Composition d'informatique n°6

Sujet unique (Durée : 4 heures)

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

⋆ ⋆ ⋆

Le sujet contient deux problèmes indépendants (un à chaque partie). Il est un peu trop long pour tenir en 4h. Pour un sujet type Informatique Fondamentale (ENS), on commencera par la partie 2. On essaiera de consacrer au moins une heure à chaque problème.

1 Clustering en dimension 1

On souhaite mettre en place du soutien différencié au travail dans une classe de lycée. Pour ce faire, on souhaite séparer les élèves en plusieurs groupes de niveaux homogènes pour leur proposer des exercices adaptés à leur niveau. On dispose pour cela des notes des élèves et on veut les regrouper selon la similitude de leurs notes.

Formellement, on dispose d'un ensemble E de N réels $x_0 \leqslant x_1 \leqslant \ldots \leqslant x_{N-1}$. On cherche à déterminer une partition de $[\![0, N-1]\!]$ en $K \leqslant N$ sous-ensembles $\mathcal{P} = \{C_0, C_1, \ldots, C_{K-1}\}$ non vides tels que le score

$$S(\mathcal{P}) = \sum_{i=0}^{K-1} \sum_{j \in C_i} (x_j - \mu_i)^2$$

soit minimal, où $\mu_i = \dfrac{1}{|C_i|} \displaystyle\sum_{j \in C_i} x_j$ est la moyenne des éléments correspondant à la classe C_i. Autrement dit, on veut minimiser la somme des carrés des écarts de chaque élément à la moyenne de sa classe.

Par exemple, pour $E = \{1, 2, 3, 5, 8, 10, 14, 15, 18\}$, une solution optimale pour $K = 3$ est donnée par la partition suivante :

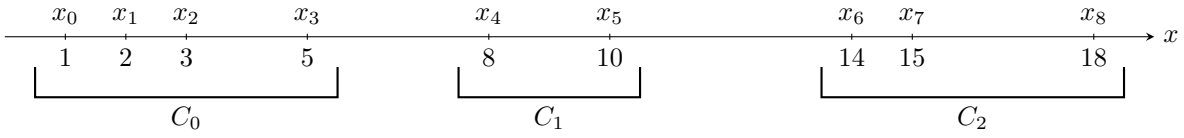


FIGURE 1 – Une partition optimale $\mathcal{P} = \{\{0, 1, 2, 3\}, \{4, 5\}, \{6, 7, 8\}\}$ de $[\![0, 8]\!]$ pour $K = 3$ et l'ensemble $E = \{1, 2, 3, 5, 8, 10, 14, 15, 18\}$, de score environ $19,42$.

Si $\{C_0, C_1, \ldots, C_{K-1}\}$ est une partition solution du problème, on remarque qu'on peut supposer sans perte de généralité que les classes sont rangées par ordre croissant, c'est-à-dire que pour $i < i'$ et $j \in C_i$, $j' \in C_{i'}$, alors $x_j \leqslant x_{j'}$. On supposera cette hypothèse vérifiée pour l'ensemble des partitions du problème.

1.1 Préliminaires

Question 1 Comment trouver une solution au problème lorsque $K = N$? Lorsque $K = N - 1$? Justifier.

Question 2 Appliquer l'algorithme des K-moyennes sur l'ensemble E de l'exemple précédent. On prendra $K = 3$, et on initialisera les barycentres à $b_0 = 1$, $b_1 = 8$ et $b_2 = 10$. On donnera les détails des étapes de calculs jusqu'à convergence de l'algorithme.

On accepte une représentation graphique pour la description des étapes de calcul.

On cherche à écrire une fonction `void tri(double* tab, int n)` qui trie un tableau `tab` de taille `n` en C. Pour ce faire, on propose les fonctions suivantes :

```c
void fusion(double* tab1, int n1, double* tab2, int n2, double* tab){
    int i1 = 0;
    int i2 = 0;
    for (int i=0; i<n1 + n2; i++){
        if (tab1[i1] <= tab2[i2]){
            tab[i] = tab1[i1];
            i1++;
        } else {
            tab[i] = tab2[i2];
        }
    }
}

void tri(double* tab, int n){
    if (n < 1) return;
    int n1 = n / 2;
    int n2 = n - n / 2;
    double* tab1 = malloc(n1 * sizeof(double));
    double* tab2 = malloc(n2 * sizeof(double));
    for (int i=0; i<n; i++){
        if (i < n1) tab1[i] = tab[i];
        else tab2[i - n1] = tab[i];
    }
    tri(tab1, n1);
    tri(tab2, n2);
    fusion(tab1, n1, tab2, n2, tab);
}
```

Question 3 Le code précédent comporte plusieurs (moins de 5) erreurs. Préciser les lignes où apparaissent ces erreurs et un moyen de les corriger. On demande de **NE PAS** recopier tout le code.

Pour la suite de cette partie, on supposera que le tableau `E` représentant un ensemble E sera toujours trié.

Pour $E = \{x_0, \ldots, x_{N-1}\}$ et $0 \leqslant i < j \leqslant N$, on note $S_{\text{emc}}(i, j)$ (pour « somme des écarts à la moyenne au carré ») la valeur

$$S_{\text{emc}}(i, j) = \sum_{k=i}^{j-1} (x_k - \mu)^2$$

où μ est la moyenne des éléments de $\{x_i, x_{i+1}, \ldots, x_{j-1}\}$.

Question 4 Écrire une fonction `double moyenne(double* E, int i, int j)` qui prend en argument un tableau E de N valeurs $\{x_0, \ldots, x_{N-1}\}$ triées et deux indices $0 \leqslant i < j \leqslant N$ et renvoie la moyenne des éléments de $\{x_i, x_{i+1}, \ldots, x_{j-1}\}$.

Question 5 Écrire une fonction `double somme_emc(double* E, int i, int j)` qui prend en argument une liste E de N valeurs triées et deux indices $0 \leqslant i < j \leqslant N$ et calcule et renvoie $S_{\text{emc}}(i, j)$.

On représente une partition $\mathcal{P} = \{C_0, C_1, \ldots, C_{K-1}\}$ de $[\![0, N-1]\!]$ par un tableau `P` de taille K telle que `P[i]` est le plus petit élément de C_i. Avec l'hypothèse faite précédemment sur les C_i, le tableau `P` doit nécessairement être trié. On remarque, de plus, que `P[0]` vaut toujours 0. Par exemple, la partition $\{\{0, 1, 2, 3\}, \{4, 5\}, \{6, 7, 8\}\}$ donnée dans l'exemple de la figure 1 est représentée par `{0, 4, 6}`.

Question 6 Dans cette question, on suppose $N = 9$. Quelle est le tableau `P` associé à la partition $\mathcal{P} = \{\{0, 1, 2\}, \{3, 4\}, \{5, 6, 7\}, \{8\}\}$? Quelle est la partition associée au tableau `{0, 1, 4, 5}` ?

Question 7 Écrire une fonction `double score(double* E, int N, int* P, int K)` qui prend en argument un ensemble `E` de taille N et une partition `P` de $[\![0, N-1]\!]$ de taille K et renvoie le score $S(\mathcal{P})$ tel qu'il a été défini précédemment.

Question 8 En déduire une fonction `int* clustering3(double* E, int N)` qui prend en argument un ensemble `E` de taille $N \geqslant 3$ et renvoie une partition `P` de $[\![0, N-1]\!]$ de taille $K = 3$ de score minimal.

Question 9 Quelle est la complexité temporelle de la fonction précédente ? Justifier.

1.2 Clustering hiérarchique ascendant

On cherche dans cette sous-partie à calculer une solution pas nécessairement optimale par une approche de clustering hiérarchique ascendant (CHA).

Question 10 Écrire une fonction `int classes_plus_proches(double* E, int N, int* P, int K)` qui prend en argument un ensemble `E` de taille N et une partition `P` de $[\![0, N-1]\!]$ de taille $K > 1$ et renvoie un indice i_{opt} tel que $C_{i_{\text{opt}}}$ et $C_{i_{\text{opt}}+1}$ sont les classes les plus proches, c'est-à-dire celles qui ont leurs moyennes les plus proches. En cas d'égalité, on choisira l'indice i_{opt} minimal.

Question 11 Écrire une fonction `int* fusion_classes(int* P, int K, int iopt)` qui prend en argument une partition `P` de $[\![0, N-1]\!]$ de taille K et un indice $0 \leqslant i_{\text{opt}} < K - 1$ et renvoie une partition de $[\![0, N-1]\!]$ de taille $K - 1$ où les classes d'indices i_{opt} et $i_{\text{opt}} + 1$ ont été fusionnées.

Question 12 En déduire une fonction `int* CHA(double* E, int N, int K)` qui calcule et renvoie une partition de taille K d'un ensemble E selon l'algorithme de clustering hiérarchique ascendant.

Question 13 Déterminer la complexité temporelle de la fonction `CHA` en fonction de N et de K.

Question 14 Montrer par un exemple bien choisi que l'algorithme de clustering hiérarchique ascendant ne permet pas de résoudre le problème de manière optimale.

On pourra prendre $N = 4$ et $K = 2$.

1.3 Solution optimale en programmation dynamique

Pour $n \in [\![0, N]\!]$ et $k \in [\![1, K]\!]$, on note $D(n, k)$ le score minimal possible d'une partition de $\{x_0, \ldots, x_{n-1}\}$ en k classes non vides.

Question 15 Que vaut $D(n, k)$ lorsque $k = 1$?

Question 16 Montrer que pour $n > 0$ et $k > 1$, $D(n, k) = \min_{i=k-1}^{n-1} (D(i, k-1) + S_{\text{emc}}(i, n))$.

Question 17 En déduire une fonction `double clustering_dynamique(double* E, int N, int K)` qui calcule le score minimal possible d'une partition de E en K classes non vides.

Question 18 Déterminer la complexité temporelle de la fonction précédente.

Question 19 Expliquer en français comment modifier la fonction précédente pour qu'elle renvoie une partition optimale plutôt que le score minimal. On ne demande pas de coder cette solution.

On cherche à améliorer la complexité temporelle de la solution précédente en effectuant des précalculs. Pour $0 \leqslant i < n \leqslant N$, notons $\mu(i,n)$ la moyenne des éléments de $\{x_i, \ldots, x_{n-1}\}$. On admet la formule suivante :

$$S_{\text{emc}}(0, n+1) = S_{\text{emc}}(0,n) + \frac{n}{n+1}(x_n - \mu(0,n))^2$$

Question 20 Décrire en français ou pseudo-code un moyen de calculer l'ensemble des $S_{\text{emc}}(i,n)$ pour $0 \leqslant i < n \leqslant N$ en complexité $\mathcal{O}(N^2)$. On détaillera les formules de récurrence utilisées.

2 Logique linéaire

D'après un sujet de Florian Hatat.

Ce sujet étudie une logique appelée **logique linéaire**, qui ressemble en partie à la déduction naturelle mais avec des variations qui permettent de mieux contrôler l'utilisation des hypothèses lors d'une preuve. En particulier, dans le premier fragment étudié en partie 2.1, il est impossible d'oublier une hypothèse et il est impossible de l'utiliser deux fois.

2.1 Présentation de la logique linéaire multiplicative

2.1.1 Ensemble des formules

On considère un ensemble dénombrable de variables, noté $\mathcal{V}$. L'ensemble $\mathcal{F}$ des formules de la logique linéaire est l'ensemble défini inductivement par :

- pour tout $x \in \mathcal{V}$, $x \in \mathcal{F}$;
- si $x \in \mathcal{V}$ alors $x^\perp \in \mathcal{F}$. Le connecteur $\_^\perp$ est appelé **négation linéaire** ;
- si $A_1, A_2 \in \mathcal{F}$ alors $A_1 \otimes A_2 \in \mathcal{F}$. Le connecteur $\otimes$ est appelé **tenseur** ;
- si $A_1, A_2 \in \mathcal{F}$ alors $A_1 \parr A_2 \in \mathcal{F}$. Le connecteur $\parr$ est appelé **par**.

On définit, si A est une formule, la notation $A^\perp$ en posant par induction sur la formule :

$$(x^\perp)^\perp = x \qquad (A \otimes B)^\perp = A^\perp \parr B^\perp \qquad (A \parr B)^\perp = A^\perp \otimes B^\perp$$

Il s'agit d'égalités syntaxiques. L'opération $(\_)^\perp : \mathcal{F} \to \mathcal{F}$ est alors une involution, c'est-à-dire que pour toute formule A, $(A^\perp)^\perp = A$.

En logique linéaire, un **séquent** est donné par une liste de formules :

$$\vdash A_1, A_2, \cdots, A_n$$

dans laquelle on s'autorise à échanger l'ordre des formules. Ainsi, $\vdash A, B, C = \vdash B, A, C$.

En revanche, comme il s'agit de listes, le nombre fois où apparait chaque formule est important. Les deux séquents suivants sont différents : $\vdash A, B, A \neq \vdash A, B$.

2.1.2 Règles de déduction de la logique linéaire

Les quatre règles de déduction de la logique linéaire sont les suivantes :

$$\frac{x \in \mathcal{V}}{\vdash x, x^\perp}\,\text{ax} \qquad \frac{\vdash \Gamma, A \qquad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}\,\text{cut} \qquad \frac{\vdash \Gamma, A \qquad \vdash \Delta, B}{\vdash \Gamma, A \otimes B, \Delta}\,\otimes \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\,\parr$$

On peut remarquer les faits suivants :

- pour la règle du tenseur la liste des formules des prémisses est une partition des formules du séquent conclusion : aucune formule n'est dupliquée ni supprimée,
- il en est de même pour la règle de coupure (cut),
- l'axiome n'est valable que pour les **variables**, pas pour des formules en général.

Dans un séquent de la logique linéaire, il n'y a que des formules à droite du symbole $\vdash$. Intuitivement, le tenseur se comporte comme une conjonction et le par se comporte comme une disjonction.

2.1.3 Premiers arbres de preuve

Question 21 Donner un arbre de preuve de $\vdash ((x \otimes y^{\perp}) \parr x^{\perp}) \parr y$.

On définit l'**implication linéaire** (dite **sucette**) $A \multimap B$ comme étant égale à la formule $A^{\perp} \parr B$. Il s'agit juste d'une notation syntaxique, le connecteur $\multimap$ n'étant pas un constructeur des formules.

Exemple

On a l'égalité $(x \otimes (y \parr t^{\perp})) \multimap z = (x^{\perp} \parr (y^{\perp} \otimes t)) \parr z$.

Question 22 Montrer que le connecteur $\parr$ est associatif, c'est-à-dire que les deux séquents suivants sont prouvables :

- $\vdash ((x \parr y) \parr z) \multimap (x \parr (y \parr z))$;
- $\vdash (x \parr (y \parr z)) \multimap ((x \parr y) \parr z)$;

où x, y et z sont des variables.

On prouve de la même façon que $\otimes$ est associatif et on admet, dans toute la suite du sujet, que l'on peut supposer ces deux connecteurs associatifs dans toute formule.

Question 23 La règle axiome n'est valable que pour des variables. Cependant, on voudrait l'utiliser également pour des formules, sous la forme :

$$\frac{}{\vdash A, A^{\perp}}\, axF$$

Montrer, par induction structurelle sur la formule A, que cette règle est **admissible**, c'est-à-dire que l'on peut construire un arbre de preuve dont la conclusion est $\vdash A, A^{\perp}$ en utilisant uniquement les quatre règles de la partie précédente.

Question 24 Démontrer que la règle suivante est dérivable en logique linéaire en incluant la règle (axF) :

$$\frac{\vdash \Gamma, A \multimap B \qquad \vdash \Delta, A}{\vdash \Gamma, \Delta, B}\, \multimap_e$$

Soit A une formule et x une variable. On définit par induction sur la formule le nombre de fois où x apparait dans A, que l'on note $|A|_x$ par (où $y \in \mathcal{V} \setminus \{x\}$) :

$$|x|_x = 1 \qquad |x^{\perp}|_x = 0 \qquad |y|_x = |y^{\perp}|_x = 0 \qquad |A \otimes B|_x = |A \parr B|_x = |A|_x + |B|_x$$

On définit de manière similaire le nombre de fois où $x^{\perp}$ apparait dans A.

Question 25 Soit A une formule démontrable en logique linéaire. Soit x une variable. Montrer que $|A|_x = |A|_{x^{\perp}}$.

Indication : on pourra utiliser, en la justifiant très rapidement, l'égalité $|A^{\perp}|_x = |A|_{x^{\perp}}$.

Question 26 Justifier que la formule $(x \otimes x) \multimap x$ n'est pas prouvable en logique linéaire.

2.2 Réseaux de preuve

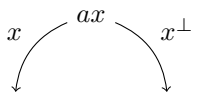
2.2.1 Structures de preuve

Une **structure de preuve** est un graphe orienté dont les arêtes sont étiquetées par des formules. Chaque nœud possède un type, parmi les types suivants : *ax*, *tens*, *par*, *concl*, *cut*. Pour un nœud donné, on dit que les arêtes entrantes de ce nœud sont les **arêtes prémisses du nœud**. Les arêtes sortantes d'un nœud sont appelées **arêtes conclusions du nœud**.

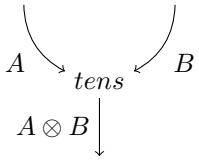
Une structure de preuve peut contenir plusieurs nœuds différents pour un même type (il peut y avoir plusieurs nœuds *tens*, *concl*, etc.).

Selon le type de chaque nœud, on impose les contraintes suivantes :

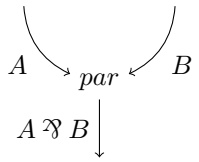
- un nœud *ax* ne possède aucune arête prémisse et possède deux arêtes conclusions dont les étiquettes sont respectivement x et $x^\perp$, où x est une variable ;



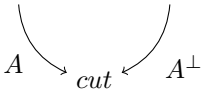
- un nœud *tens* possède deux arêtes prémisses, étiquetées par des formules A et B, et une arête conclusion étiquetée $A \otimes B$;



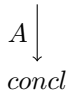
- un nœud *par* possède deux arêtes prémisses, étiquetées par des formules A et B, et une arête conclusion étiquetée $A \parr B$;



- un nœud *cut* possède deux arêtes prémisses étiquetées respectivement par les formules A et $A^\perp$ et ne possède aucune arête conclusion ;



- un nœud *concl* possède une arête prémisse et ne possède aucune arête conclusion. Ce nœud sert à ajouter un sommet destination fictif à une arête qui n'en a pas autrement et à identifier les conclusions de toute la structure.



Définition : Conclusions d'une structure de preuve

On considère tous les nœuds *concl* d'une structure de preuve. Chacun possède une arête entrante, qui est étiquetée par une formule. La liste de ces étiquettes forme les **conclusions** de la structure de preuve.

Question 27 Montrer qu'une structure de preuve ne possède pas de cycle orienté.

On peut adopter une représentation simplifiée où :

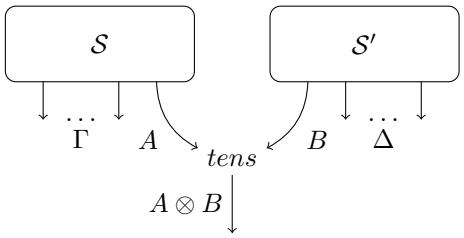
- l'orientation des arêtes se lit toujours du haut vers le bas
- les nœuds *ax* et *cut* sont représentés par de simples fils horizontaux,
- les nœuds *concl* ne sont pas dessinés,
- les nœuds *tens* sont étiquetés $\otimes$ et les nœuds *par* sont étiquetés $\parr$,
- on ne note les étiquettes des arêtes que pour les axiomes, les autres étiquettes étant alors imposées par les règles de construction.

La figure 2 donne un exemple d'une représentation simplifiée.

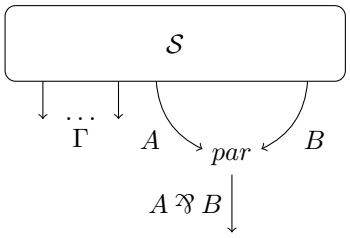
2.2.2 Réseaux de preuve

À chaque arbre de preuve de $\vdash A_1, \cdots, A_n$, on associe par induction une structure de preuve dont les conclusions sont $A_1, \cdots, A_n$:

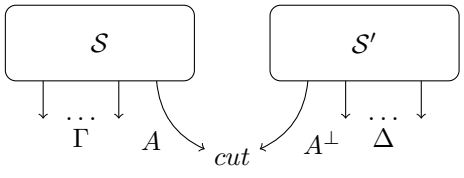
- la règle axiome donne un nœud *ax* et on ajoute des nœuds *concl* aux extrémités des deux arêtes ;
- pour la règle du tenseur, si on a une structure $\mathcal{S}$ de conclusions Γ, A et une structure $\mathcal{S}'$ de conclusions Δ, B, alors on peut construire une structure de conclusions $\Gamma, A \otimes B, \Delta$ en reliant l'arête de conclusion A de $\mathcal{S}$ et celle de conclusion B de $\mathcal{S}'$ à un nouveau nœud *tens* ;



- pour la règle du par, on procède de même en reliant à un nouveau nœud *par* les arêtes conclusion A et B d'un réseau de conclusions Γ, A, B ;



- pour la règle de la coupure, on relie les arêtes conclusions A et $A^\perp$ en entrée d'un nouveau nœud *cut*.



Définition

Un **réseau de preuve** est une structure de preuve obtenue de cette façon à partir d'un arbre de preuve.

La figure 2 donne un exemple de réseau de preuve.

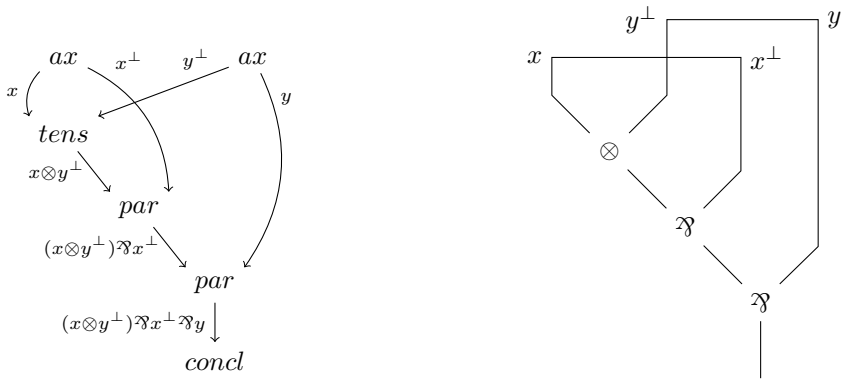


FIGURE 2 – Un réseau de preuve de $\vdash (x \otimes y^\perp) \parr x^\perp \parr y$ et sa représentation simplifiée

Question 28 Donner un réseau de preuve simplifié de $\vdash (((x \parr y) \otimes x^\perp) \parr (x^\perp \otimes y^\perp)) \parr x$.

Question 29 Montrer qu'il existe une structure de preuve qui n'est pas un réseau de preuve.

2.2.3 Critère de Danos et Régnier

2.2.3.1 Énoncé

Pour toute la suite du sujet, dans un graphe non orienté, lorsqu'on dit qu'on **déconnecte** une arête a du sommet s, où s et t sont les extrémités de a, cela signifie qu'on rajoute un nouveau sommet factice s', et qu'on remplace l'arête a par l'arête $\{s', t\}$. Dans un souci de simplification, lorsqu'on parlera de a dans le nouveau graphe, on fera référence à cette nouvelle arête.

Définition : Interrupteur

En oubliant l'orientation des arêtes, un **interrupteur** d'une structure de preuve $\mathcal{S}$ est un sous-graphe obtenu en déconnectant de chaque nœud *par* l'une de ses arêtes entrantes.

Un interrupteur est donc un graphe non orienté, avec éventuellement des arêtes multiples entre deux mêmes sommets.

Question 30 Dessiner un exemple d'interrupteur du réseau de preuve de la figure 2.

Question 31 Déterminer le nombre d'interrupteurs d'une structure de preuve possédant p nœuds *par*.

Le but de cette partie est de démontrer le théorème suivant :

Théorème : Critère de correction de Danos et Régnier

Une structure de preuve sans nœud *cut* est un réseau de preuve si et seulement si tous ses interrupteurs sont connexes et acycliques (c'est-à-dire sans tenir compte de l'orientation des arêtes).

Question 32 En procédant par induction sur les arbres de preuve, démontrer le sens direct du théorème : si $\mathcal{S}$ est un réseau de preuve alors tous ses interrupteurs sont connexes et acycliques.

2.2.3.2 Constitution d'un empire

Dans cette partie, on considère uniquement des structures de preuves qui n'ont pas de nœud *cut* et qui vérifient le critère de Danos et Régnier, c'est-à-dire telles que tout interrupteur est connexe et acyclique.

Le but de cette partie est de démontrer le lemme en question 40, qui à son tour permet de démontrer le sens réciproque du critère de Danos et Régnier.

Définition : a-interrupteur

Soit $\mathcal{S}$ une structure de preuve, a une arête de $\mathcal{S}$ et I un interrupteur de $\mathcal{S}$. Le graphe I' obtenu à partir de I en déconnectant a du nœud dont elle est prémisse est un a-interrupteur.

Définition : Empire

Soit a une arête d'une structure de preuve $\mathcal{S}$. L'empire de a, noté $\text{emp}(a)$ est le sous-graphe de $\mathcal{S}$ obtenu en ne conservant que les arêtes et les sommets qui reliés (par un chemin) à a dans tous les a-interrupteurs de $\mathcal{S}$.

Soit $\mathcal{S}$ une structure de preuve qui n'a pas de nœud *cut* et qui vérifie le critère de Danos et Régnier.

Question 33 Soit a une arête de $\mathcal{S}$. Soit b une arête conclusion d'un nœud axiome et b' l'autre arête conclusion de ce nœud. Montrer que $b \in \text{emp}(a)$ si et seulement si $b' \in \text{emp}(a)$.

Question 34 Soient b et c les deux arêtes prémisses d'un nœud *par*. Soit d la conclusion de ce même nœud. Soit a une arête distincte de b et de c. Montrer que $b \in \text{emp}(a)$ et $c \in \text{emp}(a)$ si et seulement si $d \in \text{emp}(a)$.

Question 35 Soit a une prémisse d'un nœud et b une conclusion du même nœud. Montrer que $b \notin \text{emp}(a)$.

Question 36 Lemme de clôture par le haut
Montrer que $\text{emp}(a)$ est clos vers le haut, c'est-à-dire que : pour tout nœud qui possède une prémisse b et une conclusion c, alors $c \in \text{emp}(a) \Rightarrow b \in \text{emp}(a)$.

Question 37 Lemme de quasi clôture par le bas
Soit a une arête. Soit s un nœud de $\mathcal{S}$, ayant une arête conclusion c telle que $c \notin \text{emp}(a)$. Soit b une arête prémisse de s telle que $b \in \text{emp}(a)$. Montrer que :

- soit a est prémisse de s,
- soit s est un nœud *par* et l'autre prémisse n'est pas dans $\text{emp}(a)$.

Question 38 Lemme d'emboitement d'empires
Soit a et b deux arêtes telles que $b \notin \text{emp}(a)$. Montrer que l'on a :

- soit $\text{emp}(a) \subseteq \text{emp}(b)$,
- soit $\text{emp}(a) \cap \text{emp}(b) = \emptyset$.

Question 39 En déduire que $\text{emp}(a)$ (en remettant les orientations initiales) est la plus grosse structure de preuve (pour l'inclusion) contenue dans $\mathcal{S}$ qui vérifie le critère de Danos et Régnier et qui a a parmi ses arêtes conclusions.

Question 40 Lemme du tenseur scindant
On suppose que $\mathcal{S}$ ne possède aucune arête allant d'un nœud *par* vers un nœud *concl*. Montrer que s'il existe des arêtes d'un nœud *tens* vers un nœud *concl*, alors l'un de ces nœuds *tens*, d'arêtes prémisses a et b, est tel

que emp(a) et emp(b) sont deux structures de preuves disjointes qui vérifient le critère de Danos et Régnier et forment une partition de la structure $\mathcal{S}$ privée du nœud *tens* et du nœud *concl* associé.

Question 41 Démontrer le sens réciproque du critère de Danos et Régnier.

2.2.4 Un théorème d'élimination des coupures

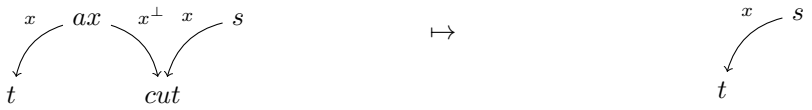
On considère un réseau de preuve obtenu à partir d'une preuve qui utilise éventuellement la règle (cut).

Question 42 Montrer qu'un tel réseau vérifie le critère de Danos et Régnier.

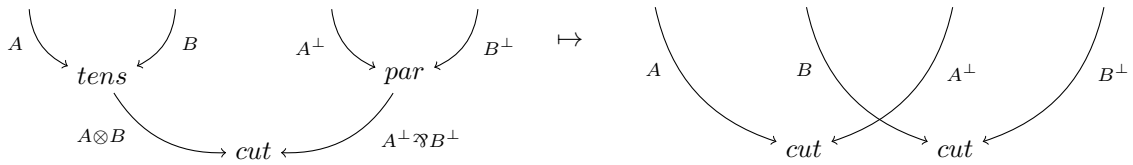
Question 43 Montrer qu'un nœud *cut* est relié d'un côté à un nœud *tens* si et seulement s'il est relié de l'autre côté à un nœud *par*.

Sur une structure de preuve, on définit une étape d'élimination d'une coupure de la façon suivante :

– si un nœud *cut* est relié à un nœud ax, peu importe de quel côté, on supprime ces deux nœuds :



– si un nœud *cut* est relié à un nœud *tens* d'un côté et un nœud *par* de l'autre côté, on le découpe en deux nœuds *cut* de la façon suivante :



L'algorithme d'élimination des coupures consiste à répéter l'étape d'élimination d'une coupure tant que c'est possible.

Question 44 Montrer que si la structure de preuve vérifie le critère de Danos et Régnier initialement alors la structure obtenue à la fin de l'élimination des coupures vérifie le critère de Danos et Régnier (c'est-à-dire que les interrupteurs sont connexes et acycliques).

Question 45 Montrer que l'algorithme d'élimination des coupures termine et que la structure obtenue n'a plus de coupure.

Question 46 Montrer que la règle (cut) est admissible en logique linéaire n'utilisant que les règles (ax), ($\parr$) et ($\otimes$).

⋆ ⋆ ⋆