel que pour $p \in Q$, $a \in \Sigma_\varepsilon$ et $Z \in \Gamma$, `A.Dstack[p][a][Z]` vaut β s'il existe une transition $p \xrightarrow{a,Z|\beta} q$ et `NULL` sinon.

Par principe du déterminisme, on remarque que si `A.Dstate[p]['\0'][Z]` est différent de -1, alors `A.Dstate[p][a][Z]` vaut -1 pour tout $a \in \Sigma$.

Question 17 Écrire une fonction `bool delta(dpda A, int* q, list** gamma, char a)` qui prend en argument un automate à pile déterministe A, un pointeur vers un état q, un pointeur vers un mot non vide $\gamma \in \Gamma^*$ et $a \in \Sigma_\varepsilon$ et :

- s'il existe une transition de configuration $(q, \gamma) \xrightarrow{a} (q', \gamma')$, alors la fonction modifie les valeurs pointées par `q` et `gamma` en q' et γ' et renvoie `true` ;
- sinon, la fonction ne modifie rien et renvoie `false`.

Attention, la fonction ne devra pas modifier l'automate.

Pour $q \in Q$, $\gamma \in \Gamma^*$ et $u \in \Sigma^*$, on note $\Delta^*(q, u, \gamma)$ la configuration (q', γ') atteinte par le plus long calcul de u depuis (q, γ) (c'est-à-dire qu'on lit tous les ε possibles après la lecture de u).

Question 18 En déduire une fonction `int delta_star(dpda A, int q, list** gamma, char* u)` qui prend en argument un automate à pile déterministe A, un état q, un pointeur vers un mot non vide $\gamma \in \Gamma^*$ et un mot $u \in \Sigma^*$ et, si $\Delta^*(q, u, \gamma) = (q', \gamma')$, alors la fonction modifie γ en γ' et renvoie q'.

Question 19 En déduire une fonction `bool accepted(dpda A, char* u)` qui détermine si un mot u est accepté par état final par un automate à pile déterministe A.

On pose $\mu = \max\{|\beta|, (q, \beta) \in \Delta(Q, \Sigma_\varepsilon, \Gamma)\}$, le plus grand nombre de symboles qui peuvent être ajoutés à la pile lors d'une transition.

Question 20 On suppose que pour $q \in Q$ et $Z \in \Gamma$, $\Delta(q, \varepsilon, Z) = \emptyset$. Déterminer la complexité temporelle de la fonction `accepted` en fonction de $|u|$ et μ.

Question 21 Montrer qu'il existe un automate à pile déterministe complet à un seul état tel que $\mu = 2$, $|\Gamma| = N$ et tel qu'un calcul de ε peut être de longueur exponentielle en N.

4.3 Différences avec les automates à pile non déterministes

Contrairement aux automates à pile non déterministe, il n'y a pas équivalence au fait d'être accepté par un automate à pile déterministe pour les différents modes d'acceptation.

Question 22 Donner, en justifiant, un exemple de langage rationnel L tel qu'il n'existe aucun automate à pile déterministe A tel que $L = L_P(A)$.

On dit qu'un langage $L \subseteq \Sigma^*$ est **sans préfixes** si et seulement si aucun mot de L n'est préfixe d'un autre, c'est-à-dire si pour tout $u, v \in L^2$, $u \neq v$, alors u n'est pas un préfixe de v.

Question 23 Soit A un automate à pile déterministe. Montrer que $L_P(A)$ est un langage déterministe et sans préfixes.

Question 24 Réciproquement, soit L un langage déterministe sans préfixes. Montrer que L est accepté par pile vide par un certain automate déterministe.

On cherche à montrer qu'il existe des langages algébriques non déterministes. On s'intéresse pour cela au langage $L_2 = \{u\overline{u} \mid u \in \{a, b\}^*\}$. On suppose que L_2 est déterministe et on considère un automate à pile déterministe $A = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F)$ acceptant L_2.

Question 25 Montrer que A est nécessairement complet.

Pour $q \in Q$, $u \in \Sigma^*$ et $\gamma \in \Gamma^*$, on note $\Delta^*(q, u, \gamma)$ la configuration atteinte par le plus long calcul de u depuis (q, γ) (autrement dit, on lit tous les ε possibles à la fin du mot u).

Question 26 Montrer que pour tout $u \in \Sigma^*$, il existe $v_u \in \Sigma^*$ et $Z_u \in \Gamma$ tels que si $(q_u, Z_u\gamma) = \Delta^*(q_0, uv_u, Z_0)$, alors pour tout $w \in \Sigma^*$, tout calcul de w depuis $(q_u, Z_u\gamma)$ mène vers une configuration ayant γ comme suffixe (autrement dit, on ne vide jamais la pile en dessous de Z_u).

Question 27 En déduire qu'il existe deux mots $u, v \in \Sigma^*, u \neq v$ tels que pour tout $w \in \Sigma^*$, $\Delta^*(q_0, uw, Z_0)$ et $\Delta^*(q_0, vw, Z_0)$ sont dans le même état.

Question 28 En déduire que L_2 n'est pas déterministe.

Question 29 Montrer que $L_3 = \{uc\overline{u} \mid u \in \{a, b\}^*\}$, sur $\Sigma = \{a, b, c\}$, est déterministe.

5 Équivalence avec les grammaires hors-contexte

Dans cette partie, on souhaite montrer un équivalent du théorème de Kleene pour les langages algébriques, à savoir qu'un langage est algébrique si et seulement s'il est accepté par un automate à pile (quel que soit le mode d'acceptation, d'après la question 13).

Pour $G = (\Sigma, V, P, S)$ une grammaire hors-contexte, on pose $A_G = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F)$ l'automate à pile défini par :

- $Q = \{q_0, q_1, q_2\}$;
- $\Gamma = \{Z_0\} \cup \Sigma \cup V$;
- $F = \{q_2\}$;
- Δ contient les transitions suivantes :
  - $q_0 \xrightarrow{\varepsilon, Z_0 | SZ_0} q_1$;
  - pour $X \to \beta \in P$: $q_1 \xrightarrow{\varepsilon, X | \beta} q_1$;

* pour $a \in \Sigma$: $q_1 \xrightarrow{a,a|\varepsilon} q_1$;
* $q_1 \xrightarrow{\varepsilon,Z_0|\varepsilon} q_2$.

Question 30 Montrer que pour $X \in V$, $u \in \Sigma^*$, $\alpha \in (\Sigma \cup V)^*$, il y a équivalence entre :

1. $X \Rightarrow_g^* u\alpha$;
2. pour $\gamma \in \Gamma^*$, $(q_1, X\gamma) \xrightarrow{u}{}^* (q_1, \alpha\gamma)$.

Question 31 En déduire que $L(G) = L_{PF}(A_G)$.

Soit $A = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F)$ un automate à pile. On pose $G_A = (\Sigma, V, P, S)$ la grammaire hors-contexte définie par :

- $V = \{S\} \cup (Q \times \Gamma \times Q)$;
- P contient les règles de production suivantes :
  * pour $q \in Q$:
  $$S \to (q_0, Z_0, q) \in P$$
  * pour chaque transition $p \xrightarrow{a,Z|\beta} q$, avec $a \in \Sigma_\varepsilon$ et $\beta = Z_1 \ldots Z_k \in \Gamma^+$, alors pour $(p_1, p_2, \ldots, p_k) \in Q^k$:
  $$(p, Z, p_k) \to a(q, Z_1, p_1)(p_1, Z_2, p_2)\ldots(p_{k-1}, Z_k, p_k) \in P$$
  * pour chaque transition $p \xrightarrow{a,Z|\varepsilon} q$, avec $a \in \Sigma_\varepsilon$, alors pour $p_0 \in Q$:
  $$(p, Z, p_0) \to a \in P$$

Question 32 Montrer que pour $u \in \Sigma^*$, $(p, q) \in Q^2$ et $Z \in \Gamma$, il y a équivalence entre :

1. $(p, Z) \xrightarrow{u}{}^* (q, \varepsilon)$;
2. $(p, Z, q) \Rightarrow^* u$.

Question 33 En déduire que $L(G_A) = L_P(A)$.

⋆ ⋆ ⋆