Composition d'informatique n°5

Corrigé

⋆ ⋆ ⋆

Automates à pile

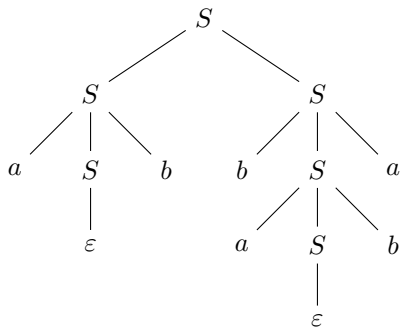
1 Préliminaires sur les grammaires

Question 1 Supposons que $L_=$ est rationnel et soit n sa longueur de pompage. On pose $u = a^n b^n \in L_=$. Par le lemme de pompage, u admet une décomposition $u = xyz$ telle que :

1. $|xy| \leqslant n$;
2. $|y| > 0$;
3. pour $k \in \mathbb{N}$, $xy^k z \in L_=$.

Avec les deux premières conditions, on en déduit que $y = a^i$, avec $i \in [\![1, n]\!]$. Mais alors, $xz = xy^0 z = a^{n-i} b^n \notin L_=$, ce qui est contradictoire avec la troisième condition. Par l'absurde, on en déduit que $L_=$ n'est pas rationnel.

Question 2 On obtient, par exemple :



Question 3 On montre ce résultat par double inclusion :

– $L_= \subseteq L(G_0)$: soit $u = a_1 \dots a_n \in L_=$. Montrons par récurrence sur n que $u \in L(G_0)$:
  * si $n = 0$, alors $u = \varepsilon$ et $S \Rightarrow \varepsilon$ est une dérivation de u dans G_0.
  * supposons le résultat établi pour tout mot de taille $< n$, $n \in \mathbb{N}$ fixé. Supposons $a_1 = a$ (on raisonne de même si $a_1 = b$). Soit alors $i \in [\![2, n]\!]$ l'indice minimal tel que $|a_1 \dots a_i|_a = |a_1 \dots a_i|_b$. Un tel indice existe bien, car $i = n$ vérifie cette égalité par hypothèse. De plus, $a_i = b$, sinon cela contredirait la minimalité de i. On peut donc écrire $u = avbw$, avec $v, w \in L_=$. Par hypothèse de récurrence, $S \Rightarrow^* v$ et $S \Rightarrow^* w$, donc $S \Rightarrow SS \Rightarrow aSbS \Rightarrow^* avbw = u$. On en déduit que $u \in L(G_0)$.

  On conclut par récurrence.

– $L(G_0) \subseteq L_=$: soit $\alpha \in \{a, b, S\}^*$ tel que $S \Rightarrow^n \alpha$. Montrons par récurrence sur n que $|\alpha|_a = |\alpha|_b$:
  * si $n = 0$, alors $\alpha = S$ et vérifie bien l'égalité ;
  * si on suppose $S \Rightarrow^{n-1} \alpha' \Rightarrow \alpha$, avec $|\alpha'|_a = |\alpha'|_b$, alors comme toutes les règles de la grammaire G_0 sont de la forme $S \to \beta$, avec $|\beta|_a = |\beta|_b$, on en déduit que $|\alpha|_a = |\alpha|_b$.

  On conclut par récurrence.

Question 4 La grammaire est ambiguë, car $S \Rightarrow_g \varepsilon$ et $S \Rightarrow_g SS \Rightarrow_g S \Rightarrow_g \varepsilon$ sont deux dérivations gauches de $\varepsilon \in L(G_0)$.

Attention, la grammaire définie par $S \to aSbS \mid bSaS \mid \varepsilon$ est ambiguë. Il faut passer par des mots de Dyck pour obtenir une grammaire non ambiguë :

- $S \to aAbS \mid bBaS \mid \varepsilon$;
- $A \to aAbA \mid \varepsilon$;
- $B \to bBaB \mid \varepsilon$.

Ici, A engendre les mots de Dyck, où a est une parenthèse ouvrante et b une parenthèse fermante, et B de même, en inversant les rôles de a et b.

2 Listes en C

Question 5 Pas de difficulté particulière ici.

```
list* cons(int hd, list* tl){
    list* lst = malloc(sizeof(list));
    lst->hd = hd;
    lst->tl = tl;
    return lst;
}
```

Question 6 Par une fonction récursive, par exemple.

```
void list_free(list* lst){
    if (lst != NULL){
        list_free(lst->tl);
        free(lst);
    }
}
```

Question 7 On copie récursivement la queue et on ajoute la tête.

```
list* list_copy(list* lst){
    if (lst == NULL) return NULL;
    return cons(lst->hd, list_copy(lst->tl));
}
```

Question 8 On commence par chercher le dernier maillon de la première liste, à qui on donne la deuxième liste comme queue.

```
void concat(list* l1, list* l2){
    assert (l1 != NULL);
    if (l1->tl == NULL){
        l1->tl = l2;
    } else {
        concat(l1->tl, l2);
    }
}
```

Question 9 Il y a un appel récursif par maillon de la liste `l1`, donc la complexité temporelle est linéaire en $|\ell_1|$.

3 Automates à pile

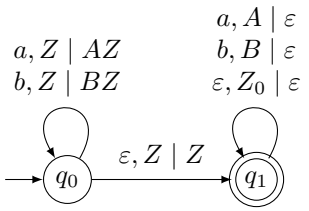
Question 10 Le calcul $(q_0, Z_0) \xrightarrow{a} (q_0, AZ_0) \xrightarrow{a} (q_0, AAZ_0) \xrightarrow{a} (q_0, AAAZ_0) \xrightarrow{\varepsilon} (q_1, AAAZ_0) \xrightarrow{b} (q_1, AAZ_0)$ est acceptant par état final.

Question 11 Remarquons les propriétés suivantes : si (q, α) est une configuration accessible depuis (q_0, Z_0) en ayant lu un mot u, alors :

- si $q = q_1$, alors $\alpha = A^k Z_0$, avec $k = |u|_a - |u|_b - 1$ (si on lit un a, on rajoute un A, si on lit un b, on supprime un A) ;
- symétriquement, si $q = q_2$, alors $\alpha = B^k Z_0$ avec $k = |u|_b - |u|_a - 1$;
- si $q = q_0$, alors $\alpha = Z_0$ et $u \in L_=$.

On en déduit que $L_P(A_1) = L_{PF}(A_1) = \emptyset$ (la pile ne se vide jamais) et $L_F(A_1) = L_=$.

Question 12 L'automate suivant convient :



On suit à peu près le même principe que pour le langage des $a^n b^n$, sauf qu'on empile le mot u sur la première moitié, et on dépile son miroir sur la deuxième moitié. Ici, l'alphabet de pile est $\{A, B, Z_0\}$, et le symbole Z désigne n'importe quelle symbole de pile (pour simplifier les notations).

Question 13 Montrons ce résultat par implications circulaires. Pour l'ensemble des questions, on considère $A = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F)$ un automate à pile.

- $1 \Rightarrow 2$: on ajoute un nouveau symbole de pile initial, qui permet d'éviter de vider la pile. On rajoute la possibilité d'atteindre un nouvel état puits depuis les états finaux pour vider la pile. Formellement, on pose $B = (Q', \Sigma, \Gamma', \Delta', q_0', Z_0', \emptyset)$ où :
  - $Q' = Q \cup \{q_0', s\}$;
  - $\Gamma' = \Gamma \cup \{Z_0'\}$;
  - Δ' contient les mêmes transitions que Δ, auxquelles on rajoute :
    - $q_0' \xrightarrow{\varepsilon, Z_0' | Z_0 Z_0'} q_0$: permet d'atteindre l'ancien état initial en rajoutant Z_0 au sommet de la pile ;
    - pour $q \in F$ et $Z \in \Gamma'$, $q \xrightarrow{\varepsilon, Z | \varepsilon} s$: permet d'atteindre l'état puits depuis n'importe quel état final ;
    - pour $Z \in \Gamma'$, $s \xrightarrow{\varepsilon, Z | \varepsilon} s$: permet de vider la pile.

  On a alors $L_F(A) = L_P(B)$.
- $2 \Rightarrow 3$: on transforme chaque état en état final. Formellement, on pose $B = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, Q)$. On a $L_P(A) = L_{PF}(B)$.
- $3 \Rightarrow 1$: on ajoute un nouveau symbole de pile initial, qui permet de repérer quand la pile est censée être vide. On ajoute alors une transition vers un nouvel état final. Formellement, on pose $B = (Q', \Sigma, \Gamma', \Delta', q_0', Z_0', \{s\})$ où :
  - $Q' = Q \cup \{q_0', s\}$;
  - $\Gamma' = \Gamma \cup \{Z_0'\}$;
  - Δ' contient les mêmes transitions que Δ, auxquelles on rajoute :
    - $q_0' \xrightarrow{\varepsilon, Z_0' | Z_0 Z_0'} q_0$: permet d'atteindre l'ancien état initial en rajoutant Z_0 au sommet de la pile ;

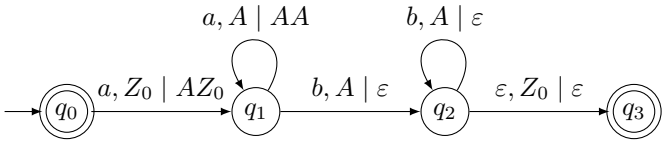
— pour $q \in F$, $q \xrightarrow{\varepsilon, Z_0' | \varepsilon} s$: permet d'atteindre l'état puits depuis n'importe quel état final, en supposant que la pile est censée être vide.

On a alors $L_{PF}(A) = L_F(B)$.

4 Langages algébriques déterministes

4.1 Automate à pile déterministe

Question 14 Il faut reprendre l'automate A_0 et le modifier pour obtenir un automate déterministe. On propose l'automate suivant :



L'idée est la suivante :

- si $u = \varepsilon$, alors u est reconnu, car l'état initial est final ;
- si $u = a^n b^n$ avec $n > 0$, alors on passe de q_0 à q_1 en lisant le premier a, on empile n fois un A, puis on passe de q_1 à q_2 en lisant le premier b, on dépile les A, puis on passe à q_3 quand on arrive au fond de la pile.

L'automate est bien déterministe.

Question 15 Soit L un langage rationnel et soit $A = (Q, \Sigma, \delta, q_0, F)$ un automate fini déterministe complet tel que $L = L(A)$. On définit $B = (Q, \Sigma, \{Z_0\}, \Delta, q_0, Z_0, F)$ où Δ contient toutes les transitions $q \xrightarrow{a, Z_0 | Z_0} \delta(q, a)$, pour $q \in Q$ et $a \in \Sigma$. Autrement dit, la pile est ignorée à chaque transition. Cet automate est un automate à pile déterministe et on a bien $L(A) = L_F(B)$.

Question 16 Soit $A = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F)$ un automate à pile déterministe. L'idée pour construire le nouvel automate est la suivante :

- on ajoute un nouveau symbole de pile initial, comme précédemment, pour éviter de vider la pile ;
- on ajoute trois nouveaux états : un nouvel état initial, un état final (acceptant) et un non final (rejetant) ;
- après l'état initial, si on lit le nouveau symbole initial, on va vers l'état rejetant ;
- pour $(q, Z) \in Q \times \Gamma$, s'il existe un cycle infini étiqueté par ε, depuis l'état q, avec Z au dessus de la pile, qui n'enlève jamais Z de la pile, alors on ajoute une transition vers l'état acceptant si ce cycle contient un état final, et vers l'état rejetant sinon. On dira que (q, Z) est une position cyclique, acceptante dans le premier cas, rejetante dans le second.

Formellement, on pose $B = (Q', \Sigma, \Gamma', \Delta, q_0', Z_0', F')$ où :

- $Q' = Q \cup \{q_0', q_a, q_r\}$;
- $\Gamma' = \Gamma \cup \{Z_0'\}$;
- $F' = \begin{cases} F \cup \{q_0, q_a\} & \text{si } q_0 \in F \\ F \cup \{q_a\} & \text{sinon} \end{cases}$;
- Δ' contient les transitions de la forme :
  - $q_0' \xrightarrow{\varepsilon, Z_0' | Z_0 Z_0'} q_0$;
  - pour $a \in \Sigma$ et $Z \in \Gamma'$: $q_a \xrightarrow{a, Z | Z} q_r$ et $q_r \xrightarrow{a, Z | Z} q_r$;
  - pour $a \in \Sigma$, $q \in Q$ et $Z \in \Gamma$ tels que $|\Delta(q, a, Z) \cup \Delta(q, \varepsilon, Z)| = 0$: $q \xrightarrow{a, Z | Z} q_r$;
  - pour $q \in Q$ et $a \in \Sigma$: $q \xrightarrow{a, Z_0' | Z_0'} q_r$;

* pour $(q, Z) \in Q \times \Gamma$ une position cyclique acceptante : $q \xrightarrow{\varepsilon, Z|Z} q_a$. On supprime alors l'ancienne transition $\Delta(q, \varepsilon, Z)$;
* pour $(q, Z) \in Q \times \Gamma$ une position cyclique rejetant : $q \xrightarrow{\varepsilon, Z|Z} q_r$. On supprime alors l'ancienne transition $\Delta(q, \varepsilon, Z)$.

4.2 Implémentation

Question 17 L'idée est la suivante :

- on commence par déterminer si la transition existe ou non, en fonction du symbole du haut de la pile ;
- si elle existe, on copie le nouveau mot de pile (pour éviter de modifier l'automate), puis on ajoute le reste de la pile ;
- on pense à libérer la mémoire occupée par le symbole en haut de la pile.

```
bool delta(dpda A, int* q, list** gamma, char a){
    assert (*q >= -1 && *gamma != NULL);
    list* tmp = *gamma;
    int Z = tmp->hd;
    int p = A.Dstate[*q][a][Z];
    if (p == -1) return false;
    *gamma = tmp->tl;
    free(tmp);
    list* beta = A.Dstack[*q][a][Z];
    *q = p;
    if (beta != NULL){
        beta = list_copy(beta);
        concat(beta, *gamma);
        *gamma = beta;
    }
    return true;
}
```

Question 18 On lit les transitions tant que possible. On s'arrête si on est arrivé à la fin du mot u, et qu'il n'y a plus d'ε-transitions à lire. L'indice i correspond à la lettre en cours de lecture dans u.

```
int delta_star(dpda A, int q, list** gamma, char* u){
    int i = 0;
    while (true){
        if (!delta(A, &q, gamma, '\0') && u[i] == '\0') return q;
        if (u[i] != '\0' && delta(A, &q, gamma, u[i])) i++;
    }
    return q;
}
```

Question 19 On crée une pile contenant initialement uniquement le symbole de pile initial, puis on lit le mot u et on vérifie si l'état atteint est final ou non. On pense à libérer la mémoire occupée par la pile.

```
bool accepted(dpda A, char* u){
    list* gamma = cons(0, NULL);
    int q = delta_star(A, 0, &gamma, u);
    list_free(gamma);
    return A.F[q];
}
```

Question 20 On remarque que la fonction `delta` est de complexité $\mathcal{O}(\mu)$ (copie de β et concaténation). On en déduit que la complexité de la fonction `delta_star` est en $\mathcal{O}(\mu \times |u|)$ (la boucle tant que est de taille $|u|$ car il n'y a pas de transition étiquetée par ε). C'est également la complexité de `accepted`, qui doit en plus vider la pile, de taille au plus $1 + |u| \times (\mu - 1)$.

Question 21 On propose l'automate $A = (\{q_0\}, \{a\}, \{Z_0, \ldots, Z_N\}, \Delta, q_0, Z_0, \{q_0\})$ tel que Δ contient les transitions suivantes (on n'indique pas l'état à chaque fois, puisqu'il est unique) :

- $\varepsilon, Z_0 \mid Z_2 Z_1$
- $a, Z_1 \mid Z_1$
- $\varepsilon, Z_N \mid \varepsilon$
- pour $i \in [\![2, N-1]\!]$, $\varepsilon, Z_i \mid Z_{i+1} Z_{i+1}$

L'idée est la suivante : on commence par ajouter $Z_2 Z_1$ sur la pile. Ensuite, chaque Z_i est transformé en deux exemplaires de Z_{i+1}. Finalement, les Z_N sont effacés sans modifications. Quand il ne reste que Z_1 sur la pile, on peut lire les lettres a (ces transitions ne sont là que pour garantir que l'automate est complet). Ainsi, la lecture de ε ajoutera 2 fois Z_3, 4 fois Z_4, …, 2^{N-2} fois Z_N. La longueur du calcul est bien exponentielle en $N = |\Gamma| - 1$.

4.3 Différences avec les automates à pile non déterministes

Question 22 Il suffit de poser $L = a^*$. En effet, comme $\varepsilon \in L$, alors il existe un calcul $(q_0, Z_0) \xrightarrow{\varepsilon}{}^* (q_f, \varepsilon)$, avec $q_f \in F$. Mais alors, la pile étant vide, il n'est pas possible de continuer un calcul en lisant a, qui n'est donc pas reconnu.

Question 23 Soit $u \in L_P(A)$ et v tel que u est un préfixe propre de v. Comme le calcul de u mène à une configuration de la forme (q, ε), il n'est pas possible de continuer le calcul de v. Ainsi, $v \notin L_P(A)$, qui est donc sans préfixes.

Pour montrer que $L_P(A)$ est déterministe, on applique la transformation vue en question 13 : on ajoute un nouvel état initial qui permet d'avoir un nouveau symbole de pile initial, un nouvel état qui devient l'unique état final, vers lequel on peut aller en dépilant le nouveau symbole de pile initial. Ces modifications ne change pas le caractère déterministe.

Question 24 L étant sans préfixe, on peut supprimer les transitions sortant des états finaux d'un automate à pile déterministe reconnaissant L par état final, sans changer le langage. On peut rajouter, pour chaque état final, des transitions qui vident la pile pour reconnaître le langage. Quitte à rajouter un nouveau symbole de pile initial, on peut éviter que la pile se vide dans un état non final.

Question 25 Supposons que A n'est pas complet. Soit alors $u \in \Sigma^*$ tel qu'il n'existe pas de calcul de u dans A. Alors il n'existe pas non plus de calcul de $u\overline{u}$ dans A, ce qui est absurde, car $u\overline{u} \in L(A)$.

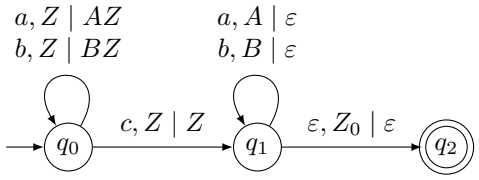
Question 26 L'automate A étant complet, on sait que la lecture de tout mot ne vide jamais la pile. Soit alors $v_u \in \Sigma^*$ tel que la pile de la configuration $\Delta^*(q_0, uv_u, Z_0)$ est de taille minimale. Nécessairement, cette pile est de la forme $Z_u\gamma$. De plus, par minimalité de la pile, les symboles de γ ne seront jamais dépilé si on lit un autre mot w après avoir lu uv_u.

Question 27 Il y a un nombre fini de couples $(q_u, Z_u) \in Q \times \Gamma$ possibles. Soit alors $u_1 \neq u_2$ tels que $q_{u_1} = q_{u_2}$ et $Z_{u_1} = Z_{u_2}$. On a alors $\Delta^*(q_0, u_1 v_{u_1}, Z_0) = (q_{u_1}, Z_{u_1}\gamma_{u_1})$ et $\Delta^*(q_0, u_2 v_{u_2}, Z_0) = (q_{u_2}, Z_{u_2}\gamma_{u_2})$. Mais comme le contenu de γ_{u_1} ou de γ_{u_2} n'est jamais lu si on continue la lecture d'un mot après avoir lu $u_1 v_{u_1}$ ou $u_2 v_{u_2}$, les configurations resteront dans le même état si on lit le même mot. On pose alors $u = u_1 v_{u_1}$ et $v = u_2 v_{u_2}$. Ces deux mots sont distincts et vérifient bien les conditions.

Question 28 Avec les notations de la question précédente, $\Delta^*(q_0, u\overline{u}, Z_0)$ et $\Delta^*(q_0, v\overline{u}, Z_0)$ sont censés être dans le même état, ce qui est absurde, car l'un doit être final (car $u\overline{u} \in L_2$) et l'autre non (car $v\overline{u} \notin L_2$). On

en déduit que L_2 n'est pas déterministe.

Question 29 Ce langage est reconnu par l'automate à pile déterministe suivant, qui est une variante de l'automate non déterministe reconnaissant L_2.



Comme précédemment, Z désigne n'importe quelle variable de $\Gamma = \{A, B, Z_0\}$. On a rajouté un troisième état pour garantir de reconnaître par état final.

5 Équivalence avec les grammaires hors-contexte

Question 30 $1 \Rightarrow 2$: par récurrence sur la taille n de la dérivation $X \Rightarrow_g^n u\alpha$:

- si $n = 0$, alors $u\alpha = X$, donc $u = \varepsilon$ et $\alpha = X$. On a bien $(q_1, X\gamma) \xrightarrow{\varepsilon}{}^* (q_1, X\gamma)$;
- supposons le résultat établi pour une dérivation de taille $< n$, $n \in \mathbb{N}$ fixé. Par principe de la dérivation gauche, il existe $u' \in \Sigma^*$, $Y \to \beta \in P$ et $\alpha' \in (\Sigma \cup V)^*$ tels que :

$$X \Rightarrow_g^{n-1} u'Y\alpha' \Rightarrow_g u'\beta\alpha' = u\alpha$$

Par hypothèse de récurrence, on a, pour $\gamma \in \Gamma^*$, $(q_1, X\gamma) \xrightarrow{u'}{}^* Y\alpha'\gamma$. On a alors :

$$(q_1, X\gamma) \xrightarrow{u'}{}^* Y\alpha'\gamma \xrightarrow{\varepsilon} \beta\alpha'\gamma \xrightarrow{u''}{}^* \alpha\gamma$$

où u'' est tel que $u = u'u''$.

On conclut par récurrence.

$2 \Rightarrow 1$: par récurrence sur la longueur n du calcul $(q_1, X\gamma) \xrightarrow{u}{}^n (q_1, \alpha\gamma)$:

- si $n = 0$, alors $u = \varepsilon$ et $\alpha = X$ et on a bien $X \Rightarrow_g^* X$;
- supposons le résultat établi pour un calcul de taille $< n$, $n \in \mathbb{N}$ fixé. Les transitions de q_1 à q_1 étant de deux formes différentes, distinguons :
  - si $(q_1, X\gamma) \xrightarrow{u}{}^{n-1} (q_1, Y\alpha'\gamma) \xrightarrow{\varepsilon} (q_1, \beta\alpha'\gamma)$, avec $Y \to \beta \in P$ et $\alpha = \beta\alpha'$, alors par hypothèse de récurrence, $X \Rightarrow_g^* uY\alpha' \Rightarrow_g u\beta\alpha' = u\alpha$;
  - si $(q_1, X\gamma) \xrightarrow{u'}{}^{n-1} (q_1, a\alpha\gamma) \xrightarrow{a} (q_1, \alpha\gamma)$, avec $u = u'a$, alors par hypothèse de récurrence, $X \Rightarrow_g^* u'a\alpha = u\alpha$.

On conclut par récurrence.

Question 31 Par double inclusion :

- si $u \in L(G)$, alors $S \Rightarrow_g^* u$. Par la question précédente, on a alors un calcul dans A_G de la forme :

$$(q_0, Z_0) \xrightarrow{\varepsilon} (q_1, SZ_0) \xrightarrow{u}{}^* (q_1, Z_0) \xrightarrow{\varepsilon} (q_2, \varepsilon)$$

donc $u \in L_{PF}(A_G)$.
- on traite le cas réciproque de même, en remarquant que si $u \in L_{PF}(A_G)$, alors la première et la dernière transition du calcul sont les mêmes que précédemment.

Question 32 $1 \Rightarrow 2$: par récurrence sur la longueur n du calcul :

- si $n = 1$, alors la transition lue est de la forme $p \xrightarrow{a,Z|\varepsilon} q$, donc $(p, Z, q) \to a \in P$. Comme $u = a$, on obtient le résultat ;
- supposons le résultat établi pour tout calcul de longueur $< n$, $n \in \mathbb{N}$ fixé. dans un calcul de u de longueur n de (p, Z) à (q, ε), la première transition est de la forme $p \xrightarrow{a,Z|Z_1\ldots Z_k} p_0$. Supposons que le calcul est de la forme :

$$(p, Z) \xrightarrow{a} (p_0, Z_1 \ldots Z_k) \overset{u_1}{\longrightarrow}{}^* (p_1, Z_2 \ldots Z_k) \overset{u_2}{\longrightarrow}{}^* \ldots \overset{u_{k-1}}{\longrightarrow}{}^* (p_{k-1}, Z_k) \overset{u_k}{\longrightarrow}{}^* (p_k = q, \varepsilon)$$

où p_i correspond à l'état atteint juste avant de dépiler Z_{i+1}.

Alors, par principe du calcul, $(p_{i-1}, Z_i) \overset{u_i}{\longrightarrow}{}^* (p_{i+1}, \varepsilon)$. Par hypothèse de récurrence, $(p_{i-1}, Z_i, p_{i+1}) \Rightarrow^* u_i$. On a alors la dérivation :

$$(p, Z, q) \Rightarrow a(p_0, Z_1, p_1)\ldots(p_{k-1}, Z_k, p_k) \Rightarrow^* au_1 \ldots u_k = u$$

On conclut par récurrence.

$2 \Rightarrow 1$: par récurrence sur la longueur n de la dérivation :

- si la dérivation est de longueur 1, alors $u = a \in \Sigma_\varepsilon$ et il existe une transition $p \xrightarrow{a,Z|\varepsilon} q$. On a bien $(p, Z) \overset{u}{\longrightarrow}{}^* (q, \varepsilon)$;
- sinon, la première dérivation immédiate est de la forme $(p, Z, p_k) \Rightarrow a(p_0, Z_1, p_1)(p_1, Z_2, p_2)\ldots(p_{k-1}, Z_k, p_k)$, avec $q = p_k$ et $k > 0$. Par le principe des dérivations, on peut écrire $u = au_1 \ldots u_k$, avec $(p_{i-1}, Z_i, p_i) \Rightarrow^* u_i$. Par hypothèse de récurrence, $(p_{i-1}, Z_i) \overset{u_i}{\longrightarrow}{}^* (p_i, \varepsilon)$. Il existe donc un calcul de u dans A de la forme :

$$(p, Z) \xrightarrow{a} (p_0, Z_1 \ldots Z_k) \overset{u_1}{\longrightarrow}{}^* (p_1, Z_2 \ldots Z_k) \overset{u_2}{\longrightarrow}{}^* \ldots \overset{u_k}{\longrightarrow}{}^* (q_k, \varepsilon)$$

On conclut par récurrence.

Question 33 Par double inclusion :

- soit $u \in L(G_A)$. Alors $S \Rightarrow^* u$. La première dérivation immédiate est nécessairement de la forme $S \Rightarrow (q_0, Z_0, q)$. On a donc $(q_0, Z_0, q) \Rightarrow^* u$. Par la question précédente, $(q_0, Z_0) \overset{u}{\longrightarrow}{}^* (q, \varepsilon)$. On en déduit que $u \in L_P(A)$;
- soit $u \in L_P(A)$. Alors $(q_0, Z_0) \overset{u}{\longrightarrow}{}^* (q, \varepsilon)$ pour un certain $q \in Q$. Il existe donc une dérivation de u de la forme $S \Rightarrow (q_0, Z_0, q) \Rightarrow^* u$, donc $u \in L(G_A)$.

$\star\star\star$