RAPPORT DE L'ÉPREUVE ORALE D'INFORMATIQUE FONDAMENTALE ULC MP 2013
ÉCOLES CONCERNÉES : ENS DE CACHAN, ENS DE LYON, ENS DE PARIS
Coefficients : **PARIS MPI : 20 groupe I : 4**
LYON MPI : 3 groupe I : 4
CACHAN MPI : 12 groupe I : 5
MEMBRES DE JURY : P. BAILLOT, N. OLLINGER & A. SAURIN

L'épreuve orale d'informatique concerne les candidats aux trois Écoles Normales Supérieures du concours MPI et ceux du concours informatique. L'épreuve concerne de plus en plus de candidats : le jury a ainsi examiné 217 candidats cette année contre 193 et 152 les années précédentes.

L'exercice consiste en un oral de 45 minutes sans préparation. Les notes se sont étalées entre 5 et 20 sur 20, avec une moyenne de 11,85 et un écart-type de 3,32 (voir l'histogramme en fin de document).

Le jury rappelle que l'épreuve est publique et conseille vivement aux futurs candidats et à leurs enseignants de venir assister au moins une fois à l'épreuve afin de mieux l'appréhender. Il est malheureusement nécessaire de rappeler que ceci ne doit pas être au détriment du candidat : les personnes souhaitant assister à un oral doivent s'assurer auprès du candidat qu'il n'en est pas gêné et s'astreindre à la plus grande discrétion durant l'oral.

Le jury recommande aux candidats de faire une lecture complète du programme officiel, disponible sur le site internet du ministère de l'Éducation Nationale [1]. L'oral d'informatique fondamentale se concentre principalement sur les trois dernières parties du programme (Structures de données et algorithmes, Automates finis, Notions de logique), tout en faisant très fréquemment appel à la première (Méthodes de programmation) qui forme, en quelque sorte, un socle aux trois autres.

Le jury attend des candidats la connaissance de l'ensemble des notions et résultats au programme, que ce soit dans la partie abordée en première année ou dans celle de deuxième année. Une déficience sur une question de cours est sévèrement sanctionnée.

Cette année certains candidats ne maîtrisaient pas bien, en logique propositionnelle, les notions de syntaxe (formules) et de sémantique (valuation/distribution de valeur de vérité, satisfiabilité) alors que le programme, aussi bien dans sa version de 2004 que le nouveau programme, insiste sur le fait que le but de cette partie du programme *«est de familiariser progressivement les étudiants avec la différence entre syntaxe et sémantique»*. D'autres candidats n'étaient pas à l'aise avec la notion de relation, ainsi que celle de relation d'ordre ou ont eu beaucoup de mal à conduire des raisonnements sur les mots.

Lorsqu'un algorithme est demandé, les candidats devraient éviter deux écueils : (1) perdre du temps à écrire dans un langage de programmation un programme syntaxiquement correct, (2) se contenter de donner une simple définition mathématique de l'objet calculé. On attend au contraire dans ce cas un simple programme en pseudo-code.

1. Voir `ftp://trf.education.gouv.fr/pub/edutel/bo/2004/hs3/annexe7.pdf` pour le programme encore en vigueur pour le concours 2014 et `http://cache.media.enseignementsup-recherche.gouv.fr/file/special_3_ESR/50/9/programme-option-informatique_252509.pdf` (annexe du Bulletin officiel spécial n° 3 du 30 mai 2013) pour le programme en vigueur à partir du concours 2015.

Le jury souligne également que trop de candidats oublient de justifier la correction et la terminaison de leur algorithme voire ne semblent pas comprendre le problème de la correction d'un algorithme.

Les exercices proposés ont fréquemment testé la capacité des candidats à assimiler rapidement une notion nouvelle, ainsi qu'à développer une démarche heuristique qu'ils fassent partager à l'examinateur. Si une bonne culture informatique est appréciée par le jury, il faut noter que la connaissance de notions hors programme n'a pas nécessairement avantagé les candidats.

Un sujet typique est organisé de la manière suivante :
– l'énoncé introduit une notion nouvelle qui s'appuie sur le programme ;
– les premières questions sont des applications des définitions de l'énoncé dont le but est de permettre au candidat d'assimiler les notions et à l'examinateur d'apprécier comment le candidat se comporte face à des notions nouvelles ;
– les questions suivantes conduisent à la démonstration d'un résultat non trivial d'informatique fondamentale ;
– les dernières questions de l'énoncé sont en général des questions beaucoup plus avancées, parfois résolues récemment ; si le jury n'attend pas, même d'un bon candidat, qu'il termine le sujet, il a parfois eu de très heureuses surprises.

Le jury attend du candidat qu'il soit autonome et qu'il justifie ses réponses, en apportant des arguments rigoureux sans y passer plus de temps que nécessaire. En particulier, le candidat ne doit pas attendre un signe d'assentiment de la part de l'examinateur à la fin de chacune de ses phrases pour poursuivre.

Le jury souhaite également formuler les quelques conseils suivants qui, pour élémentaires qu'ils soient, amélioreraient grandement la qualité de nombreuses prestations :
– éviter de perdre du temps au départ en recopiant tout l'énoncé au tableau ;
– les dessins ou schémas sont souvent utiles pour s'approprier les notions du sujet ainsi que pour expliquer l'idée qu'on souhaite développer ;
– ne pas effacer précipitamment le tableau à la fin de la question, avant que l'examinateur ait pu en terminer la lecture, ou lorsqu'une erreur a été détectée — il n'y a parfois que quelques éléments à modifier pour corriger un raisonnement ;
– tenir compte des éventuelles indications de l'examinateur : celles-ci ont pour but de permettre au candidat d'avancer et de montrer ce qu'il peut faire sur la suite de l'exercice et ne sont pas des pièges.

Quelques exemples de sujets sont disponibles sur le site `http://banques-ecoles.fr`, dans la rubrique *Annales*.

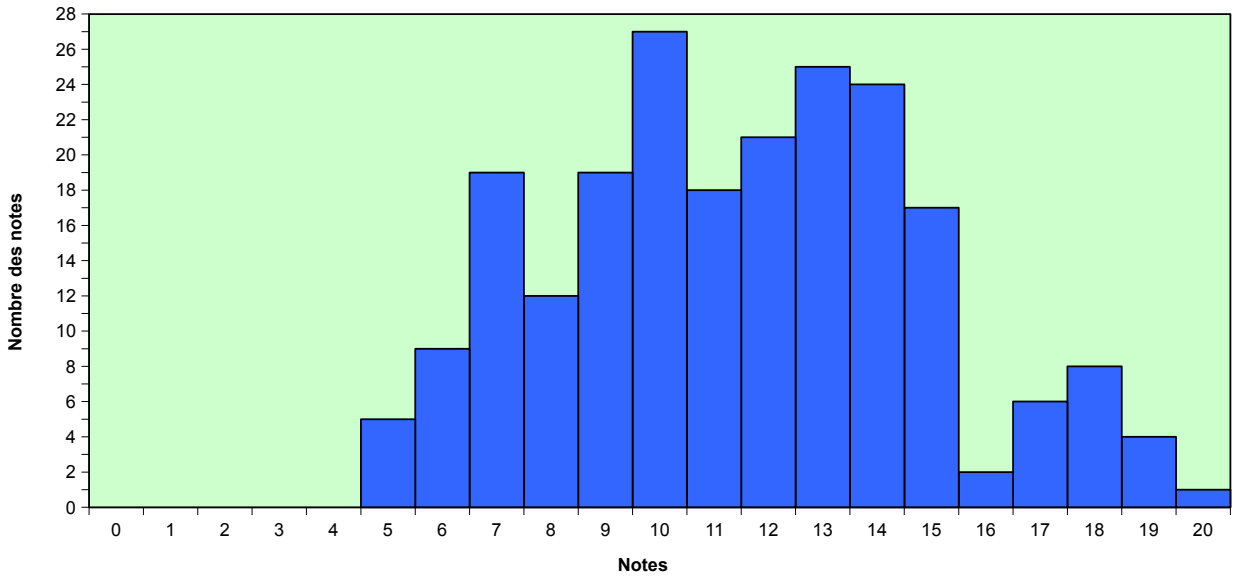


FIGURE 1 – Histogramme des notes obtenues par les candidats.

Épreuve orale d'Informatique Fondamentale

ULC MP 2012

Exemples de sujets

Patrick Baillot, Anne Bouillard, Alexis Saurin

Ce document consiste en une sélection, à titre d'exemples, de 3 sujets proposés à l'épreuve d'informatique fondamentale ULC MP en 2012. Cette épreuve consiste en un oral de 45 minutes sans préparation. Le lecteur pourra se référer au rapport de jury pour plus d'informations sur le déroulement de l'épreuve.

1 Autour des mots de Lukasiewicz

Soit $A = \{a_0, a_1, \ldots, a_n, \ldots\}$, $n \in \mathbb{N}$ un alphabet infini et δ une fonction de pondération telle que pour tout $n \in \mathbb{N}$, $\delta(a_n) = n - 1$. Si $w = b_1 \ldots b_m \in A^*$ est un mot, le poids de w est la somme des poids des lettres qui le composent : $\delta(w) = \sum_{i=1}^{m} \delta(b_i)$.

Un mot $w \in A^*$ est dit de **Lukasiewicz** s'il vérifie les propriété suivantes :

1. $\delta(w) = -1$.
2. Pour tout préfixe propre y de w, $\delta(y) \geq 0$.

On note L l'ensemble de mots de Lukasiewicz.

Question 1. Les mots suivants sont-ils des mots de Lukasiewicz ?
- $a_2a_1a_0a_2a_0a_1a_0$
- $a_3a_2a_0a_2a_0a_1a_1a_0$

Question 2. Caractériser $L \cap \{a_0, a_1\}^*$. Est-il rationnel ?

Question 3. Le langage $L \cap \{a_0, a_2\}^*$ est-il rationnel ?

On considère maintenant un système de réécriture. On se donne des variables X et des règles de transformation :

$$\forall n \in \mathbb{N},\ X \to a_n X \ldots X \qquad (= a_n X^n\,;\ X \text{ répété } n \text{ fois}).$$

Un mot est **engendré** par ce système de réécriture s'il peut s'obtenir à partir de X par application successives de règles de réécriture. Par exemple, le mot $a_2a_0a_1a_0$ est obtenu par

$$X \to a_2XX \to a_2Xa_1X \to a_2a_0a_1X \to a_2a_0a_1a_0.$$

On note alors $X \to^* a_2a_0a_1a_0$ pour signifier que $a_2a_0a_1a_0$ peut être obtenu à partir de X en un nombre quelconque d'étapes.

Plus généralement, pour $u, v \in (A \cup \{X\})^*$, on peut appliquer une règle qui transforme u et v si $u = u_1Xu_2$ et $v = u_1a_nX^nu_2$, et on note $u \to v$. On note aussi $u \to^* v$ si v est obtenu à partir de u en appliquant un nombre arbitraire de règles de réécriture. En particulier, on a toujours $u \to^* u$.

Question 4. Montrer que $a_2a_1a_0a_2a_0a_1a_0 \in L$ en donnant les règles de réécritures à appliquer à partir de X.

Question 5. Soit $w \in (A \cup \{X\})^*$. Montrer que si $XX \to^* w$, alors il existe w_1, w_2 tels que $w = w_1w_2$, $X \to^* w_1$ et $X \to^* w_2$.

Question 6. Supposons que $X \to^* w \in A^*$. Montrer que $w \in L$.

Question 7. Montrer que si $w \in L$, alors $X \to^* w$.

Un **arbre binaire** est défini récursivement comme

$$\text{ArbreBin} = \text{Feuille} \ + \ \text{ArbreBin} \ \times \ \text{Nœud} \ \times \ \text{ArbreBin}.$$

Un **arbre planaire enraciné** est un arbre dont on distingue la racine r, dont les nœuds ont un nombre arbitraires de fils ordonnés (l'ordre dans lequel on les représente importe) :

$$\text{Arbre} = \text{Nœud} \ + \ \text{Nœud} \ \times \ \text{Arbre} + \ \text{Nœud} \ \times \ \text{Arbre} \times \ \text{Arbre...}$$

Question 8. Montrer que les mots de Lukasiewicz sont en bijection avec les arbres planaires enracinés.

Question 9. Donner une relation entre le nombre d'arbres planaires enracinés de taille n et le nombre d'arbres binaires de taille $n-1$ (la taille d'un arbre est son nombre de Nœuds).

Question 10. En déduire le nombre de mots de L de longueur n.

2 Facteurs itérants d'un langage rationnel

Soient A un alphabet fini et L un langage de A^*. Un mot v est un *facteur itérant* de L s'il existe deux mots u, w tels que : $uv^*w \subseteq L$. On désigne par $\mathcal{I}(L)$ l'ensemble des facteurs itérants de L. D'autre part, si $k \geq 2$ soit $\mathcal{R}_k(L) = \{w \in A^* \setminus\ w^k \in L\}$ le langage des racines k-èmes de L.

L'objet de cet exercice est de montrer que si L est rationnel alors $\mathcal{R}_k(L)$ et $\mathcal{I}(L)$ sont aussi des langages rationnels, et de construire des automates les reconnaissant.

On représentera ici les automates déterministes sous la forme $\mathcal{A} = (Q, A, \delta, q_1, F)$ où q_1 est l'état initial et F est l'ensemble des états acceptants. On rappelle qu'un état q de l'automate est dit *utile* s'il existe un chemin de l'état initial à q, et un chemin de q à un état acceptant.

Étant donné un tel automate, un état q et un mot w, on notera $q \cdot w$ l'état (s'il existe) atteint par exécution de $\mathcal{A}$ depuis q, avec le mot w. On dit que l'automate $\mathcal{A}$ est *régulier à gauche* si :

$$\forall u_1, u_2 \in A^*, q_1 \cdot u_1 = q_1 \cdot u_2 \Rightarrow (\forall v \in A^*,\ q_1 \cdot vu_1 = q_1 \cdot vu_2).$$

On va chercher à montrer que tout automate déterministe complet est équivalent à un automate déterministe complet régulier à gauche.

Soit $\mathcal{A} = (Q, A, \delta, q_1, F)$ un automate déterministe complet et $Q = \{q_1, \ldots, q_n\}$. On va définir un automate dont l'ensemble des états est inclus dans Q^n, l'ensemble des vecteurs de longueur n à valeurs dans Q. Si $r \in Q^n$ on écrira r_i ou r_{q_i} sa i-ème composante. On note I le vecteur défini par $\forall j \in \{1, \ldots, n\},\ I_j = q_j$.

On définit alors $\mathcal{A}_g = (Q^n, A, \gamma, I, F_g)$ par :

- pour $a \in A$ et $r \in Q^n$, $\gamma(a, r) = s$ avec $\forall j \in \{1, \ldots, n\},\ s_j = \delta(a, r_j)$,
- $F_g = \{r \in Q^n,\ r_1 \in F\}$.

On considérera en fait juste les états *accessibles* de $\mathcal{A}_g$ (c'est-à-dire qui peuvent être atteints depuis l'état initial) et on notera encore $\mathcal{A}_g$ l'automate correspondant.

Question 11. Soit $\mathcal{D} = (Q, A, \delta, q_1, F)$ défini par $Q = \{q_1, q_2, q_3\}$, $A = \{a, b\}$, $F = \{q_3\}$, et δ donné par :

	q_1	q_2	q_3
a	q_2	q_1	q_2
b	q_1	q_3	q_1

L'automate $\mathcal{D}$ est-il régulier à gauche ? Construire l'automate $\mathcal{D}_g$.

Question 12. Montrer que si $\mathcal{A}$ est un automate déterministe complet alors $\mathcal{A}_g$ est un automate déterministe complet équivalent à $\mathcal{A}$ et régulier à gauche.

Soit $\mathcal{A} = (Q, A, \delta, q_1, F)$ un automate déterministe reconnaissant L. On note V_2 le sous-ensemble de Q défini par :

$$V_2 = \{q \in Q \setminus \exists w \in A^*\ q_1 \cdot w = q \text{ et } q_1 \cdot ww \in F\}.$$

Soit $\mathcal{A}_{\mathcal{R}_2} = (Q, A, \delta, q_1, V_2)$, c'est-à-dire qu'il est obtenu à partir de $\mathcal{A}$ en prenant V_2 comme ensemble d'états acceptants.

Question 13. Montrer que si $\mathcal{A}$ est régulier à gauche, alors :

$$L(\mathcal{A}_{\mathcal{R}_2}) = \mathcal{R}_2(L(\mathcal{A})). \tag{1}$$

Question 14. Montrer qu'on peut calculer l'ensemble V_2.

Question 15. Généraliser le résultat précédent en donnant la construction, pour $k \geq 3$, d'un automate $\mathcal{A}_{\mathcal{R}_k}$ qui reconnaît le langage $\mathcal{R}_k(L(\mathcal{A}))$.

Soit L un langage rationnel. On en vient maintenant au sujet du langage $\mathcal{I}(L)$ des facteurs itérants de L.

Soit $\mathcal{A} = (Q, A, \delta, q_1, F)$ un automate déterministe reconnaissant L.

Question 16. Étant donné $p \in Q$, soit $\mathcal{A}_p = (Q, A, \delta, p, \{p\})$. Montrer que pour tout état p utile on a :

$$L(\mathcal{A}_p) \subseteq \mathcal{I}(L).$$

Question 17. Montrer que si v est un facteur itérant de L, alors il existe un entier $k \geq 1$ et un état p utile tels que v^k appartient à $L(\mathcal{A}_p)$.

Question 18. Montrer que $\mathcal{I}(L)$ est reconnu par un automate fini qu'on peut effectivement construire à partir de $\mathcal{A}$.

3 Logique et théorie des graphes

Dans ce problème, on va d'abord établir un résultat fondamental de la logique propositionnelle, le ***théorème de compacité***, puis on l'appliquera à la théorie des graphes.

Théorème de compacité du calcul des propositions.

Question 19. Rappeler la définition des formules de la logique propositionnelle, de la notion de valuation et de satisfaisabilité.

Le théorème de compacité affirme que :

Théorème 1 (de Compacité) *Soit $\mathfrak{T}$ un ensemble de formules propositionnelles. On a l'équivalence entre les deux assertions suivantes :*

1. *$\mathfrak{T}$ est satisfaisable ;*
2. *toute partie finie de $\mathfrak{T}$ est satisfaisable.*

Question 20. Montrer $1 \Rightarrow 2$.

Question 21. Le cas où $\mathfrak{T}$ est fini est évident. Pourquoi ?

Question 22. Dans cette question, on admet le théorème de compacité. Que peut-on dire d'un ensemble infini de formules propositionnelles $\mathfrak{T}$ qui n'est pas satisfaisable ?

On introduit la notion de ***valuation finie adaptée*** à un ensemble de formules : On considère un ensemble $\mathfrak{F}$ de formules propositionnelles, une énumération des variables propositionnelles $(P_i)_{i\in\mathbb{N}}$ et une valuation σ d'une partie finie $(P_i)_{i\leq n}$ des variables propositionnelles. On dit que σ est une ***valuation finie adaptée à $\mathfrak{F}$*** si elle vérifie que toute partie finie $\mathfrak{F}' \subseteq \mathfrak{F}$ peut être satisfaite par un prolongement de σ à l'ensemble des variables propositionnelles apparaissant dans $\mathfrak{F}'$.

Question 23. Montrer que si $\mathfrak{T}$ est satisfaisable, toute valuation finie adaptée à $\mathfrak{T}$, définie sur n variables propositionnelles peut se prolonger en une valuation finie définie sur $n+1$ variables propositionnelles.

Question 24. Démontrer le théorème de compacité.

La suite du problème va consister à appliquer le théorème de compacité à la théorie des graphes.

Graphes. Un ***graphe*** $\mathcal{G}$ est la donnée d'un ensemble au plus dénombrable $\mathcal{S}_\mathcal{G}$, les sommets du graphe, et d'un ensemble $\mathcal{A}_\mathcal{G} \subseteq \{a \in \mathcal{P}(\mathcal{S}_\mathcal{G}), \#a = 2\}$ de paires non orientées de sommets, les arêtes du graphe. Dans le cas où $\mathcal{S}_\mathcal{G}$ est fini, on parlera de ***graphe fini*** et de ***graphe infini*** sinon.

On représentera les sommets d'un graphe $\mathcal{G}$ par des points du plan et les arêtes $a = \{s, t\}$ par une ligne joignant les deux sommets s et t.

Un graphe ***complet*** est un graphe $\mathcal{G}$ tel que toute paire de sommets disjoints est reliée par une arête (c'est-à-dire $\mathcal{A}_\mathcal{G} = \{a \in \mathcal{P}(\mathcal{S}_\mathcal{G}), \#a = 2\}$).

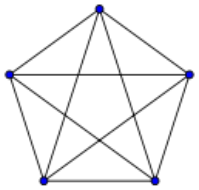


FIGURE 1 – K_5, graphe complet à 5 sommets.

Question 25. Donner des exemples illustrant les notions de graphe fini, infini et de graphe complet.

Graphes rouge et bleu. Dans la suite, on ne considèrera plus que des graphes complets dont les arêtes sont colorées soit en bleu, soit en rouge. Formellement, on se donne une application c de $\mathcal{A}_\mathcal{G}$ dans $\{\text{rouge}, \text{bleu}\}$. Un graphe sera dit monochrome si toutes ses arêtes ont la même couleur.

On va s'intéresser à l'existence de sous-graphes complets monochromes (on dira SGCM pour faire court) dans des graphes finis ou infinis.

Question 26. Montrer que dans tout graphe, l'intersection d'un SGCM rouge et d'un SGCM bleu est au plus réduite à un sommet.

Question 27. Montrer que tout graphe à 6 sommets contient un SGCM de taille 3.

Existence de SGCM. Dans la suite du problème, on va s'intéresser à deux généralisations du résultat de la question précédente :

Théorème 2 (SGCM infinis) *Tout graphe rouge et bleu infini contient un SGCM infini.*

Théorème 3 (SGCM finis) *Pour tout entier $n \in \mathbb{N}$, il existe un entier $k \in \mathbb{N}$ tel que tout graphe ayant au moins k sommets, contient un SGCM de taille n.*

La question précédente nous indique donc que pour $n = 3$, on peut choisir $k = 6$. On va d'abord démontrer le théorème pour les graphes finis se déduit du cas infini par le théorème de compacité puis on démontrera le cas infini.

Question 28. En admettant le théorème des SGCM infinis, démontrer le théorème des SGCM finis en utilisant le théorème de compacité.

Question 29. Démontrer le théorème des SGCM infinis.

Épreuve orale d'Informatique Fondamentale

Patrick Baillot, Nicolas Ollinger, Alexis Saurin

ULC MPI 2013

Résumé

Ce document consiste en une sélection, à titre d'exemples, de 3 sujets proposés à l'épreuve d'informatique fondamentale ULC MPI en 2012. Cette épreuve consiste en un oral de 45 minutes sans préparation. Le lecteur pourra se référer au rapport de jury pour plus d'informations sur le déroulement de l'épreuve.

1. Automates, bisimulation et minimisation

L'objectif de cet exercice est d'étudier une notion d'équivalence entre automates finis non-déterministes (AFN) plus fine que celle définie simplement en identifiant les automates acceptant le même langage. On va montrer en particulier que cette nouvelle notion d'équivalence est pertinente pour étudier les AFN minimaux.

Un automate fini non-déterministe $\mathcal{A}$ est ici défini par la donnée de :

$$(Q, \Sigma, \delta, S, F)$$

où :

— Q est un ensemble fini d'états,
— Σ est un alphabet fini,
— $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ est la fonction de transition,
— S est l'ensemble (non-vide) d'états initiaux,
— F est l'ensemble des états acceptants.

Si E est un ensemble d'états et $w \in \Sigma^\star$, on note $\delta(E, w)$ l'ensemble des états accessibles dans l'automate à partir d'un état dans E, en lisant le mot w. En particulier pour $w = \epsilon$ on a $\delta(E, \epsilon) = E$.

On consière deux AFN $\mathcal{A}_1$ et $\mathcal{A}_2$, avec les notations suivantes ($i = 1, 2$) :

$$\mathcal{A}_i = (Q_i, \Sigma, \delta_i, S_i, F_i).$$

Soit $\approx$ une relation entre Q_1 et Q_2, c'est-à-dire $\approx \subseteq Q_1 \times Q_2$.

On étend $\approx$ en une relation entre sous-ensembles de Q_1 et Q_2, aussi notée $\approx$:

$E_1 \approx E_2$ ssi $(\forall p \in E_1,\ \exists q \in E_2,\ p \approx q)$ et $(\forall q \in E_2,\ \exists p \in E_1,\ p \approx q)$.

Observer qu'on a : $\{p\} \approx \{q\}$ ssi $p \approx q$.

Pour $E_2 \subseteq Q_2$, $E_1 \subseteq Q_1$, on définit :

$$\begin{aligned} C_\approx(E_2) &= \{p \in Q_1 \ / \ \exists q \in E_2,\ p \approx q\}, \\ C_\approx(E_1) &= \{q \in Q_2 \ / \ \exists p \in E_1,\ p \approx q\}. \end{aligned}$$

On dit que $\approx$ est une *bisimulation* entre $\mathcal{A}_1$ et $\mathcal{A}_2$ si on a :

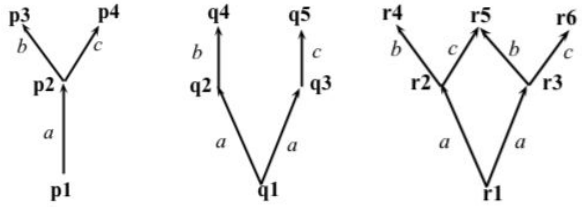
1. $S_1 \approx S_2$,
2. si $p \approx q$, alors pour tout $a \in \Sigma$, $\delta_1(p, a) \approx \delta_2(q, a)$,
3. si $p \approx q$, alors $p \in F_1$ ssi $q \in F_2$.

On dit que $\mathcal{A}_1$ est *bisimilaire* à $\mathcal{A}_2$ s'il existe une bisimulation entre $\mathcal{A}_1$ et $\mathcal{A}_2$.

Question 1. Montrer que la relation de bisimililarité est :

— symétrique : si $\mathcal{A}_1$ est *bisimilaire* à $\mathcal{A}_2$, alors $\mathcal{A}_2$ est *bisimilaire* à $\mathcal{A}_1$,
— transitive : si $\mathcal{A}_1$ est *bisimilaire* à $\mathcal{A}_2$ et si $\mathcal{A}_2$ est *bisimilaire* à $\mathcal{A}_3$, alors $\mathcal{A}_1$ est *bisimilaire* à $\mathcal{A}_3$.

Question 2. On considère les trois AFN $\mathcal{A}_1$, $\mathcal{A}_2$ et $\mathcal{A}_3$ suivants :



$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$
$S_1 = \{p_1\}$	$S_2 = \{p_2\}$	$S_3 = \{p_3\}$
$F_1 = \{p_3, p_4\}$	$F_2 = \{q_4, q_5\}$	$F_3 = \{r_4, r_5, r_6\}$

Montrer que $\mathcal{A}_1$ et $\mathcal{A}_3$ sont bisimilaires, et que $\mathcal{A}_1$ et $\mathcal{A}_2$ ne le sont pas.

Question 3. Montrer que si $I \neq \emptyset$ et si pour tout $i \in I$, $\approx_i$ est une bisimulation entre $\mathcal{A}_1$ et $\mathcal{A}_2$, alors la relation $\approx$ définie comme $\cup_{i \in I} \approx_i$ est une bisimulation.

Question 4. Soit $\approx$ une bisimulation entre $\mathcal{A}_1$ et $\mathcal{A}_2$, E_1 et E_2 deux ensembles d'états de $\mathcal{A}_1$ et $\mathcal{A}_2$ respectivement, tels que $E_1 \approx E_2$. Soit w un mot. Montrer que $\delta_1(E_1, w) \approx \delta_2(E_2, w)$.

Question 5. Montrer que deux automates bisimilaires acceptent le même langage.

Soit $\approx$ une bisimulation entre $\mathcal{A}_1$ et $\mathcal{A}_2$. Le *support de* $\approx$ *dans* $\mathcal{A}_1$ est l'ensemble $C_\approx(Q_2)$, défini par :

$$C_\approx(Q_2) = \{p \in Q_1 \ / \ \exists q \in Q_2, \ p \approx q\}.$$

Question 6. Montrer qu'un état de A appartient au support dans $\mathcal{A}_1$ de toutes les bisimulations entre $\mathcal{A}_1$ et un autre automate ssi il est accessible.

Question 7. Soit $\mathcal{A}_1^{acc}$ l'automate obtenu en restreignant $\mathcal{A}_1$ aux états accessibles. Montrer que $\mathcal{A}_1$ et $\mathcal{A}_1^{acc}$ sont bisimilaires.

Une *auto-bisimulation* est une bisimulation entre un automate et lui-même.

Dans la suite on considère un AFN $\mathcal{A} = (Q, \Sigma, \delta, S, F)$.

Question 8. On considère l'ensemble des relations de $Q \times Q$ muni de l'ordre d'inclusion $\subseteq$. Montrer qu'il existe une autobisimulation $\equiv_\mathcal{A}$ maximale pour $\subseteq$. Montrer que la relation $\equiv_\mathcal{A}$ est une relation d'équivalence (c'est-à-dire réflexive, symétrique, transitive) sur Q.

(On rappelle qu'une relation R sur Q est symétrique si : $\forall q \in Q,\ qRq$).

Supposons maintenant que tous les états de $\mathcal{A}$ sont accessibles et notons $\equiv_{\mathcal{A}}$ par $\equiv$.

On définit alors :

— si $p \in Q$, $[p] =_{def} \{q \in Q \ / \ p \equiv q\}$

— $\gtrsim =_{def} \{(p, [p]) \ / \ p \in Q\}$,

— pour tout $E \subseteq Q$, $E' =_{def} \{[p] \ / \ p \in E\}$.

Question 9. Montrer les propriétés suivantes, pour tous $D, E \subseteq Q$:

1. $D \subseteq C_{\equiv}(E) \Leftrightarrow D' \subseteq E'$,
2. $D \equiv E \Leftrightarrow D' = E'$,
3. $D \gtrsim D'$.

On définit maintenant l'AFN $\mathcal{A}'$ appelé *automate quotient de* $\mathcal{A}$:

$$\mathcal{A}' = (Q', \Sigma, \delta', S', F'),$$

où Q', S', F' sont construits comme indiqué précédemment, et δ' est défini par :

$$\delta'([p], a) =_{def} \delta(p, a)'.$$

Question 10. Montrer que δ' est bien définie.

Question 11. Montrer que la relation $\gtrsim$ est une bisimulation entre $\mathcal{A}$ et $\mathcal{A}'$.

Question 12. Montrer que l'unique auto-bisimulation sur $\mathcal{A}'$ est la relation identité $=$.

Question 13. Démontrer le résultat suivant :

Proposition 1 *Soit $\mathcal{A}$ un AFN sans état inaccessible et $\equiv$ l'autobisimulation maximale sur $\mathcal{A}$.*

L'automate quotient $\mathcal{A}'$ est minimal, pour le nombre d'états, parmi les automates bisimilaires à $\mathcal{A}$. De plus si $\mathcal{A}''$ est un automate bisimilaire à $\mathcal{A}$ minimal, alors $\mathcal{A}''$ est isomorphe à $\mathcal{A}'$.

2. Mots et périodes

Dans cet exercice, Σ est un alphabet fini contenant au moins deux lettres. On note ε le mot vide et $|u|$ la *longueur* d'un mot $u \in \Sigma^*$. On note u^n le mot u répété n fois (donc $|u^n| = n|u|$). Soit $u = a_1 \dots a_n \in \Sigma^n$ un mot de longueur n. Les *préfixes* de u sont les mots de la forme $a_1 \dots a_i$ et les *suffixes* de u sont les mots de la forme $a_i \dots a_n$ où $1 \leqslant i \leqslant n$. Un préfixe (resp. suffixe) v d'un mot u est un *préfixe propre* (resp. *suffixe propre*) si $v \neq u$.

Question 1. Montrer que deux mots u et v commutent, *i.e.* $uv = vu$, si et seulement s'il existe un mot z et deux entiers m et n tels que $u = z^m$ et $v = z^n$.

Un mot $u = a_1 \dots a_n$ est périodique de période $m > 0$ si pour toute paire d'entiers $1 \leqslant i \leqslant n$ et $1 \leqslant j \leqslant n$, si $j - i = m$ alors $a_i = a_j$. On note $p(u)$ la plus petite période de u.

Question 2. Calculer la période du mot $w = abbabaab$.

Question 3. Montrer que si u et v commutent alors $p(u) \leqslant \mathrm{pgcd}(|u|, |v|)$.

La *rotation* $\pi(u)$ d'un mot $u = a_1 \dots a_n$ est le mot obtenu en déplaçant la première lettre de u de la première à la dernière position, *i.e.* $\pi(u) = a_2 \dots a_n a_1$. On note π^k l'application de k rotations successives.

Question 4. Calculer les rotations $\pi^n(w)$ du mot $w = abbabaab$.

On fixe un ordre total $\leqslant$ sur Σ. L'*ordre lexicographique* induit par $\leqslant$ sur Σ^* est défini par $u \leqslant v$ si les mots u et v satisfont l'un des deux cas suivants :

— u est un préfixe de v ;

— $u = wau'$ et $v = wbv'$ avec $a < b$.

Question 5. Montrer que si le mot u n'est pas un préfixe du mot v alors si $u \leqslant v$, pour toute paire de mots w, z, on a $uw \leqslant vz$.

Un mot est *Lyndon* s'il est strictement inférieur à chacun de ses suffixes propres.

Question 6. Monter qu'un mot est Lyndon si et seulement si pour tout entier $1 \leqslant n < |u|$ on a $u < \pi^n(u)$.

Question 7. Montrer que si u est Lyndon et si $u = vw$ alors $u \leqslant wv$.

Un *bord* v d'un mot u est un mot non vide tel que u se factorise en vwv pour un certain mot w.

Question 8. Montrer que si u est Lyndon alors u est sans bord.

Un mot w non vide est un *chevauchement* pour une paire de mots (u, v) si $\Sigma^* w$ intersecte $\Sigma^* u$ et $w\Sigma^*$ intersecte $v\Sigma^*$. La *période locale* $l(u, v)$ d'une paire de mots (u, v) est la longueur d'un plus petit chevauchement pour (u, v).

Question 9. Calculer chacune des périodes locales des factorisations (u, v) où $uv = abbabaab$.

Question 10. Montrer que $l(u, v) \leqslant p(uv)$ pour toute paire de mots u, v.

Question 11. Montrer que si $l(u, v) \geqslant \max\{|u|, |v|\}$ alors $l(u, v) = p(uv)$, pour toute paire de mots u, v.

Question 12. Montrer que pour tout mot u tel que $p(u) = |u|$ on a $l(u, u) = |u|$.

Question 13. Montrer que pour tout mot w tel que $|w| \geqslant 3p(w)$, la borne de la question 10 est atteinte : $p(w) = l(u, v)$ pour une factorisation $w = uv$ de w.

3. Théorème de complétude pour la logique propositionnelle

On considère dans ce problème la question de la correspondance entre la validité d'une formule et sa prouvabilité dans un système de déduction, c'est-à-dire un système formel dont les objets représentent des démonstrations.

Question 1. Rappeler la définition des formules propositionnelles construites à partir d'un ensemble de variables propositionnelles $\mathcal{P}(\ni p, q, \dots)$ avec l'implication, la négation, la conjonction et la disjonction, qu'on notera $\Rightarrow$, $\neg$, $\wedge$ et $\vee$ et la définition des assignations de valeurs de vérité.

Question 2. Rappeler comment définir l'implication en fonction des autres connecteurs et énoncer les lois de de Morgan.

On considère l'ensemble $\mathcal{F}$ défini comme le plus petit ensemble de formules propositionnelles contenant les littéraux ainsi que les disjonctions et les conjonctions de formules de $\mathcal{F}$.

On peut associer à toute formule logique une formule de $\mathcal{F}$ qui lui est logiquement équivalente.

Question 3. Pour $A \in \mathcal{F}$, définir une formule $\overline{A}$ de $\mathcal{F}$ associée à $\neg A$.

LK. Dans la suite de l'exercice, on ne considérera plus que des formules de $\mathcal{F}$.

Un ***séquent*** est la donnée d'une liste de formules, noté $\vdash A_1, \dots, A_n$. Un séquent $\vdash A_1, \dots, A_n$ a pour interprétation intuitive : "la disjonction des formules $A_1, \dots, A_n$ est vraie". On étend la notion d'assignation de valeur de vérité aux séquents de la manière suivante :

$$\delta(\vdash A_1, \dots, A_n) = max(0, \delta(A_1), \dots, \delta(A_n))$$

On dira qu'un séquent $\vdash \mathcal{S}$ est ***valide*** lorsque $\delta(\vdash \mathcal{S}) = 1$ pour toute assignation δ et qu'il est ***satisfaisable*** lorsqu'il existe une assignation δ telle que $\delta(\vdash \mathcal{S}) = 1$.

Question 4. Les séquents $\vdash p \vee \neg p$, $\vdash p \vee p$ et $\vdash p \wedge \neg p$ sont-ils valides ?

Les ***règles d'inférence*** de LK sont données dans la figure 1. Les règles d'inférence ont en commun que chaque règle possède

- une unique ***conclusion*** qui est le séquent en-dessous de la barre d'inférence et
- 0,1 ou 2 ***prémisses***, qui sont les séquents au-dessus de la barre d'inférence.

Les règles de la figure 1 définissent des schémas de règles d'inférence où les variables A, B (resp. Γ, Δ) désignent n'importe quelle formule (resp. liste de formules) de $\mathcal{F}$ et p n'importe quelle variable propositionnelle.

On travaillera dans la suite avec des ***instances*** de ces règles. Par exemple

$$\frac{\vdash p \vee q, \neg p \vee \neg r \quad \vdash r, \neg p \vee \neg r}{\vdash (p \vee q) \wedge r, \neg p \vee \neg r}\ \wedge^- \qquad \text{est une instance de} \qquad \frac{\vdash A, \Gamma \quad \vdash B, \Gamma}{\vdash A \wedge B, \Gamma}\ \wedge^-$$

FIGURE 1 – Règles d'inférence de LK.

$$\frac{}{\vdash p, \neg p}\ Ax \qquad \frac{\vdash \Gamma, B, A, \Delta}{\vdash \Gamma, A, B, \Delta}\ E \qquad \frac{\vdash \Gamma}{\vdash A, \Gamma}\ A \qquad \frac{\vdash A, A, \Gamma}{\vdash A, \Gamma}\ C$$

$$\frac{\vdash A, \Gamma \quad \vdash B, \Gamma}{\vdash A \wedge B, \Gamma}\ \wedge^- \qquad \frac{\vdash A, B, \Gamma}{\vdash A \vee B, \Gamma}\ \vee^-$$

$$\frac{\vdash A, \Gamma \quad \vdash B, \Delta}{\vdash A \wedge B, \Gamma, \Delta}\ \wedge^+ \qquad \frac{\vdash A, \Gamma}{\vdash A \vee B, \Gamma}\ \vee_1^+ \qquad \frac{\vdash B, \Gamma}{\vdash A \vee B, \Gamma}\ \vee_2^+$$

Question 5. Montrer que pour toute règle d'inférence de prémisses $(\vdash \mathcal{S}_i)_{i \in I}$ et de conclusion $\vdash \mathcal{S}$ et pour toute assignation δ, si $\forall i \in I, \delta(\vdash \mathcal{S}_i) = 1$ alors $\delta(\vdash \mathcal{S}) = 1$.

Preuves dans LK. L'ensemble des preuves des séquents de LK est défini comme le plus petit ensemble tel qu'***une preuve d'un séquent*** $\vdash \mathcal{S}$ est la donnée

- d'une instance d'une règle d'inférence (issue de la figure ci-dessus), de prémisses $(\vdash \mathcal{S}_i)_{i \in I}$ et de conclusion $\vdash \mathcal{S}$, et
- de preuves π_i des séquents $\vdash \mathcal{S}_i$ pour tout $i \in I$

Un séquent est dit ***prouvable*** lorsqu'il existe une preuve de ce séquent.

On représentera les preuves de manière arborescente comme suit :

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash p, \neg p}\ Ax}{\vdash A, p, \neg p}\ A}{\vdash A \vee p, \neg p}\ \vee^- \quad \dfrac{\dfrac{\dfrac{}{\vdash q, \neg q}\ Ax}{\vdash \neg q, q}\ E}{\vdash \neg q \vee B, q}\ \vee_1^+}{\vdash (A \vee p) \wedge q, \neg p, \neg q \vee B}\ \wedge^+ \quad \dfrac{\vdash q, \neg q \vee B}{}\ E$$

Question 6. Donner une preuve pour le séquent :

$$\vdash p \wedge q, \neg p \vee \neg q.$$

Question 7. Montrer que tout séquent prouvable est valide.

LK$_{\wedge^\epsilon\vee^{\epsilon'}}$. Pour $\epsilon, \epsilon' \in \{+, -\}$, on notera $LK_{\wedge^\epsilon\vee^{\epsilon'}}$ le sous-système de LK qui ne contient que les règles de disjonction et de conjonction correspondant au signe qui lui est assigné.

Question 8. Montrer que le choix de ϵ et ϵ' ne change pas les séquents prouvables dans $LK_{\wedge^\epsilon\vee^{\epsilon'}}$ et qu'il s'agit exactement des séquents prouvables de LK.

Complétude de LK. Le reste du problème consiste à démontrer le théorème suivant qui est une réciproque de la propriété de la question 7 :

Théorème 1 (Complétude de LK) *Tout séquent $\vdash \mathcal{S}$ valide est prouvable dans LK.*

Question 9. Montrer que pour toute instance des inférences $\wedge^-$ et $\vee^-$, une assignation δ qui satisfait la conclusion d'une de ces règles satisfait chacune de ses prémisses.

En déduire en particulier que si la conclusion d'une de ces inférences est valide, alors toutes ses prémisses le sont.

Comparer cette propriété à une propriété déjà établie.

Question 10. Montrer qu'on peut remplacer la règle axiome par la règle $Ax^\star$:

$$\frac{}{\vdash p, \neg p, \Gamma}\ Ax^\star$$

et que le système $LK^\star$ ainsi obtenu prouve les mêmes séquents que LK.

Preuve de la complétude de LK. On va en fait montrer que le fragment de $LK^\star$ contenant simplement les règles $Ax^\star$, E, $\vee^-$ et $\wedge^-$ est suffisant pour avoir la complétude complet, c'est-à-dire que tout séquent $\vdash \mathcal{S}$ valide est prouvable en utilisant simplement les règles $Ax^\star$, E, $\vee^-$ et $\wedge^-$.

On omettra dans la suite de prendre en compte la règle d'échange en supposant qu'on peut appliquer les inférences à n'importe quelle formule d'un séquent.

On ajoute une nouvelle règle d'inférence, ✠, dont tout séquent de littéraux $\vdash l_1, \ldots, l_n$ tel que $\forall 1 \leq i, j \leq n, l_i \neq \neg l_j$ est conclusion :

$$\frac{}{\vdash l_1, \ldots, l_n}\ ✠ \qquad \text{Si pour tous } 1 \leq i, j \leq n, l_i \neq \neg l_j$$

On appellera ***parapreuve*** une preuve dans $LK^-_✠ = LK^- \cup \{✠\}$

Question 11. Montrer que tout séquent admet une parapreuve.

Question 12. Montrer que si un séquent admet une parapreuve contenant ✠, alors il n'est pas valide.

Question 13. En déduire la complétude de LK.