Préparation aux oraux CCINP n°1

Logique & graphes

Définition. Étant donnée une formule φ de la logique proportionnelle, on dit que :
- c'est un *littéral* si $\varphi = p$ ou $\varphi = \neg p$, pour $p \in \mathscr{P}$;
- c'est une *clause* si $\varphi = \ell_1 \vee \cdots \vee \ell_n$, où les ℓ_i sont des littéraux ;
- c'est une *FNC* si $\varphi = c_1 \wedge \cdots \wedge c_m$, où les c_j sont des clauses ;
- c'est une *n-FNC* si φ est une FNC et que chaque clause de φ contient au plus n littéraux.

Q1. Parmi les formules suivantes, lesquelles sont des FNC ? Pour toutes les formules sous FNC, lesquelles sont des n-FNC ? Quelle sont les valeurs de n ? Quel est le nombre de clauses de ces formules ? Finalement, lesquelles sont satisfiable ?

(1) $\psi_1 = (x_0^{1} \vee x_1^{2} \vee x_2^{3}) \wedge (\neg x_0^{1} \vee \neg x_1^{2} \vee \neg x_2^{3})$ 3-FNC avec 2 clauses

(2) ~~$\psi_2 = (x_0 \leftrightarrow x_1) \vee (x_2 \leftrightarrow x_3) \vee (x_4 \leftrightarrow x_5)$~~

(3) ~~$\psi_3 = (x_0 \wedge x_1 \wedge x_2) \rightarrow (\neg x_0 \vee \neg x_1 \vee \neg x_2)$~~

(4) $\psi_4 = (x_0^{1} \vee x_1^{2}) \wedge (\neg x_0^{1} \vee x_2^{2}) \wedge (\neg x_1^{1} \vee \neg x_2^{2})$ 2-FNC avec 3 clauses

(5) ~~$\psi_5 = (x_0 \wedge \neg x_1) \vee (x_2 \leftrightarrow x_3) \vee (x_4 \leftrightarrow \neg x_5)$~~

(6) ~~$\psi_6 = (x_0 \wedge \neg x_1 \wedge x_2) \vee (x_0 \wedge x_1 \wedge \neg x_2) \vee (\neg x_0 \wedge x_1 \wedge x_2)$~~

ψ_1 est satisfiable : x_0 à VRAI ; x_1 à Faux.

ψ_4 est satisfiable : x_0 à VRAI ; x_1 à Faux ; x_2 à VRAI

Q2. Mettre sous forme FNC la formule $\varphi = (x_0 \rightarrow (x_1 \wedge x_2)) \wedge \neg(x_1 \vee x_2)$.

Méthode 1 : Equivalence

$\varphi \equiv (x_0 \rightarrow (x_1 \wedge x_2)) \wedge \neg x_1 \wedge \neg x_2$ avec DE MORGAN

$\equiv (\neg x_0 \vee (x_1 \wedge x_2)) \wedge \neg x_1 \wedge \neg x_2$ avec $a \rightarrow b \equiv b \vee \neg a$

$\equiv (\neg x_0 \vee x_1) \wedge (\neg x_0 \vee x_2) \wedge \neg x_1 \wedge \neg x_2$ avec distributivité

$\equiv \neg x_0 \wedge \neg x_1 \wedge \neg x_2$.

Méthode 2 Tableau de Karnaugh

$x_1 x_2$ \ x_0	V	F
VV	F	F
VF	F	F
FF	F	V ← avec $x_0 \rightarrow$ F, $x_1 \rightarrow$ F, $x_2 \rightarrow$ F
FV	F	F

D'où $\varphi \equiv \neg x_0 \wedge \neg x_1 \wedge \neg x_2$, qui est une FNC.

On rappelle la définition du problème n-SAT, où $n \in \mathbb{N}$.

n-SAT :	**Entrée.** Une formule φ sous forme n-FNC **Sortie.** La formule φ est-elle satisfiable ?

Q3. Pour quelles valeurs de n le problème n-SAT est-il **NP**-complet ? Justifier dans les grandes lignes.

- n-SAT $\in$ NP (certificat = la valuation)
- 1-SAT est solvable par un gloutон.
- 2-SAT est analogue à un problème de graphe (CFC d'un graphe défini à partir d'une formule)
- 3-SAT se réduit à SAT $\Rightarrow$ NP-difficile
  On considère une variable représentant chaque sous-formule.
- $(n+1)$-SAT se réduit à n-SAT avec $n \geq 3$.

$\cdots \geq_p (n+1)\text{-SAT} \geq_p n\text{-SAT} \geq_p (n-1)\text{-SAT} \geq_p \cdots \geq_p 3\text{-SAT}$

n-SAT est NP difficile ssi $n \geq 3$.

Définition. On dit d'un graphe (non orienté) qu'il est *coloriable* si on peut colorier tous les sommets du graphe tels que deux sommets voisins sont de couleurs différentes. Un graphe est dit *k-coloriable* s'il est coloriable avec k couleurs.

Q4. En considérant le graphe ci-dessous, quelle est la valeur minimale de $k > 0$ telle que le graphe soit k-coloriable ?

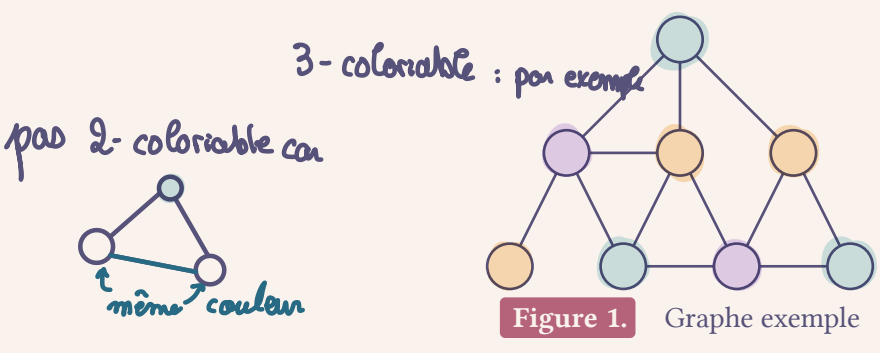


Figure 1. Graphe exemple

On considère un ensemble de couleurs $\mathscr{C} = [\![1, k]\!]$, un graphe $G = (S, A)$ avec $S = [\![1, N]\!]$. On crée les littéraux $p_{i,c}$ pour chaque sommet $i \in S$ et chaque couleur $c \in \mathscr{C}$.

Q5. (1) Pour i et j deux sommets voisins et une couleur c, créer une clause qui traduit le fait que les sommets i et j ne peuvent pas être de la même couleur $c \in \mathscr{C}$.
(2) Créer une 2-FNC qui traduit le fait que deux sommets i et j ne peuvent pas être de la même couleur.
(3) Créer une FNC qui traduit le fait que deux sommets voisins dans le graphe $G = (S, A)$ ne peuvent pas être de la même couleur.

(1) $\neg p_{ic} \vee \neg p_{jc}$ 

(2) $\bigwedge_{c \in \mathscr{C}} (\neg p_{ic} \vee \neg p_{jc})$ → 2-FNC

(3) $\psi = \bigwedge_{(i,j) \in A} \bigwedge_{c \in \mathscr{C}} (\neg p_{ic} \vee \neg p_{jc})$ → toujours une 2-FNC

Q6. On souhaite traduire le fait qu'un graphe est k-coloriable. Pour cela, il est nécessaire que les sommets voisins ont des couleurs différentes, et que tous les sommets soient coloriés.
(1) Traduire cette deuxième condition pour le graphe G sous la forme d'une FNC.
(2) Construire une FNC qui traduit le fait que G est k-coloriable.

(1) $\varphi = \bigwedge_{i \in S} \left(\left(\bigvee_{c \in \mathscr{C}} p_{ic} \right) \wedge \bigwedge_{c \in \mathscr{C}} \bigwedge_{c' \in \mathscr{C}} (\neg p_{ic} \vee \neg p_{ic'}) \right)$

le sommet i est colorié par une unique couleur

(2) $\Phi = \varphi \wedge \psi$. C'est une k-FNC

On définit le problème ci-dessous :

k-COLORATION :	**Entrée.** Un graphe $G = (S, A)$ non orienté **Sortie.** Le graphe G est-il k-coloriable ?

Q7. Montrer que k-COLORATION se réduit au problème k-SAT.

Etant donné entrée de k-coloration $G = (S, A)$ un graphe, on crée en temps polynomial une formule Φ sous k-FNC entrée de k-SAT. vérifiant

G est k-coloriable $\Leftrightarrow$ φ est satisfiable.

il existe un coloriage avec k-couleurs

il existe une valuation rendant φ vraie.

D'où la réduction.

Q8. Montrer que k-COLORATION est dans **NP**. Pour quelles valeurs de k le problème k-COLORATION est-il **NP**-complet ?

- Il suffit de parcourir chaque arrête pour vérifier la coloration. (certificat = liste de paires (sommet, couleur)).
- k-coloration est NP-complet ssi $k \geq 3$.

Préparation aux oraux CCINP n°2

Polynômes

Soient P et Q deux polynômes de degré inférieur ou égal à $2n-1$. Soient P_0, Q_0, P_1, et Q_1 quatre polynômes de degré inférieurs ou égaux à $n-1$ tels que

$$P = P_0 + X^n P_1 \quad \text{et} \quad Q = Q_0 + X^n Q_1.$$

On remarque que 1 multiplication (2n) ⟿ 4 multiplications (n)

$$\begin{aligned} P \times Q &= P_0Q_0 + X^n(P_1Q_0 + P_0Q_1) + X^{2n}P_1Q_1 \\ &= P_0Q_0 + X^n((P_0+P_1)(Q_0+Q_1) - P_0Q_0 - P_1Q_1) + X^{2n}P_1Q_1 \end{aligned}$$

1 multiplication (2n) ⟿ 3 multiplications (n)

Q1. En déduire un algorithme « *diviser pour régner* » calculant $P \times Q$ utilisant les fonctions :

- degré(P) donne le degré de P ;
- ajout(P,Q) (resp. soustrait(P,Q)) calcule $P+Q$ (resp. $P-Q$) ;
- décale(P,n) calcule $X^n \times P$;
- découpe(P,n) calcule P_0 et P_1 de degrés inférieurs ou égaux à $n-1$ tels que l'égalité polynomiale $P = P_0 + X^n P_1$ soit vérifiée.
- produit_degré_1(P,Q) calcule $P \times Q$ lorsque P et Q sont de degrés inférieurs ou égaux à 1.

On pourra supposer P et Q de même degré.

Entrée : $P, Q \in \mathbb{R}[X]$ de même degré.
Sortie : $P \times Q$.
$n \leftarrow \deg P$ c'est aussi deg Q
Si $n \leq 1$ alors Retourner produit_deg_1(P,Q)
$n' \leftarrow \lfloor n/2 \rfloor$
$P_0, P_1 \leftarrow$ découpe(P, n')
$Q_0, Q_1 \leftarrow$ découpe(Q, n')
On calcule P_0Q_0, P_1Q_1 et $\underbrace{(P_0+P_1)(Q_0+Q_1)}_{R}$. On simplifie la compréhension comme ça.
Renvoyer $P_0Q_0 + X^{n'}(R - P_0Q_0 - P_1Q_1) + X^{2n'}P_1Q_1$
décale$(R - P_0Q_0 - P_1Q_1, n')$ décale$(P_1Q_1, 2n')$

Q2. On note $C(n)$ le coût de cet algorithme lorsque P et Q sont représentés par des listes de taille n.

(1) Montrer que $C(n) \leq 3C\left(\frac{n}{2}\right) + Kn$, où K est un entier indépendant des entrées et de n.

(2) On étudie le cas où $n = 2^p$, pour $p \in \mathbb{N}$. Chercher α minimal tel que $C(n) = O(n^\alpha)$. (On se contentera d'exhiber un tel α sans montrer son caractère minimal.)

(3) Expliquer comment faire dans le cas où n est quelconque.

(1) $C(n) \leq 3C\left(\frac{n}{2}\right) + Kn$
↑ trois multiplications de degré $n/2$

Kn = découpage ×2 ($2n'$ avec $n' = n/2$, donc $= n$)
+ degré ×1 (n)
+ somme/soustraction ×6 ($6n$ (on approxime))
+ décale ×2 ($n' + 2n' = \frac{3}{2}n$)

D'où $K = 9{,}5$.
↳ ne dépend pas des entrées.

(2) On pose $v_p = C(2^p)$.

$v_{p+1} \leq 3v_p + 2^{p+1} \times K$

D'où, $v_p \leq 3^p v_0 + K\frac{2^{p+1}-1}{2-1}$

Or $2^p = O(3^p)$ d'où, $v_p = O(3^p)$

et donc $C(n) = O(3^{\log_2 n})$
$= O(2^{\log_2 3 \log_2 n})$
$= O(n^{\log_2 3})$ avec n une puissance de 2.
Ou $\alpha = \log_2 3$

(3) n quelconque : on montre que $C(n)$ est croissante. C'est le cas.

D'où, $C(n) = O(n^\alpha)$ pour tout $n \in \mathbb{N}$.

Q3. On s'intéresse maintenant au calcul de Q^k, pour un polynôme Q de degré n donné.

(1) Remarquons que $Q^{k+1} = Q \times Q^k$ et que $Q^0 = 1$. En utilisant cette remarque et l'algorithme de Q1, écrire un algorithme permettant de calculer Q^k.

(2) Quelle est la complexité de cet algorithme ?

(3) En utilisant une stratégie « *diviser pour régner* », proposer un algorithme ayant une meilleur complexité.

(1)
Entrée $Q \in \mathbb{R}[X]$, $k \in \mathbb{N}$
Sortie $Q^k \in \mathbb{R}[X]$.
$P \leftarrow 1$ polynôme constant valant 1.
Pour i allant de 1 à k faire
  $P \leftarrow P \times Q$
  ↑ Mult de Q1
Retourner P.

⚠ P et Q n'ont pas le même degré donc la complexité est en

(2) $O(k n^2)$.

(3) Exponentiation rapide

Entrée : $Q \in \mathbb{R}[X]$, $k \in \mathbb{N}$.
Sortie : $Q^k \in \mathbb{R}[X]$.
Si $k = 2p$ alors
  $P \leftarrow Q^p$
  Renvoyer $P \times P$
Sinon ($k = 2p+1$)
  $P \leftarrow Q^p$
  Renvoyer $P \times P \times Q$

Complexité en $O(\log k + \log n)$.

Bien mieux !

Préparation aux oraux CCINP n°3

Arbres de preuve

On définit $\mathscr{R}$ le système de preuves formé des règles usuelles de dérivation en logique intuitionniste. Ces règles sont :

$$\frac{}{\Gamma \vdash \top}\,\top\text{i} \qquad \frac{}{\Gamma, \varphi \vdash \varphi}\,\text{ax} \qquad \frac{\Gamma \vdash \bot}{\Gamma \vdash \varphi}\,\bot\text{e} \qquad \frac{\Gamma, \varphi \vdash \bot}{\Gamma \vdash \neg\varphi}\,\neg\text{i} \qquad \frac{\Gamma \vdash \varphi \qquad \Gamma \vdash \neg\varphi}{\Gamma \vdash \bot}\,\neg\text{e}$$

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}\,\to\text{i} \qquad \frac{\Gamma \vdash \varphi \to \psi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi}\,\to\text{e} \qquad \frac{\Gamma \vdash \varphi \qquad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi}\,\wedge\text{i}$$

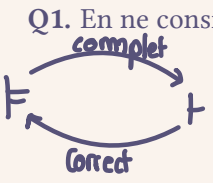
$$\frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \varphi}\,\wedge\text{e,g} \qquad \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \psi}\,\wedge\text{e,d} \qquad \frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi}\,\vee\text{i,g} \qquad \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \vee \psi}\,\vee\text{i,d}$$

$$\frac{\Gamma \vdash \phi \vee \varphi \qquad \Gamma, \phi \vdash \psi \qquad \Gamma, \varphi \vdash \psi}{\Gamma \vdash \psi}\,\vee\text{e}$$

L'arbre ci-dessous est la première partie de l'exercice. Il donne une dérivation fausse.

$$\dfrac{\dfrac{\dfrac{}{A \vee B \vdash A \vee B}}{A \vee B \vdash A} \qquad \dfrac{\dfrac{}{A \vee B \vdash A \vee B}}{A \vee B \vdash B}}{\dfrac{A \vee B \vdash A \wedge B}{\vdash (A \vee B) \to (A \wedge B)}}$$

Q1. En ne considérant *que* la racine de l'arbre, montrer que la dérivation est fausse.



$\mathscr{R}$ est complet. Il suffit de montrer que $\not\vdash (A \vee B) \to (A \wedge B)$.

Par exemple avec $A \leftarrow$ VRAI et $B \leftarrow$ FAUX, on a $A \vee B \leftarrow$ VRAI mais $A \wedge B \leftarrow$ FAUX

Q2. Étiqueter l'arbre, et indiquer les erreurs de dérivations si elles existent.

On a deux erreurs de dérivation

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vee B \vdash A \vee B}\,\text{Ax}}{A \vee B \vdash A}\,\text{E} \qquad \dfrac{\dfrac{}{A \vee B \vdash A \vee B}\,\text{Ax}}{A \vee B \vdash B}\,\text{E}}{A \vee B \vdash A \wedge B}\,\wedge\text{i}}{\vdash (A \vee B) \to (A \wedge B)}\,\to\text{i}$$

On définit deux nouvelles règles « ra » et « $\neg\neg$e », comme montré ci-dessous. On définit le système de preuve S_1 comme le système $\mathscr{R}$ auquel on ajoute la règle « ra ». On définit le système S_2 comme le système $\mathscr{R}$ auquel on ajoute la règle « $\neg\neg$e ».

$$\frac{\Gamma, \neg\varphi \vdash \bot}{\Gamma \vdash \varphi}\,\text{ra} \qquad \frac{\Gamma \vdash \neg\neg\varphi}{\Gamma \vdash \varphi}\,\neg\neg\text{e}$$

On dit qu'une règle se dérive d'une autre si on peut créer un arbre de preuve montrant la conséquence de la règle, en supposant que l'on peut vérifier ses causes. Par exemple, la règle « cut » se déduit de la règle « $\to$e ». En effet

$$\frac{\Gamma, \varphi \vdash \psi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi}\,\text{cut} \qquad\qquad \dfrac{\dfrac{\dfrac{\textit{Supposé}}{\Gamma, \varphi \vdash \psi}}{\Gamma \vdash \varphi \to \psi}\,\to\text{i} \qquad \dfrac{\textit{Supposé}}{\Gamma \vdash \varphi}}{\Gamma \vdash \psi}\,\to\text{e}$$

Q3. Dériver la règle « ra » en utilisant les règles de S_2.

Supposé

$$\dfrac{\dfrac{\Gamma, \neg\varphi \vdash \bot}{\Gamma \vdash \neg\neg\varphi}\,\neg\text{i}}{\Gamma \vdash \varphi}\,\neg\text{e}$$

Q4. Peut-on dériver la règle « $\neg\neg$e » en utilisant les règles du système S_2 ?

Oui

$$\dfrac{\dfrac{\dfrac{}{\Gamma, \neg\varphi \vdash \neg\varphi}\,\text{Ax} \qquad \dfrac{\textit{Supposé}}{\Gamma, \neg\varphi \vdash \neg\neg\varphi}\,\text{Aff}}{\Gamma, \neg\varphi \vdash \bot}\,\neg\text{e}}{\Gamma \vdash \varphi}\,\text{RA}$$

Q5. Que dire des systèmes S_1 et S_2 ?

Ils sont équivalents :

- pour transformer une preuve dans S_1 en une preuve dans S_2, on remplace avec des morceaux d'arbres.
- De même dans l'autre sens.

Hugo Salou

Préparation aux oraux CCINP n°7

Langages réguliers

Q1. On considère $L_{\mathrm{GL}} = \mathscr{L}(a^\star\, b\, (a|b)^\star)$ un langage sur l'alphabet $\Sigma = \{a, b\}$.

(1) Décrire simplement le langage L_{GL}.

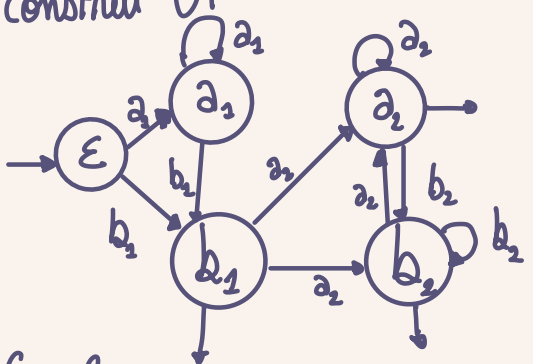
(2) À l'aide de la construction de Glushkov, construire un automate $\mathscr{A}$ reconnaissant L_{GL}.

(1) L_{GL} est le langages des mots commençant par des 'a' puis un 'b'.

(2)

	Λ	P	S	F
a_1	$\varnothing$	a_1	a_1	–
a_1^*	ε	a_1	a_1	a_1a_1
$a_1^*b_2$	$\varnothing$	a_1, b_2	b_2	a_1a_2, a_1b_1
$a_2\|b_2$	$\varnothing$	a_2, b_2	a_2, b_2	–
$(a_2\|b_2)^*$	ε	a_2, b_2	a_2, b_2	$a_2a_2, a_2b_2, b_2a_2, b_2b_2$
f	$\varnothing$	a_1, b_2	a_2, b_2 b_1	a_2a_2, a_2b_2 b_2a_2, b_2b_2, a_1a_1, a_1b_1, b_1a_2, b_1b_2

- On linéarise $a^*\, b\, (a|b)^*$ en $f = a_1^*\, b_1\, (a_2|b_2)^*$.
- On calcule le lang. local. eng. par f.
- On construit $\mathcal{A}$:



- On le dénumérote.

Q2. On considère l'alphabet $\Sigma = \{0, 1, +, =\}$. On pose

$$L_{\mathrm{ADD}} = \left\{ x = y + z \;\middle|\; \begin{array}{l} x, y \text{ et } z \text{ sont des entiers binaires} \\ x \text{ est la somme de } y \text{ et } z \end{array} \right\}.$$

Montrer que L_{ADD} n'est pas régulier. Pour cela, on pourra appliquer le lemme de l'étoile.

Supposons L_{ADD} régulier : il est donc reconnaissable par un automate déterministe à n états.

Soit $w = \text{“}\underbrace{1\,0\,0\cdots0}_{n+2}\,1 = 1\,\underbrace{0\cdots0}_{n+1}0 + \underbrace{0\cdots0}_{n+1}\,1\text{”}$ de longueur $3(n+2)+2 \geq n$.

D'après le lemme de l'étoile, il existent $x, y, z \in \Sigma^*$ tels que

- $x \cdot y \cdot z = w$
- $y \neq \epsilon$
- $|xy| \leq n$
- $\forall p \in \mathbb{N},\ xy^pz \in L_{ADD}$.

Deux cas à traiter

Cas 1 $x = \varepsilon$, alors $y = 1\,0^k$ (avec $k \neq 0$), xz vaut donc $0\cdots01 = 10\cdots00 + 0\cdots01$ et $xz \in L_{ADD}$ (Absurde !)

Cas 2 $x = 10^k$ et $y = 0^l$ (avec $l \neq 0$), $xyyz$ vaut $\underbrace{10\cdots00\cdots0}_{>n+1}\,1 = \underbrace{100\cdots0}_{n+1} + \underbrace{0\cdots01}_{n+1}$ et $xyyz \in L_{ADD}$. (Absurde !)

Q3. Soient A et B deux langages sur un alphabet Σ fini quelconque. On définit le *mélange parfait* de A et B comme le langage

$$\left\{ a_1b_1a_2b_2\ldots a_nb_n \;\middle|\; \begin{array}{l} a = a_1\ldots a_n \in A \\ b = b_1\ldots b_n \in B \end{array} \right\},$$

où chaque a_i et b_i est un élément de Σ (pour $i \in [\![1, n]\!]$).

Montrer que la classe des langages réguliers est close sous l'opération « mélange parfait ». Pour cela, on pourra construire un automate reconnaissant le mélange parfait de A et B à partir d'un automate reconnaissant A, et d'un autre reconnaissant B.

Cette construction fonctionne-t-elle toujours si a_i et b_i sont des mots sur Σ ?

Soit $\mathcal{A}$ un automate déterministe complet reconnaissant $A \in LR$.
De même soit $\mathcal{B}$ reconnaissant B.
On construit $\mathcal{C}$ l'automate défini par

$$\Sigma_{\mathcal{C}} = \Sigma ;\quad Q_{\mathcal{C}} = Q_{\mathcal{A}} \times Q_{\mathcal{B}} \times \{A, B\} \text{ (symboles)} ;\quad F_{\mathcal{C}} = F_{\mathcal{A}} \times F_{\mathcal{B}} \times \{A\}$$

$$q_0^{\mathcal{C}} = (q_0^{\mathcal{A}}, q_0^{\mathcal{B}}, A) \text{ (états initiaux)} ;\quad \delta_{\mathcal{C}}((p, q, A), x) = (\delta_{\mathcal{A}}(p, x), q, B)$$

$$\delta_{\mathcal{C}}((p, q, B), x) = (p, \delta_{\mathcal{B}}(q, x), A) \text{ (fonction de transition.)}$$

L'idée est qu'on alterne entre “lire dans $\mathcal{A}$” et “lire dans $\mathcal{B}$”. Les symboles A et B servent à savoir dans quel automate lire à l'étape suivante. On peut le démontrer assez facilement par double inclusion.

Non, ça ne marche pas avec des mots de Σ^*. Mais, par une construction similaire, on peut démontrer que LR est stable par cette opération V.

Hugo Salou

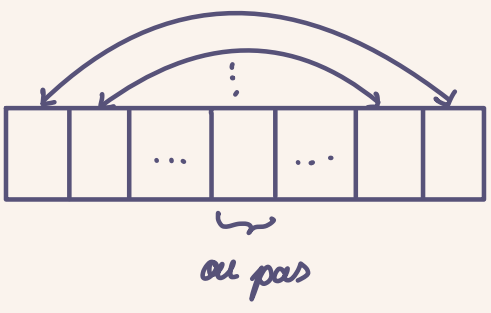
Préparation aux oraux CCINP n°8

Décidabilité

Q1. (1) Rappeler la définition du langage $\mathscr{L}(\mathscr{M})$ d'un machine $\mathscr{M}$.
(2) Construire une machine dont le langage est $\{w \in \Sigma^\star \mid w \text{ est un palindrome}\}$.

$\mathscr{L}(\mathscr{M}) = \{ w \in \Sigma^* \mid \mathscr{M}$ termine et renvoie VRAI sur l'entrée w ?

Entrée. $w \in \Sigma^*$.
Sortie. $w^{rev} \stackrel{?}{=} w$ *i.e. w est-il un palindrome*
$n \leftarrow |w|$
Pour i allant de 0 à $\lfloor n/2 \rfloor$ faire
  Si $w_i \neq w_{n-i}$ alors
    Renvoyer Faux
Renvoyer VRAI



Q2. Montrer l'indécidabilité du problème ARRÊT défini par :

ARRÊT : **Entrée.** Un mot $w \in \Sigma^\star$ et une machine $\langle \mathscr{M} \rangle$
Sortie. La machine $\mathscr{M}$ s'arrête-t-elle sur l'entrée w ?

Par l'absurde, soit $\mathscr{A}$ décidant ARRÊT. Soit $\mathscr{M}$ la machine

Que fait $\mathscr{M}$ sur l'entrée $\langle \mathscr{M} \rangle$?

- Cas 1 si $\mathscr{A}(\langle \mathscr{M} \rangle, \langle \mathscr{M} \rangle) =$ VRAI, alors $\mathscr{M}$ termine sur $\langle \mathscr{M} \rangle$ absurde car branche "si"
- Cas 2 si $\mathscr{A}(\langle \mathscr{M} \rangle, \langle \mathscr{M} \rangle) =$ faux, alors $\mathscr{M}$ ne termine pas sur $\langle \mathscr{M} \rangle$ absurde car branche "sinon"

Entrée $w \in \Sigma^*$
Si $\mathscr{A}(w, w)$ alors (→ termine sur toute entrée)
  Tant que VRAI faire
    ... rien ...
Sinon
  Renvoyer 42.

Étant donné un langage $L \subseteq \Sigma^\star$, on définit le problème APPARTIENT$_L$ comme :

APPARTIENT$_L$: **Entrée.** Un mot $w \in \Sigma^\star$.
Sortie. Est-ce que $w \in L$?

On dira ainsi que L est *indécidable* (resp. *décidable*) dès lors que le problème APPARTIENT$_L$ est indécidable (resp. décidable).

Q3. Montrer qu'il existe un sous-ensemble indécidable de $\{1\}^\star$. On procédera en quatre étapes, comme décrit ci-dessous.
(1) Démontrer l'indécidabilité du problème ESTVIDE

ESTVIDE : **Entrée.** La sérialisation $\langle \mathscr{M} \rangle$ d'une machine.
Sortie. Le langage de $\mathscr{M}$ est-il vide ?

Soit $(\langle \mathscr{M} \rangle, w)$ une entrée de CoARRÊT. On construit $\mathscr{N}$ comme la machine ci-dessous. On a :

Machine $\mathscr{N}(x)$:
  Exécute $\mathscr{M}$ sur w
  Renvoyer VRAI

$$\mathscr{L}(\mathscr{N}) = \begin{cases} \emptyset & \text{si } \mathscr{M} \text{ ne termine pas sur } w \\ \Sigma^* & \text{si } \mathscr{M} \text{ termine sur } w \end{cases}$$

Ainsi,

$$\langle \mathscr{N} \rangle \in \text{EstVide}^+ \Leftrightarrow \mathscr{L}(\mathscr{N}) = \emptyset$$
$$\Leftrightarrow \mathscr{M} \text{ ne termine pas sur } w$$
$$\Leftrightarrow (\langle \mathscr{M} \rangle, w) \in \text{CoArrêt}^+$$

D'où la réduction. On en déduit que est vide est décidable. (stabilité par complémentaire)

(2) Exprimer mathématiquement le langage des « instances positives » du problème ESTVIDE, c'est-à-dire l'ensemble des entrées vérifiant la condition du problème. Ce langage est-il decidable ?

$\text{EstVide}^+ = \{ \langle \mathscr{M} \rangle \mid \mathscr{L}(\mathscr{M}) = \emptyset \}$ indécidable : en effet,

$\text{Appartient}^+_{\text{EstVide}^+} = \{ w \in \Sigma^* \mid w \in \text{EstVide}^+ \} = \text{EstVide}^+$

(3) Définir une fonction injective calculable $f : \Sigma^\star \to \{1\}^\star$.

On suppose Σ fini (sinon, c'est impossible).

$$\Sigma^* \xrightarrow{\text{encodage binaire (injectif)}} \{0,1\}^* \xrightarrow{\text{bijectif}} \{1\}^*$$
$$w \longmapsto 1^{\text{bin } w}$$

bin w est l'entier représenté par w en binaire.

(4) En déduire qu'il existe un sous-ensemble indécidable de $\{1\}^\star$.

Soit f la fonction de (3), et $g : \text{Im } f \to \Sigma^*$, $f(w) \mapsto w$

On pose $E = f(\text{EstVide}^+) \subseteq \{1\}^*$

$w \in \text{Appartient}_E \Leftrightarrow w \in f(\text{EstVide}^+)$
$\Leftrightarrow g(w) \in \text{EstVide}^+$

d'où la réduction. On en déduit que E est indécidable.

Hugo SALOU

Préparation aux oraux CCINP n°8

k-ième plus petit élément

Dans cet exercice, on s'intéresse au problème k-PlusPetitÉlément.

k-PlusPetitÉlément : **Entrée.** Un ensemble $A = \{a_1, \dots, a_n\}$ ordonné par $\preccurlyeq$ de cardinal $n > k$.
Sortie. Quel est le k-ième plus petit élément de A ?

Q1. Proposer une méthode déterministe répondant au problème. On la décrira sous forme d'un algorithme synthétique. Quelle est la complexité de cette méthode ? On supposera avoir une complexité en $\mathrm{O}(1)$ pour la comparaison $x \preccurlyeq y$, et pour le calcul du cardinal.

Entrée. $A = \{a_1, \dots, a_n\}$ et $k \in [\![1, n]\!]$
Sortie. x le k-ième plus petit élément de A

On trie A avec un tri fusion : $(b_1, \dots, b_n)$

Retourne b_k

Complexité en $O(n \log n)$.

Au vu de la question précédente, on peut faire mieux. Pour cela, on utilise un algorithme probabiliste.

Procédure Rs(A, k)
- $n \leftarrow \operatorname{card} A$
- $\{a_1, \dots, a_n\} \leftarrow A$
- **Si** $n = 1$ **alors Renvoyer** a_1
- $i \leftarrow \mathcal{U}([\![1, n]\!])$
- $A_\prec \leftarrow \{x \in A \mid x \prec a_i\}$
- $A_\succ \leftarrow \{y \in A \mid x \succ a_i\}$
- **Si** $\operatorname{card} A_\prec \geqslant k$ **alors Renvoyer** Rs($A_\prec$, k) ← merci Ame-Lise
- **Sinon si** $\operatorname{card} A_\prec = k - 1$ **alors Renvoyer** a_i
- **Sinon Renvoyer** Rs($A_\succ$, $k - (\operatorname{card} A_\prec) - 1$)

Q2. De quel type d'algorithme cela s'agit-t-il ?

Algorithme Las Vegas
Lo s'il termine, alors c'est correct.
On peut s'en convaincre sur un exemple.

Q3. En notant $C(n)$ la variable aléatoire donnant le nombre d'opérations pour un appel de Rs(A, k) avec un ensemble A de cardinal n, démontrer que $\mathbb{E}[C(n)] = \mathrm{O}(n)$.

Pour $l \in [\![1, n]\!]$, soient E_l l'évènement " $i = l$ dans l'algo "
et X_l la variable aléatoire telle que
$(X_l = 1) = E_l$
$(X_l = 0) = \bar{E}_l$.

Comme i est choisie uniformément, $P(E_l) = \frac{1}{n}$.

Relation de récurrence :

$$C(n) \leq \sum_{i=1}^{n} X_i \, C(\max(n-i, i-1)) + O(n)$$

$O(n)$: construction de $A_\prec$ et $A_\succ$

$\sum_{i=1}^{n} X_i \, C(\max(n-i, i-1))$: Différents cas en fonction de i

On pose $F(n) = \mathbb{E}[C(n)]$.

Montrons par récurrence forte $F(n) \leq Cn$.

↳ On utilise la définition d'un O : $F(n) = O(n)$ ⇕ $F(n) \leq \alpha n$ (α fixé)

→ D'où, $F(n) = \mathbb{E}[C(n)] = O(n)$.