DS4-Durée 4h

25 janvier 2023

Ce sujet est composé de deux exercices et d'un problème.

La partie 3.3 et plus précisemment les questions 27 et 29 sont un peu plus délicates et nécessitent surtout d'avoir bien compris les notions définies. La partie 3.4 peut être en partie traitée même si la partie 3.3 n'a pas été bien réussie.

De nombreuses questions sont accessibles et doivent être soignées.

1 Jeu de Hex

Le jeu de Hex se joue sur un plateau $n\times$n à cases hexagonales. Tour à tour, chaque joueur marque une case de sa couleur (blanc ou noir). Le premier joueur qui arrive à faire un chemin de cases contigües reliant ses deux bords (gauche/droite pour le noir, haut/bas pour le blanc) gagne la partie. On suppose que c'est le joueur noir qui commence à jouer. La figure 1 Illustre une partie de Hex gagnée par le joueur noir.

1. Montrer qu'il n'est pas possible que la grille soit remplie avec les deux joueurs gagnants.
2. Montrer qu'il n'est pas possible que la grille soit remplie avec aucun joueur gagnant.
3. En déduire qu'il existe une stratégie gagnante pour le joueur noir (on pourra utiliser un argument de "vol" de stratégie).

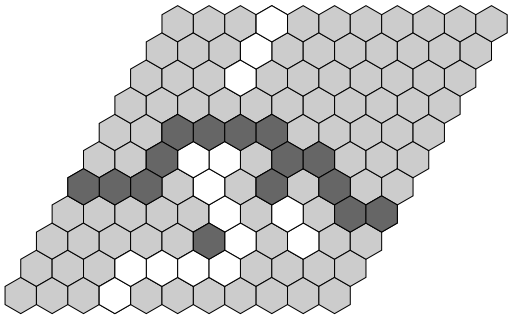

Figure 1 – Une partie de Hex sur un plateau 11×11.

2 Décidabilité

Pour chacun des problèmes A suivants, déterminer, en justifiant, s'il est décidable.

1. ARRÊT :
   - ⋆ **Instance** : un programme p et un argument x.
   - ⋆ **Question** : l'exécution de p sur x termine-t-elle ?
2. $\text{ARRÊT}_{\geq 10}$:
   - ⋆ **Instance** : un programme p.
   - ⋆ **Question** : le nombre d'arguments x pour lesquels l'exécution de p sur x se termine est-il supérieur ou égal à 10 ?
3. SOMME :
   - ⋆ **Instance** : une fonction OCAML $f : int \to int \to int \to bool$.
   - ⋆ **Question** : $f\ a\ b\ c$ renvoie-t-elle $true$ lorsque $a + b = c$?

3 Grammaires non contextuelles déterministes

On supposera dans ce problème que toutes les grammaires considérées ne font jamais apparaitre leur symbole initial dans un membre droit d'une règle.

1. Expliquer comment on peut se ramener à cela sans perte de généralité.

On supposera de plus qu'elles ne font intervenir que des non terminaux utiles c'est-à-dire qui apparaissent dans une dérivation du symbole initial à un mot généré.

3.1 Grammaire $\mathcal{G}_1$ et réductions

Considérons la grammaire suivante : $\mathcal{G}_1 = (\mathcal{V}_1, \Sigma_1, S, R_1)$ avec $\mathcal{V}_1 = \{S, R, T\}$, $\Sigma_1 = \{a, b\}$ et R_1 :

$$S \to R|T$$
$$R \to aRb|ab$$
$$T \to aTbb|abb$$

2. Montrer proprement que le langage engendré par $\mathcal{G}_1$ est inclus dans $\{a^n b^n : n \in \mathbb{N}^*\} \cup \{a^n b^{2n} : n \in \mathbb{N}^*\}$.
3. Démontrer l'inclusion réciproque afin de caractériser le langage engendré par $\mathcal{G}_1$.
4. Donner une dérivation gauche depuis le symbole initial pour le mot $aaabbb$.
5. Donner une dérivation droite depuis le symbole initial pour le mot $aabbbb$.
6. Ecrire un programme en C de signature `bool genere_1(char* m)` qui prend en entrée une chaîne de caractères quelconque et qui renvoie `true` si et seulement si celle-ci correspond à un mot généré par $\mathcal{G}_1$. On attend une complexité linéaire en la longueur de `m`. Il n'est pas attendu de s'appuyer sur les règles de la grammaire mais uniquement sur le langage engendré qui a été identifié.

   On définit maintenant la notion de réduction comme l'opération contraire à la dérivation. On dit que pour $\mathcal{G} = (\mathcal{V}, \Sigma, S, R)$, $u \in (\mathcal{V} \cup \Sigma)^*$ se réduit en $v \in (\mathcal{V} \cup \Sigma)^*$ ssi v se dérive en u. On note cette réduction $u \hookrightarrow^* v$.
   Une réduction directe se notera $u \hookrightarrow v$ et correspondra à $v \Rightarrow u$.
   Une réduction de u est une réduction de u jusqu'au symbole initial.
   On dit qu'une réduction de u est gauche ssi, à chaque étape, on réduit le facteur le plus à gauche possible.

7. Donner une réduction gauche pour le mot $aabbbb$ dans $\mathcal{G}_1$.
8. Montrer qu'une réduction gauche correspond à une dérivation droite pour toute grammaire.
9. Montrer que si une grammaire est non ambigüe alors tout mot engendré par celle-ci admet une unique réduction gauche.
10. Montrer que si on a $u \hookrightarrow v$ alors il existe $x, h, y \in (\mathcal{V} \cup \Sigma)^*$ et une règle $T \to h \in R$ tels que $u = xhy$ et $v = uTy$. On dit que h est un facteur réductible de u.
11. Soit $w = u_1 \hookrightarrow u_2 \hookrightarrow u_3 \ldots \hookrightarrow u_k$ une réduction gauche avec $w \in \Sigma^*$. On a alors pour chaque u_i l'existence d'une factorisation sous la forme $u_i = xhy$ et d'une règle de la forme $T \to h$ tels que $u_{i+1} = xTy$. Justifier que y est composé uniquement de symboles terminaux.

    On appelle mot valide un mot $m \in (\mathcal{V} \cup \Sigma)^*$ pour lequel il existe $u \in \mathcal{L}(\mathcal{G})$ tel que m apparait dans une réduction gauche de u.
12. Dire (et justifier) si les mots suivants sont valides pour la grammaire $\mathcal{G}_1$:
    (a) $aaRbb$
    (b) $aTbb$
    (c) $aaRbbbbb$
    (d) $aaabbbbbb$
13. Pour tout mot valide m, on appelle facteur réductible gauche, un facteur réductible de m situé le plus à gauche possible tel que si on lui applique la réduction on obtient un mot valide.
    Pour chacun des mots valides identifiés dans la question précédente, donnez son facteur réductible gauche.
14. Une grammaire est dite déterministe si pour tout mot m valide de $(\mathcal{V} \cup \Sigma)^*$, il existe une unique réduction directe gauche vers un mot valide de règle $T \to h$ avec $m = xhy$ et que pour tout $y' \in \Sigma^*$, cette même réduction est l'unique réduction gauche de $m' = xhy'$ vers un mot valide quand m' est valide.
    Justifier à l'aide des mots $aaabbb$ et $aaabbbbbb$ que $\mathcal{G}_1$ n'est pas déterministe.

3.2 Grammaire $\mathcal{G}_2$

On définit $\mathcal{G}_2 = (\mathcal{V}_2, \Sigma_2, S_2, R_2)$ avec $\mathcal{V}_2 = \{S_2, E, T\}$, $\Sigma_2 = \{a, +, *\}$ et R_2 :

$$S_2 \to E$$
$$E \to E + T | T$$
$$T \to T * a | a$$

15. Est ce que le mot $a * a + a * a$ est généré par $\mathcal{G}_2$? Si oui, donner un arbre de dérivation pour ce mot et une dérivation gauche.
16. Est ce que le mot $a + a * a + a$ est généré par $\mathcal{G}_2$? Si oui, donner un arbre de dérivation pour ce mot et une dérivation droite.
17. On considère la grammaire $\mathcal{G}_2'$ ayant les mêmes caractéristiques que $\mathcal{G}_2$ mais de symbole initial T. Donner (sans justification) une expression régulière décrivant le langage engendré par $\mathcal{G}_2'$.
18. On considère la grammaire $\mathcal{G}_2''$ ayant les mêmes caractéristiques que $\mathcal{G}_2$ mais de symbole initial E. Donner (sans justification) une expression régulière décrivant le langage engendré par $\mathcal{G}_2''$.
19. En déduire une description du langage engendré par $\mathcal{G}_2$ et justifier que cette grammaire n'est pas ambigüe. Montrer, à l'aide d'un shéma, la forme d'un arbre de dérivation de cette grammaire.
20. Ecrire un programme en C de signature `bool genere_2(char* m)` qui prend en entrée une chaîne de caractères quelconque et qui renvoie `true` si et seulement si celle-ci correspond à un mot généré par $\mathcal{G}_2$. On attend une complexité linéaire en la longueur de `m`. Il n'est pas attendu de s'appuyer sur les règles de la grammaire mais uniquement sur le langage engendré qui a été identifié.
21. Donner une réduction gauche pour le mpt $a + a + a * a * a + a * a$.

3.3 L'automate DK

On va construire à partir d'une grammaire $\mathcal{G} = (\mathcal{V}, \Sigma, S, R)$ un automate non déterministe appelé $DK(\mathcal{G})$.

22. (question indépendante du reste du sujet) On souhaite construire un automate fini non déterministe J sur l'alphabet $\mathcal{V} \cup \Sigma$ qui reconnait le langage des mots dont un suffixe est le membre droit d'une règle de $\mathcal{G}$. Expliquer comment construire J dans le cas général et illustrer votre méthode en construisant l'automate associé à la grammaire $\mathcal{G}_1$.

    On va construire un automate fini non déterministe K sur l'alphabet $\mathcal{V} \cup \Sigma$ de la manière suivante :
    Pour chaque règle de la forme $V \to u_1 u_2 \dots u_k$ on crée $k+1$ états de la forme $V \to \bullet u_1 u_2 \dots u_k$, $V \to u_1 \bullet u_2 \dots u_k$, $V \to u_1 u_2 \bullet u_3 \dots u_k \dots V \to u_1 u_2 \dots u_{k-1} \bullet u_k$ et $V \to u_1 u_2 \dots u_k \bullet$. Ici chaque u_i est un symbole de $\mathcal{V}$ ou de Σ et donc une lettre du membre droit de la règle considérée.
    On crée une transition étiquetée par le symbole u_i pour chaque état de la forme $V \to u_1 \dots u_{i-1} \bullet u_i \dots u_k$ vers $V \to u_1 \dots u_i \bullet u_{i+1} \dots u_k$ ainsi qu'une ϵ-transition d'un état de la forme $V \to u \bullet Cv$ vers $C \to \bullet r$ pour chaque non terminal C et chaque règle associée $C \to r$.
    Les états initiaux seront tous les états de la forme $S \to \bullet r$ pour chaque règle avec S comme membre gauche.
    Les états terminaux seront les états de la forme $V \to u \bullet$ pour toute règle $V \to u$.

23. Construire l'automate DK($\mathcal{G}_2$).
24. Déterminiser l'automate obtenu à la question précédente.
25. Considérons l'automate DK($\mathcal{G}$) avec $\mathcal{G}$ quelconque. Supposons qu'à la lecture d'un mot z, il existe un chemin menant d'un état initial à l'état $T \to u \bullet v$.
    Considérons $S_1, \dots S_1, S_2, \dots, S_2, S_3, \dots, S_3, \dots, S_k \dots S_k$ la suite des symboles terminaux qui sont des membres gauches des états sur le chemin parcouru à la lecture de z. Ainsi, le chemin parcourt des états de membre gauche S_1 où le symbole $\bullet$ avance dans le membre droit puis suit une ϵ-transition qui lui permet de passer dans un état ayant pour membre gauche S_2 etc... Pour chaque S_i, on aura un dernier état dans le chemin l'ayant pour membre gauche avant de changer de membre gauche et on note cet état $S_i \to u_i \bullet S_{i+1} v_i$. On remarque que $S_k = T$.

    (a) En reprenant les notations introduites ici, exprimer z en fonction de u, de v, des u_i et des v_i. On remarquera que u est un suffixe de z.

    (b) En déduire qu'il existe $y \in (\mathcal{V} \cup \Sigma)^*$ tel que le mot zvy est dérivable depuis S et peut se réduire en xTy où $z = xu$.

    (c) En déduire l'existence de $y' \in \Sigma^*$ tel que $xuvy'$ soit un mot valide de facteur réductible uv. Justifier que uv est un facteur réductible gauche (on pourra s'appuyer sur un dessin de l'arbre de dérivation de $xuvy$).

26. Considérons maintenant un mot $m = xuvy$ valide de facteur réductible gauche uv associé à la règle $T \to uv$. Ici $x \in (\mathcal{V} \cup \Sigma)^*$ alors que $y \in \Sigma^*$ d'après la question 11.

    (a) Représenter l'arbre de dérivation de m en faisant apparaitre clairement le symbole initial S ainsi que le symbole T.

    Considérons les symboles non terminaux $S = S_0, S_1, \dots S_k = T$ sur le chemin de S à T dans cet arbre.

    (b) Justifier que tout symbole non terminal apparaissant dans un sous arbre gauche de l'un des sous arbres enraciné en l'un de S_i avec $i \leq k-1$ est un facteur du préfixe x, autrement dit n'a pas été dérivé.

    Ainsi chaque S_i dispose dans la grammaire d'une règle de la forme $S_i \to u_i S_{i+1} v_i$ et u_i est un facteur de x mais v_i n'est pas nécéssairement un facteur de y. On a même $x = u_0 \dots u_{k-1}$.

    (c) Montrer qu'à la lecture du mot $z = xu$ l'automate $DK(\mathcal{G})$ peut se retrouver dans l'état $T \to u \bullet v$.

27. Montrer que sur l'entrée z, l'automate $DK(\mathcal{G})$ peut se retrouver dans l'état $T \to h\bullet$ ssi $z = xh$ avec h un facteur réductible gauche associé à la règle $T \to h$ d'un mot valide de la forme xhy avec $x \in (\mathcal{V} \cup \Sigma)^*$ et $y \in \Sigma^*$.

28. Considérons DDK l'automate des parties de $DK(\mathcal{G})$. Supposons qu'il existe un état acceptant q de DDK contenant un état de $DK(\mathcal{G})$ de la forme $B \to u \bullet av$ avec a un symbole terminal. Soit z un mot menant à cet état.

    (a) Justifier que q contient un état de la forme $T \to h\bullet$. En déduire que z est de la forme xh avec l'existence d'un mot valide de la forme xhy dont h est un facteur réductible gauche associé à la règle $T \to h$.

    (b) Montrer qu'il existe $x' \in (\mathcal{V} \cup \Sigma)^*$ tel que $z = x'u$ et qu'il existe un mot valide de la forme $x'uavy'$ (avec $y' \in \Sigma^*$) dont uav est un facteur réductible gauche associé à la règle $B \to uav$.

    (c) Montrer que $xhavy' = zavy'$ peut être réduit en utilisant une règle différente de celle utilisée pour $xhy = zy$. Pourquoi cela ne suffit pas à montrer que $\mathcal{G}_2$ n'est pas déterministe (on portera une attention précise à la définition de déterministe) ?

    (d) Montrer que la grammaire n'est pas déterministe.

29. Montrer que $\mathcal{G}_2$ n'est pas déterministe.

3.4 Grammaire $\mathcal{G}_3$ (cette partie est beaucoup plus simple que la précédente.)

$\mathcal{G}_3 = (\mathcal{V}_3, \Sigma_3, S_3, R_3)$ avec $\mathcal{V}_3 = \{S_3, T\}$, $\Sigma_3 = \{(,), \bot\}$ et R_3 :

$$S_3 \to T\bot$$
$$T \to T(T)|\epsilon$$

30. Décrire le langage engendré par $\mathcal{G}_3$ (pas de justifications attendues).

31. Ecrire un programme en C de signature `bool genere_3(char* m)` qui prend en entrée une chaîne de caractères quelconque et qui renvoie `true` si et seulement si celle-ci correspond à un mot généré par $\mathcal{G}_3$. On attend une complexité linéaire en la longueur de la liste. Il n'est pas attendu de s'appuyer sur les règles de la grammaire mais uniquement sur le langage engendré qui a été identifié.

32. Construire l'automate $DK(\mathcal{G}_3)$ ainsi que son déterminisé.

33. On admet que si pour tout état acceptant de l'automate des parties de $DDK(\mathcal{G})$ d'une grammaire $\mathcal{G}$ il n'y a qu'une seule règle (terminée) de la forme $T \to h\bullet$ et aucune règle de la forme $B \to u\bullet av$ où a est un symbole terminal, alors $\mathcal{G}$ est déterministe. Montrer que la grammaire $\mathcal{G}_2$ est déterministe.