Oral d'Informatique CCMT - Sujet 0

Exercice 1

cf annexe pour un rappel des règles de la déduction naturelle

1.1 Prouver le séquent $A \wedge B \vdash B \wedge A$

1.2 Prouver le séquent $A \wedge (B \vee C) \vdash (A \wedge B) \vee (A \wedge C)$

Soit $\Gamma \vdash C$ un séquent prouvable à l'aide d'un arbre de preuve Π.

1.3 Montrer qu'il existe un arbre de preuve de $\Gamma \vdash C$ tel que cet arbre ne possède pas la succession de la règle élimination de la conjonction puis introduction de la conjonction.

1.4 Que peut-on dire pour les successions de règles de disjonction ?

1.5 Prouver le séquent $\vdash A \vee \neg A$.

* * *

Exercice 2

On s'intéresse à un bus touristique pouvant contenir C passagers.
On suppose que l'on a $n > C$ passagers qui attendent leur tour, puis se remettent en attente à l'arrêt du bus dès qu'ils ont fini pour le revoir.
Le bus ne démarre que lorsqu'il est plein.
Pour formaliser ce problème, on utilise des fonctions fictives :

- `board` et `unboard` permettent au passager de monter et de descendre ;
- le bus doit appeler les fonctions `load` lorsqu'il se remplit, `run` lorsqu'il démarre son tour et `unload` lorsqu'il se vide.

Attention, les passagers ne peuvent pas descendre avant que le bus ait ouvert ses portes pour une fin de tour avec `unload` et ne peuvent pas monter avant que le bus ait ouvert ses portes pour un nouveau tour avec `load`.

2.1 Écrire les fonctions en pseudo-code correspondant au bus et aux passagers **sans** prendre en compte les problèmes de synchronisation dans un premier temps.

2.2 Proposer une solution en pseudo-code utilisant deux compteurs protégés par des mutexs et quatre sémaphores.

2.3 Votre solution peut-elle être utilisée dans le cas où l'on a plusieurs bus ? Justifier.

2.4 Proposer une nouvelle solution pour le pseudo-code du bus dans le cas où il y a m bus numérotés en respectant les règles suivantes :

- un seul bus peut ouvrir ses portes aux passagers à la fois ;
- plusieurs bus peuvent faire un tour en même temps ;
- les bus ne peuvent pas se doubler donc ils se chargent et se déchargent toujours dans le même ordre ;
- un bus doit avoir fini de décharger avant qu'un autre bus vienne décharger.

La solution doit utiliser deux tableaux de sémaphores en plus des sémaphores utilisés dans la solution précédente, permettant de gérer la coordination entre les bus.

* * *

Annexe 1 - Sujet 0

Rappel des règles de la déduction naturelle

Les arbres de preuves doivent être effectués à partir de l'ensemble de règles fourni ci-dessous.

$$\frac{}{\Gamma, A \vdash A}\,\text{ax} \qquad \frac{\Gamma \vdash B}{\Gamma, A \vdash B}\,\text{aff} \qquad \frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}\,\text{RAA}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\,{\to_i} \qquad \frac{\Gamma \vdash A \to B \quad \Gamma \vdash A}{\Gamma \vdash B}\,{\to_e}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\,{\wedge_i} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\,{\wedge_e^g} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\,{\wedge_e^d}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\,{\vee_i^g} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\,{\vee_i^d} \qquad \frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}\,{\vee_e}$$

$$\frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}\,{\neg_i} \qquad \frac{\Gamma \vdash \neg A \quad \Gamma \vdash A}{\Gamma \vdash \bot}\,{\neg_e}$$

Oral d'Informatique CCMT - Sujet 1

Exercice 1

Soit G un graphe non-orienté à $n \geq 1$ sommets et p arêtes.

1.1 Montrer que si G est connexe alors $p \geq n - 1$.

1.2 Montrer que si G est acyclique alors il possède un sommet de degré au plus 1.

1.3 Montrer que si G est acyclique alors $p \leq n - 1$.

1.4 Montrer que les trois propriétés suivantes sont équivalentes :

(i) G est connexe et acyclique.

(ii) G est connexe et $p = n - 1$.

(iii) G est acyclique et $p = n - 1$.

* * *

Exercice 2

On considère le problème suivant :

SUBSET-SUM

Entrée : Un tableau $T = [t_1, ..., t_n]$ de n entiers positifs et un entier c.
Sortie : La valeur maximum de $\sum_{i \in I} t_i$ où $I \subseteq \{1, ..., n\}$ et $\sum_{i \in I} t_i \leq c$.

2.1 Décrire un algorithme naïf qui résout ce problème et préciser sa complexité.

2.2 On note $s_{i,j}$ la plus grande somme inférieure à j que l'on peut obtenir avec des éléments $t_1, \ldots, t_i$.
Donner une équation de récurrence pour $s_{i,j}$ et en déduire un algorithme pour SUBSET-SUM. Comparer sa complexité avec l'algorithme précédent.

On considère l'algorithme glouton suivant :

```
Trier les éléments de T par ordre décroissant.
S ← 0
Pour i de 1 à n :
    Si S + t_i ≤ c :
        S ← S + t_i
Renvoyer S
```

2.3 Donner la complexité de cet algorithme.

2.4 Montrer que l'algorithme glouton donne une $\frac{1}{2}$-approximation de la solution optimale.

2.5 Soit $\alpha \in]\frac{1}{2}, 1]$. Donner une instance de SUBSET-SUM telle que la somme S renvoyée par l'algorithme glouton vérifie $S \leq \alpha S^*$ où S^* est la solution optimale.

* * *

Oral d'Informatique CCMT - Sujet 2

Exercice 1

Une entreprise de livraison de nourriture dispose d'une base de donnée pour représenter ses clients et les commandes passées. On présente le schéma relationnel correspondant.

Clients(id : `int`, id_addresse : `int`, nom : `text`, prenom : `text`)

Adresses(id : `int`, ville : `text`, nom_rue : `text`, numero : `int`)

Commandes(id_client : `int`, plat : `text`, prix : `int`,date : `date`)

L'entreprise souhaite ajouter une fonctionnalité pour vérifier qu'un client n'est pas allergique à un plat qu'il commande. Pour cela, il est nécessaire de pouvoir enregistrer les allergènes présents dans chaque plat ainsi que les allergies de chaque client.

1.1 Proposer des modifications à notre schéma de relation pour pouvoir ajouter ces informations. Justifier pourquoi votre choix convient.

Dans la suite, on se base dans le modèle tel que vous l'avez modifié.

1.2 Écrire une requête pour trouver les noms et prénoms des clients étant allergiques à au moins un ingrédient d'une pizza.

1.3 Écrire une requête pour trouver les trois villes où le plus d'argent est dépensé, ainsi que cette somme totale.

1.4 Écrire une requête pour effectuer la moyenne d'argent dépensé en fonction des prénoms des clients.

* * *

Exercice 2

Dans cet exercice, on ne considère que des graphes orientés. Soit $G = (S, A)$ un graphe, on appelle clôture transitive d'un graphe, qu'on note $G^* = (S, A^*)$, le graphe ayant les mêmes sommets que S et tel que (x, y) soit une arête de G^* si et seulement si il existe un chemin de x à y dans G. On note $|S| = n$.

2.1 Justifier que la relation $\mathcal{R}$ définie par $x\mathcal{R}y$ si et seulement si (x, y) appartient à A^* est transitive.

2.2 On suppose que A est donné sous la forme d'une matrice d'adjacence. Donner un algorithme pour calculer A^* en $O(n^3)$ opérations.

Une réduction transitive d'un graphe $G = (S, A)$ est un sous-graphe $G_R = (S, A_R)$ avec un nombre minimum d'arêtes tel que $G^* = G_R^*$.

2.3 Montrer que dans le cas général, une réduction transitive n'est pas unique.

Dans toute la suite, on ne considérera plus que des graphes **acyclique**.

Considérons pour tout x, y dans S $A_{x,y}$ définie par :

$$A_{x,y} = \{e \in A \mid e \text{ appartient à un chemin de longueur maximal entre } x \text{ et } y\}$$

De plus on considère

$$A' = \bigcup_{x,y \in S} A_{x,y}$$

2.4 Montrer l'égalité suivante :

$$A' = \{(x, y) \in A \mid \text{ la longueur du plus long chemin entre } x \text{ et } y \text{ est } 1\}$$

2.5 En déduire que $G' = (S, A')$ est l'unique réduction transitive de G.

2.6 Donner un algorithme pour calculer l'unique réduction transitive d'un graphe acyclique.

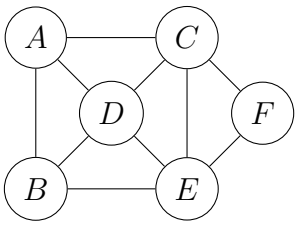
* * *

Oral d'Informatique CCMT - Sujet 3

Exercice 1

1.1 Rappeler la définition de couplage dans un graphe non orienté. Quand peut-on qualifier un couplage de maximal ? maximum ? parfait ?

1.2 Donner un couplage maximum dans le graphe suivant. Donner un couplage maximal qui n'est pas maximum.



1.3 Qu'est ce qu'un chemin augmentant alternant pour un graphe $G = (S, A)$ et un couplage $C \subset A$ sur ce graphe ?

1.4 Soit $G = (S, A)$ un graphe. Montrer qu'un couplage est maximum si et seulement s'il n'existe pas de chemin augmentant alternant pour ce couplage dans le graphe G.

* * *

Exercice 2

Soit Σ un alphabet.
Pour deux mots u, v dans Σ^*, on appelle entrelacement de u et v, un mot w qui utilise exactement les lettres de u et de v dans leur ordre dans chaque mot.
Par exemple *babaa* est un entrelacement de *bb* et *aaa* ou encore *abcabc* est un entrelacement de *aba* et *cbc*. Mais *bacb* n'est pas un entrelacement de *ab* et *cb* car l'ordre n'est pas respecté.
On peut voir le lien avec les entrelacements des tâches de plusieurs fils d'exécution.

On note $\mathcal{E}(u, v)$ l'ensemble des entrelacements de u et v pour deux mots u, v.

2.1 Donner les entrelacements de *ab* et *cb*.

2.2 Soit u, v deux mots.

1. Majorer grossièrement le cardinal de $\mathcal{E}(u, v)$.
2. Montrer que $\mathcal{E}(u, v)$ est un langage régulier.

2.3 Soit L_1, L_2 deux langages réguliers. Montrer que $\mathcal{E}(L_1, L_2) = \bigcup_{u \in L_1, v \in L_2} \mathcal{E}(u, v)$ est un langage régulier.

2.4 Soit u, v, w trois mots dans Σ^*, proposer un algorithme de programmation dynamique permettant de savoir si $w \in \mathcal{E}(u, v)$. Préciser la complexité.

* * *

Oral d'Informatique CCMT - Sujet 4

Exercice 1

Soit $F = ((a \wedge b) \vee \neg c) \wedge (a \vee \neg b)$

1.1 F est-elle satisfiable ? $\neg F$ est-elle satisfiable ?

1.2 Mettre F en forme normale conjonctive.

1.3 Mettre F en forme normale disjonctive.

1.4 Rappeler le théorème de Cook-Levin.

* * *

Exercice 2

On considère un ensemble de clés $\mathcal{K} \subseteq \mathbb{N}$ fini de taille m. On note $\mathcal{A}_\mathcal{K}$ l'ensemble des arbres binaires dont les noeuds sont étiquetés par des éléments *distincts* de $\mathcal{K}$. Pour $A \in \mathcal{A}_\mathcal{K}$, on notera $\kappa(A)$ l'ensemble des étiquettes des noe uds de A. Par convention, on dira que la racine de A est de profondeur 0.

2.1 Rappeler la définition inductive d'un *arbre binaire de recherche* (ABR).

2.2 Rappeler la complexité dans le pire des cas de la recherche dans un ABR ayant n nœuds.

Soit $A \in \mathcal{A}_\mathcal{K}$ tel que $\kappa(A) = \mathcal{K}$. Soit $\mathbb{P}$ une loi de probabilité quelconque, sur les clés de $\mathcal{K}$. On définit H_A la variable aléatoire qui, à une clé de $\mathcal{K}$, associe sa profondeur dans A. Ainsi $H_A(k)$ est la profondeur de k pour une clé k.

2.3 Soit un ABR fixé $A \in \mathcal{A}_\mathcal{K}$. Rappeler l'expression de la complexité en moyenne de la recherche d'une clé dans A.

2.4 On suppose ici que $\mathbb{P}$ est la loi uniforme sur les clés de $\mathcal{K}$. Quelles sont les particularités des ABR $A \in \mathcal{A}_\mathcal{K}$ pour lesquels $\mathbb{E}(H_A)$ est minimale. Montrer que $\mathbb{E}(H_A) = \mathrm{O}(\log(m))$.

On dit que $A \in \mathcal{A}_\mathcal{K}$ est optimal pour $\mathcal{K}$ et $\mathbb{P}$ si et seulement si $\kappa(A) = \mathcal{K}$ et il minimise $\mathbb{E}(H_A)$.

On cherche un algorithme calculant un ABR optimal étant donné un ensemble de clés fini et une loi de probabilité sur ces clés.

2.5 De quel type de problème algorithmique s'agit-il ?

2.6 Pour cette question, on suppose $\mathcal{K} = \{1, 2, 3\}$, $\mathbb{P}(1) = \frac{2}{3}$, $\mathbb{P}(2) = \mathbb{P}(3) = \frac{1}{6}$. Dessiner un ABR optimal pour $\mathcal{K}$ et $\mathbb{P}$

2.7 Soit $A \in \mathcal{A}_\mathcal{K}$ un ABR optimal non vide. A possède donc un sous-arbre gauche A_g. On se place dans le cas où A_g est non vide. On note $S_g = \sum_{k \in \kappa(A_g)} \mathbb{P}(k)$ et $\mathbb{P}_g = \frac{1}{S_g}\mathbb{P}$. Montrer que A_g est optimal pour $\kappa(A_g)$ et $\mathbb{P}_g$. Montrer le résultat analogue pour le sous-arbre droit.

2.8 Proposer un algorithme de programmation dynamique pour trouver un ABR optimal.

2.9 Quel est sa complexité ?

* * *