Chaque question peut être traitée en admettant les résultats des questions précédentes ce qui permet de les aborder dans un ordre différent de leur ordre d'apparition dans le sujet, en particulier les questions suivies d'un astérisque sont plus difficiles et pourront être traitées en dernier.

Dans ce sujet, un graphe désigne un couple $\mathcal{G} = (S, A)$ dans lequel S est l'ensemble des sommets et A est l'ensemble des arêtes du graphe. Une arête est une paire $\{s_i, s_j\}$ de sommets, c'est à dire que s_i et s_j sont deux éléments distincts de S. Un tel graphe est dit fini s'il n'admet qu'un nombre fini de sommets (et donc également un nombre fini d'arêtes). Un chemin entre deux sommets s et s' est une suite d'arêtes $(\{s_i, s'_i\}, 0 \leq i \leq n)$ telles que $s = s_0$, $s' = s'_n$ et $s'_i = s_{i+1}$ pour tout $0 \leq i < n$. Un graphe est dit connexe s'il existe au moins un chemin entre chaque paire de sommets. Un sommet s' tel que $\{s; s'\} \in A$ est dit un voisin de s. Le voisinage d'un sommet s est l'ensemble $V(s)$ constitué de s et de l'ensemble des voisins de s.

Un tel graphe nous sert à représenter un réseau de processeurs communicants : chaque sommet figure un processeur et chaque arête un canal bidirectionnel reliant les processeurs associés à ses sommets. Dans la suite *un réseau désigne un graphe connexe fini ayant au moins deux sommets distincts* (donc chaque sommet admet au moins un voisin). Un étiquetage est une application qui associe une étiquette $\ell \in L$ à chaque sommet du réseau. Un élément de L sera parfois utilisé pour représenter un état local, c'est à dire l'état d'un processeur, dans ce cas un tel étiquetage sera appelé état du réseau (relatif à L).

Un protocole $\mathcal{P} = (L, R)$ consiste en un ensemble L d'états locaux et en un ensemble R de règles de la forme $r = ((\ell'_0, \ell'_1, \ldots, \ell'_k)(\ell''_0, \ell''_1, \ldots \ell''_k))$ où les ℓ'_i et les ℓ''_i sont des états locaux; c'est à dire que R est un sous ensemble de $\bigcup_{k \geq 1} L^k \times L^k$. On dit que l'état σ'' d'un réseau $\mathcal{R}$ dérive de l'état σ' par application de la règle $r = ((\ell'_0, \ell'_1, \ldots, \ell'_k)(\ell''_0, \ell''_1, \ldots \ell''_k))$ du protocole $\mathcal{P}$ au sommet s du réseau, ce qu'on note $\sigma' - r/s \to \sigma''$ (le réseau et le protocole étant implicites), si on peut trouver une énumération $V(s) = \{s_0, s_1, \ldots, s_k\}$ du voisinage de s avec $s = s_0$ telle que $\sigma'(s_i) = \ell'_i$ et $\sigma''(s_i) = \ell''_i$ pour $0 \leq i \leq k$ et $\sigma''(s') = \sigma'(s')$ pour $s' \notin V(s)$. C'est à dire que chaque processeur peut changer son état local et celui de ses voisins lorsque l'état de son voisinage est conforme à une des règles du protocole. Une exécution du protocole sur le réseau à partir d'un état σ_0 est une suite de dérivations $(\sigma_i - r_i/s_i \to \sigma'_i, i \in I)$ avec $I = \{0, \ldots, m\}$ ou $I = \mathbb{N}$, telle que $\sigma'_i = \sigma_{i+1}$ pour tout $i \in I \setminus \{0\}$. Une exécution est dite finie si I est de la forme $\{0, \ldots, m\}$ et elle est dite terminée si de plus aucune dérivation n'est possible en σ'_m cet état est alors appelé résultat de l'exécution.

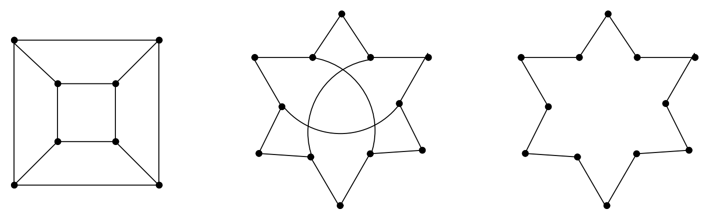
$\mathcal{P} = (L, R)$ est un protocole d'énumération pour le réseau $\mathcal{R} = (S, A)$ où $S = \{s_1, \ldots, s_n\}$, si L est de la forme $L = \mathbb{N} \times L'$ et s'il existe un état local $\ell_0 \in L$ tel que toute exécution à partir de l'état σ_0 tel que $\sigma_0(s) = \ell_0$ pour tout $s \in S$, se termine et le résultat de cette exécution est un état σ tel que $\sigma(s_i) = (n_i, \ell'_i)$ avec $\{n_i \mid 1 \leq i \leq n\} = \{1, \ldots n\}$; c'est à dire que toute exécution du protocole à partir de σ_0 termine et fournit une énumération des processus du réseau.

Le but du sujet est de montrer que pour une classe de réseaux, dits ambigus, il n'existe pas de protocole d'énumération et qu'il existe un protocole qui est un protocole d'énumération pour tous les réseaux non ambigus.

1 Réseaux ambigus

Un étiquetage $\lambda : S \to E$ d'un réseau $\mathcal{R} = (S, A)$ est dit *localement bijectif* si *(i)* il est injectif au voisinage de chaque sommet : $\forall s \in S \quad \forall s', s'' \in V(s) \quad \lambda(s') = \lambda(s'') \Rightarrow s' = s''$ et *(ii)* deux noeuds similaires (i.e. ayant la même valeur par λ) ont des voisinages similaires : $\forall s', s'' \in S \quad \lambda(s') = \lambda(s'') \Rightarrow \lambda(V(s')) = \lambda(V(s''))$ (où $\lambda(X) = \{\lambda(s) | \ s \in X\}$ pour $X \subseteq S$). Un réseau est dit *ambigu* s'il admet un étiquetage localement bijectif qui n'est pas injectif. Notons $\mathrm{sim}_\lambda(s) = \{s' \in S \mid \lambda(s') = \lambda(s)\}$ (respectivement $\mathrm{sim}_\lambda(a) = \{a' \in A \mid \lambda(a') = \lambda(a)\}$) l'ensemble des sommets (resp. des arêtes) similaires au sommet s (resp. à l'arête a) vis à vis de λ.

1.1 Montrer que les réseaux suivants sont ambigus.



1.2 Montrer que si $\lambda : S \to E$ est un étiquetage localement bijectif d'un réseau $\mathcal{R} = (S, A)$ alors

1.2.1 deux sommets similaires ont des voisinages disjoints et deux arêtes similaires sont disjointes.

1.2.2 il existe un entier k, appelé ordre de la bijection locale, tel que $\forall s \in S \quad \forall a \in A \quad |\mathrm{sim}_\lambda(s)| = |\mathrm{sim}_\lambda(a)|$ où $|X|$ désigne le nombre d'éléments de l'ensemble X. Que peut on dire d'un réseau dont toutes les bijections locales sont d'ordre 1 ?

1.3 Montrer qu'un réseau complet $\mathcal{R} = (S, A)$, c'est à dire tel que $A = \{\{s_1, s_2\} \mid s_1, s_2 \in S\}$, est non ambigu.

1.4 Montrer qu'un réseau ayant un nombre premier de sommets ou un nombre premier d'arêtes est non ambigu.

1.5 Un réseau $\mathcal{R} = (S, A)$ est un arbre si pour toute paire de sommets il existe un et un seul chemin élémentaire (c'est à dire dans lequel une même arête apparait au plus une fois) les joignant. On rappelle que dans ce cas $|A| = |S| - 1$. Montrer qu'un tel réseau est non ambigu.

1.6 Soient $s_1, \ldots, s_n$ des sommets ayant des voisinages disjoints et $r_1, \ldots, r_n$ des règles d'un protocole telles que l'on ait des dérivations $\sigma - r_i/s_i \to \sigma_i$, montrer que pour toute permutation τ de $\{1, \ldots, n\}$ on peut trouver une suite de dérivations $\sigma - r_{\tau(1)}/s_{\tau(1)} \to \ \sigma_{\tau,1} - r_{\tau(2)}/s_{\tau(2)} \to \ \sigma_{\tau,2} - r_{\tau(3)}/s_{\tau(3)} \to \ldots$ $\sigma_{\tau,n-1} - r_{\tau(n)}/s_{\tau(n)} \to \sigma_{\tau,n}$ de sorte que l'état $\sigma_{\tau,n}$ soit indépendant de la permutation τ, notons le σ_n. On dit dans ce cas que les dérivations $\sigma - r_i/s_i \to \sigma_i$ sont indépendantes en l'état s et on note $\sigma - \{r_i/s_i\}_{1\leq i \leq n} \to \sigma_n$ l'ensemble de ces suites de dérivations.

1.7 Soit $\lambda : S \to E$ un étiquetage localement bijectif d'un réseau $\mathcal{R} = (S, A)$, un état $\sigma : S \to L$ est dit λ-compatible si $\lambda(s) = \lambda(s') \Rightarrow \sigma(s) = \sigma(s')$. Montrer que si σ est un état λ-compatible et $\sigma - r/s \to \sigma'$ une dérivation alors on a une dérivation parallèle $\sigma - \{r/s' |\ s' \in \mathrm{sim}_\lambda(s)\} \to \sigma''$ avec σ'' un état λ-compatible.

1.8 Montrer qu'il n'existe pas de protocole d'énumération pour un réseau ambigu.

2 Un protocole d'énumération universel

2.1 Préliminaires

Soit X un ensemble fini. Une application $f : X \to I\!N$ est dite *connexe* si son image est un segment initial de $I\!N \setminus \{0\}$, c'est à dire $\forall x \in X \quad f(x) > 1 \Rightarrow \exists y \in X \quad f(y) = f(x) - 1$. L'ensemble $\mathcal{C}(X)$ des fonctions connexes sur X est muni de la relation d'ordre point à point : $f \leq g \Leftrightarrow \forall x \in X \quad f(x) \leq g(x)$. La fonction constante égale à 0 est le plus petit élément de $\mathcal{C}(X)$.

2.1.1 Montrer que les éléments maxima de $\mathcal{C}(X)$ sont les énumérations de X, c'est à dire les applications connexes $f : X \to I\!N$ injectives.

2.1.2 Si Y est une partie finie de $I\!N$, $\max(Y)$ désigne le plus grand entier se trouvant dans Y avec par convention $\max(\emptyset) = 0$. Si $f \in \mathcal{C}(X)$ et $Y \subseteq X$ et $x_0, x_1 \in X$ tels que $x_0 \neq x_1$ et $f(x_0) = 0$ ou $f(x_0) = f(x_1)$ montrer que l'application g définie par $g(x) = \begin{cases} \max\{f(y) |\ y \in Y\} + 1 & \text{si } x = x_0 \\ f(x) & \text{sinon} \end{cases}$ est connexe.

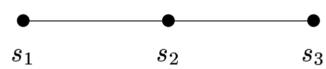
2.1.3 $(*)$ Montrer que la relation $A \prec B \Leftrightarrow \max(A \setminus B) < \max(B \setminus A)$ est une relation d'ordre totale stricte sur l'ensemble des parties finies de $I\!N$. On notera $\preceq$ l'ordre large correspondant : $A \preceq B \Leftrightarrow (A \prec B) \vee (A = B)$.

2.2 Le protocole

L'ensemble des états locaux du protocole est $L = I\!N \times 2^{I\!N} \times 2^{(I\!N \times 2^{I\!N})}$ où 2^X désigne l'ensemble des parties de l'ensemble X : $2^X = \{Y | Y \subseteq X\}$. Un état local est donc un triplet $\ell = (n, R, B)$ dans lequel n est un entier appelé *identificateur*, R est un ensemble d'entiers appelé le *registre* et enfin B appelé la *boîte aux lettres* est un ensemble de *messages*, un message étant une paire constituée d'un entier identifiant l'origine du message et du message proprement dit qui est un ensemble d'entiers. L'ensemble R des règles du protocole est défini comme suit. $r = ((\ell'_0, \ell'_1, \ldots, \ell'_k)(\ell''_0, \ell''_1, \ldots \ell''_k)) \in R$ si et seulement si $\ell'_i = (n'_i, R'_i, B'_i)$ et $\ell''_i = (n''_i, R''_i, B''_i)$ sont tels que ou bien $\exists i \in \{1, \ldots k\} \quad B'_i \neq B'_0$ et $\forall i \in \{1, \ldots k\} \quad (n''_i = n'_i \wedge R''_i = R'_i \wedge B''_i = \bigcup_{0 \le j \le k} B'_j)$, ou bien $\forall i \in \{1, \ldots, k\} \quad B'_i = B'_0$ et $n'_0 = 0 \vee (n'_0 > 0 \wedge \exists R \quad (n'_0, R) \in B'_0 \wedge R'_0 \prec R)$ et $n''_0 = 1 + \max\{n |\ \exists R \quad (n, R) \in B'_0\}$ et $n''_i = n'_i$ pour $1 \le i \le k$ et $R''_0 = R'_0$ et $R''_i = (R'_i \setminus \{n'_0\}) \cup \{n''_0\}$ pour $1 \le i \le k$ et $B''_i = B'_i \cup \{(n''_j, R''_j) |\ 0 \le j \le k\}$ pour $0 \le i \le k$. Autrement dit ou bien les boîtes à lettres des sommets dans le voisinage du sommet considéré n'ont pas le même contenu, auquel cas on unifie le contenu de ces boîtes à lettres qui sont alors mises à jour (chacune d'entre elles est remplacée par leur union) ou bien toutes ces boîtes à lettres ont le même contenu et alors si l'identificateur du sommet considéré est 0 ou si sa boîte à lettre contient un message dont le contenu est strictement plus grand que le registre de ce sommet et provenant d'un sommet ayant le même identificateur que lui, alors on modifie l'identificateur de ce sommet en lui attribuant la valeur $k+1$ où k est le plus grand identificateur apparaissant dans sa boîte à lettres et on

répercute cette modification dans les registres et les boîtes à lettres de ses voisins ainsi que dans sa propre boîte à lettres. On pose σ_0 l'état qui attribue la valeur $(0, \emptyset, \emptyset)$ à tous les sommets du réseau.
Remarquons qu'il existe au plus une règle r de ce protocole pouvant s'appliquer en un sommet donné s d'un réseau donné dans un état σ donné. On notera alors simplement $\sigma - s \rightarrow \sigma'$ pour signifier que l'état σ' dérive de l'état σ par l'application d'une règle de ce protocole au sommet s.

2.2.1 Indiquer la valeur des états σ_1 à σ_5 du réseau



rencontrés lors de l'exécution suivante du protocole

$$\sigma_0 - s_1 \rightarrow \sigma_1 - s_2 \rightarrow \sigma_2 - s_3 \rightarrow \sigma_3 - s_2 \rightarrow \sigma_4 - s_2 \rightarrow \sigma_5$$

Cette exécution est elle terminée ?

2.3 Soit $\mathcal{R} = (S, A)$ un réseau et $\sigma_0 - s_0 \rightarrow \ \sigma_1 - s_1 \rightarrow \ldots \ \sigma_i - s_i \rightarrow \sigma_{i+1} - s_{i+1} \rightarrow \ldots$ une exécution de ce protocole à partir de l'état σ_0, on pose $\sigma_i(s) = (n_i(s), R_i(s),\ B_i(s))$ et $N_i(s) = \{k | \ \exists R \quad (k, R) \in B_i(s)\}$ c'est à dire que $N_i(s)$ est l'ensemble des identificateurs "connus" par le sommet s en l'état σ_i.

2.3.1 Montrer que $n_i(s) \leq n_{i+1}(s)$, $R_i(s) \preceq R_{i+1}(s)$ et $B_i(s) \subseteq B_{i+1}(s)$.

2.3.2 $(*)$ Montrer que si $n_i(s) = n > 0$, alors il existe un $s' \in S$ tel que $n_{i+1}(s') = n$ et $R_i(s) \preceq R_{i+1}(s')$.

2.3.3 Montrer que si $s' \in V(s)$ et $n_i(s') = n > 0$, alors $n \in N_i(s)$.

2.3.4 $(*)$ Montrer que si $s', s'' \in V(s)$ et $n_i(s') = n_i(s'') > 0$, alors $s' = s''$.

2.3.5 $(*)$ Montrer que $R_i(s) = n_i(V(s) \setminus \{s\}) \setminus \{0\}$.

2.3.6 Montrer que l'application n_i est connexe et que $n_i(s) \leq |S|$.

2.3.7 Montrer que toute exécution du protocole à partir de l'état σ_0 se termine.

2.3.8 Si σ_m est le résultat d'une exécution du protocole à partir de l'état σ_0, montrer que pour $s, s', s'' \in S$ on a *(i)* $B_m(s') = B_m(s'')$, *(ii)* $n_m(s) > 0$, *(iii)* $R_m(s) = n_m(V(s) \setminus \{s\})$, *(iv)* $n_m(s') = n_m(s'') \Rightarrow n_m(V(s') \setminus \{s'\}) = n_m(V(s'') \setminus \{s''\})$, et *(v)* n_m est localement bijective.

2.3.9 Montrer que ce protocole est un protocole d'énumération pour tout réseau non ambigu.