MINISTÈRE
DE L'ÉDUCATION
NATIONALE
ET DE LA JEUNESSE
Liberté
Égalité
Fraternité

EAE INF 1

SESSION 2023

AGREGATION
CONCOURS EXTERNE

Section : INFORMATIQUE

COMPOSITION EN INFORMATIQUE

Durée : 5 heures

L'usage de tout ouvrage de référence, de tout dictionnaire et de tout matériel électronique (y compris la calculatrice) est rigoureusement interdit.

Il appartient au candidat de vérifier qu'il a reçu un sujet complet et correspondant à l'épreuve à laquelle il se présente.

Si vous repérez ce qui vous semble être une erreur d'énoncé, vous devez le signaler très lisiblement sur votre copie, en proposer la correction et poursuivre l'épreuve en conséquence. De même, si cela vous conduit à formuler une ou plusieurs hypothèses, vous devez la (ou les) mentionner explicitement.

NB : Conformément au principe d'anonymat, votre copie ne doit comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé consiste notamment en la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de la signer ou de l'identifier.
Le fait de rendre une copie blanche est éliminatoire.

Tournez la page S.V.P.

INFORMATION AUX CANDIDATS

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie.

Ces codes doivent être reportés sur chacune des copies que vous remettrez.

Concours	Section/option	Epreuve	Matière
EAE	6200A	101	9492

IMPRIMERIE NATIONALE – 23 0103 – D'après documents fournis

Dépendances. Ce sujet contient trois parties indépendantes qui doivent être traitées toutes les trois. On veillera à bien indiquer sur la copie les changements de partie.

Attendus. Il est attendu des candidates et des candidats des réponses construites. Ils seront aussi évalués sur la précision, le soin et la clarté de la rédaction.

Partie I. Provenance en bases de données

Cette partie vise à introduire une notion d'explication pour l'évaluation d'une requête sur une base de données. Un exemple de base de données est présenté, puis la notion de provenance booléenne pour les requêtes, et celle de formule booléenne pour la représentation de la provenance. Pour finir, on s'intéresse à comment calculer la provenance à partir d'une requête exprimée en algèbre relationnelle puis en SQL.

1 Exemple de base de données

1.1 Présentation du schéma de la base de données

Voici une base de données d'une société organisant des croisières et son schéma.

Croisiere(*cid* : string, depart : string, arrivee : string, distance : integer)

Bateau(*bid* : string, type : string, autonomie : integer)

Navigable(*bid* : string, *eid* : string)

Employe(*eid* : string, nom : string, age : integer)

La relation *Employe* contient tous les employés de la compagnie dont les marins pilotant les bateaux. La relation *Navigable* indique pour chaque bateau quels sont les marins pouvant piloter ce bateau. Dans la relation *Navigable*, l'attribut *eid* est une clef étrangère provenant de la relation *Employe* et l'attribut *bid* est une clef étrangère provenant de la relation *Bateau*. Un marin est un employé qui peut piloter au moins un bateau. Les clefs primaires d'une relation sont les attributs qui sont soulignés. Les distances et les autonomies sont exprimées en nœuds.

Question 1.1. Écrire en SQL les commandes pour créer le schéma présenté dans ce paragraphe.

1.2 Requêtes

Certaines requêtes sont à écrire en SQL seulement et d'autres en SQL *et* algèbre relationnelle.

Question 1.2. Écrire la requête suivante en SQL et en algèbre relationnelle : donner les marins qui peuvent piloter des catamarans. Renvoyer les identifiants de ces marins.

Question 1.3. Écrire la requête suivante en SQL et en algèbre relationnelle : donner les identifiants des bateaux et les identifiants de croisières tels que les bateaux peuvent effectuer les croisières sans réapprovisionnement.

Question 1.4. Écrire la requête suivante en SQL et en algèbre relationnelle : donner l'identifiant des bateaux pouvant être pilotés par tous les marins d'un âge strictement supérieur à 40 ans.

Question 1.5. Écrire la requête suivante en SQL et en algèbre relationnelle : donner l'identifiant des marins qui peuvent piloter des bateaux pouvant parcourir strictement plus de 3000 nœuds mais qui ne peuvent pas piloter des catamarans.

Question 1.6. Écrire la requête suivante en SQL et en algèbre relationnelle : donner l'identifiant des employés les plus âgés.

Question 1.7. Écrire la requête suivante en SQL : donner le nombre de bateaux dans la base de données.

Question 1.8. Écrire la requête suivante en SQL : pour chaque bateau, renvoyer son identifiant et le nombre de marins qui peuvent le piloter.

Question 1.9. Écrire la requête suivante en SQL : donner l'âge moyen des marins.

2 Introduction à la provenance

Dans le cadre des bases de données, il est intéressant de pouvoir expliquer les réponses de l'évaluation des requêtes sur une base de données. Pour expliquer les réponses, il existe plusieurs méthodes dont la provenance booléenne.

2.1 Requêtes booléennes

Définition 1 *Une requête booléenne est une requête qui renvoie un enregistrement fixe, en général un* 0*-uplet, si la requête est vraie et qui ne renvoie rien sinon.*

Pour information, il est possible d'écrire une requête qui renvoie un 0-uplet en ne mettant rien dans la partie SELECT. Ainsi la requête SELECT FROM R renvoie autant de fois le 0-uplet qu'il y a d'enregistrements dans la relation R.

Question 1.10. Soit Q une requête. Indiquer comment transformer Q en une requête booléenne Q' telle que Q' renvoie un enregistrement constant (par exemple 1) sur la base de donnée D si et seulement si Q renvoie au moins un enregistrement lorsqu'évaluée sur D. Expliquer cette transformation dans le cadre de SQL et de l'algèbre relationnelle. Appliquer votre méthode pour la requête de la question 1.2 à ses formulations en SQL et algèbre relationnelle.

Définition 2 *Soit Q une requête booléenne et soit D une base de données. La* provenance booléenne *de Q évaluée D et dénotée* $\mathrm{Pr}(Q, D)$ *est l'ensemble des sous-bases de données de D qui satisfont la requête Q.*

Exemple 1 *Nous prenons la base de données avec le schéma suivant : il y a une relation R qui a deux attributs A et B. L'attribut A est de type* VARCHAR *et B de type* INTEGER. *Considérons la requête Q_1 renvoyant le 0-uplet s'il existe un enregistrement ayant la valeur 3 pour l'attribut B. La base de donnée est égale à $\{R(a,4); R(b,3); R(c,3)\}$. La provenance de l'évaluation de la requête sur cette base de donnée est égale à $\{\{R(b,3)\}; \{R(c,3)\}; \{R(b,3), R(c,3)\}; \{R(b,3), R(a,4)\}; \{R(c,3), R(a,4)\}; \{R(b,3), R(c,3), R(a,4)\}\}$.*

2.2 Provenance pour des requêtes non booléennes

Il est possible de définir la notion de provenance pour des requêtes non booléennes. C'est une généralisation de la provenance pour les requêtes booléennes. Soit f un enregistrement appartenant à la réponse de Q appliquée à la base de donnée D. La provenance booléenne de f est l'ensemble des sous-bases de données pour lesquelles f appartient aux réponses de D.

Exemple 2 *Nous prenons le schéma et la base de données de l'exemple 1. La requête Q_2 renvoie l'attribut A des enregistrements ayant leur valeur de l'attribut B égale à 3. L'enregistrement b est une réponse de Q_2 évaluée sur D. Sa provenance est égale à $\{\{R(b,3)\}; \{R(b,3), R(c,3)\}; \{R(b,3), R(a,4)\}; \{R(b,3), R(c,3), R(a,4)\}\}$. L'autre réponse de Q_2 évaluée sur D est c et sa provenance est $\{\{R(c,3)\}; \{R(b,3), R(c,3)\}; \{R(c,3), R(a,4)\}; \{R(b,3), R(c,3), R(a,4)\}\}$.*

2.3 Calcul de la provenance pour notre exercice

Base de donnée pour l'évaluation Nous présentons une très petite base de données correspondant au schéma précédent.

- Croisiere(c1,NY,L,5000) ; Croisiere(c2,L,V,5)
- Bateau(b1,Catamaran,2000) ; Bateau(b2,Yacht,10000) ; Bateau(b3,Kayak,10)
- Navigable(b1,e2) ; Navigable (b2,e2) ; Navigable(b3,e1)
- Employe(e1,Bob,20) ; Employe (e2,Alice, 42)

Question 1.11. Évaluer les requêtes des questions 1.2 et 1.9 de la sous-section 1.2 sur la base de données exemple et indiquer la provenance associée pour chaque réponse. Vous pouvez utiliser les valeurs de la clef primaire pour représenter un enregistrement.

Tournez la page S.V.P.

3 Représentation compacte d'un ensemble : formules booléennes

Représenter l'ensemble des ensembles de sous-instances satisfaisant une requête booléenne peut être verbeux. Nous présentons une méthode pour représenter ces ensembles de façon plus compacte.

Soit E un ensemble fini d'éléments. Il existe une méthode pour représenter les ensembles d'ensembles d'éléments efficacement. Soit X_E, un ensemble de variables en bijection avec E. Une valuation de X_E est une fonction de X_E dans $\{\top, \bot\}$. Il existe une bijection entre les valuations et les ensembles de 2^E définie comme suit : soit P un sous-ensemble appartenant à 2^E et ν_P la valuation associée a P telle que, pour que chaque variable x_e et l'élément associé e, on a e appartient à P si et seulement si $\nu(x_e)$ est égale à $\top$. Nous pouvons remarquer qu'un ensemble de valuations peut être décrit par une formule booléenne. En décrivant l'ensemble des valuations, une formule booléenne peut décrire un ensemble d'ensembles d'éléments.

Dans cet exercice, les formules booléennes n'utilisent que les opérateurs $\wedge, \vee, \neg$.

Question 1.12. Soit E un ensemble $\{e_1, e_2\}$. On utilise les variables x_1 et x_2.

1. Écrire une formule booléenne qui représente l'ensemble $\{\{e_1, e_2\}\}$.
2. Écrire une formule booléenne qui représente l'ensemble $\{\{e_1\}, \{e_2\}\}$.

Il est intéressant de noter que l'utilisation d'une formule booléenne peut être bien plus compacte qu'une représentation d'un ensemble d'ensembles. Plus précisément, il peut y avoir un gain exponentiel dans la représentation d'un ensemble d'ensembles avec une formule booléenne. La taille d'un ensemble est la somme des tailles de ces éléments. Pour un élément simple, c'est-à-dire dans E, la taille est égale à 1. Dans le cadre d'un ensemble d'ensembles d'éléments simples, la taille est égale à la somme des tailles de chaque ensemble contenu dans l'ensemble d'ensembles. Pour une formule booléenne, sa taille est égale à la taille du nombre de symboles utilisés pour la décrire : les symboles sont $\wedge, \vee, \neg$, les parenthèses ainsi que les variables (chaque variable compte pour un).

Question 1.13. Soit $E = \{e_1, \cdots, e_n\} \cup \{f_1, \cdots, f_n\}$, un ensemble d'éléments de taille $2 \cdot n$. Soit ϕ, l'ensemble des ensembles tel que, quel que soit i entre 1 et n, on a e_i ou f_i qui appartient à l'ensemble mais pas les deux. Démontrer que ϕ est un ensemble de taille exponentielle en n. Exhiber une formule booléenne de taille linéaire en n qui décrit ϕ. Justifier formellement la taille de la formule ainsi que sa correction.

3.1 Utilisation des formules booléennes pour représenter la provenance booléenne

En utilisant le enregistrement que nous pouvons décrire les ensembles d'ensembles par des formules booléennes, nous pouvons ainsi les utiliser dans le cadre de la provenance booléenne de bases de données.

Question 1.14. Exprimer une formule booléenne décrivant la provenance booléenne de la requête de la question 1.2 appliquée à la base de données présentée dans 2.3.

4 Calcul automatique de la provenance

Le but de cette partie est de proposer plusieurs méthodes pour calculer une formule booléenne représentant la provenance booléenne d'un enregistrement.

4.1 Calcul automatique de la provenance au travers de l'algèbre relationnelle

Le but de cette partie est d'utiliser l'algèbre relationnelle pour calculer la provenance. Pour cela, nous associons à chaque enregistrement f d'un résultat d'une sous-requête Q évaluée sur la base de données D une formule qui est notée $\Pr(f, Q, D)$. Nous devons associer à chaque enregistrement une variable qui sera le n-uplet constitué des valeurs des clefs primaires des relations.

Tout d'abord, nous expliquons comment décrire la provenance d'une réponse à partir de la provenance du produit cartésien, de la projection et de la sélection.

- Pour le produit cartésien, la provenance d'un enregistrement dans la réponse est la conjonction de la provenance des deux enregistrements produisant ce tuple.
- La provenance d'un enregistrement provenant d'une sélection est la provenance de l'enregistrement sélectionné.
- La provenance d'un enregistrement f obtenu par la projection d'un ensemble d'attributs est la somme des provenances des enregistrements donnant f après projection.
- La provenance d'un enregistrement f d'une requête de la forme $Q_1 - Q_2$ est égale à $\Pr(f, Q_1) \wedge \neg \Pr(fQ_2)$.

Exemple 3 *Nous prenons l'exemple 2. La formule algébrique de la requête est* $\Pi_{R.A}(\sigma_{R.B=3}(D))$. *La provenance* $\Pr(b, Q, D)$ *est égale à* $R(b, 3)$.

Question 1.15. Décrire avec les détails les calculs de provenance associés aux expressions algébriques de vos requêtes en algèbre relationnelle des questions 1.3 et 1.6.

4.2 Monotonie de la requête et monotonie de la provenance

Dans le cadre des requêtes, il est possible de définir une notion de requête croissante. Pour cela, nous définissons la notion d'ordre sur les instances. Une base de données D est inférieure à une autre base de données D' si D est incluse dans D'. Une requête Q est croissante si quels que soient D et D' telles que $D \leq D'$, on a $Q(D) \leq Q(D')$.

Question 1.16. Indiquer en justifiant quelles sont les requêtes qui sont croissantes et celles qui ne sont pas dans les requêtes de 1.2 à 1.9.

Nous définissons également une notion d'ordre partiel sur les valuations d'un ensemble de variables X. Soit ν_1 et ν_2 deux valuations d'X. La valuation ν_1 est plus petite que la valuation

Tournez la page S.V.P.

ν_2 si pour chaque variable x dans X, si $\nu_1(x)$ est égale à $\top$ alors $\nu_2(x)$ également. Une formule booléenne φ est croissante si, pour toute paire de valuations ν_1 et ν_2 telles que ν_1 est plus petite que ν_2, si ν_1 satisfait φ alors ν_2 satisfait φ.

Question 1.17. Soit Q une requête booléenne croissante et D une base de données. Démontrer que la provenance de l'évaluation de Q sur D est croissante.

4.3 Calcul de la provenance en SQL

Dans cette partie, nous souhaitons réécrire les requêtes afin de renvoyer une chaîne de caractères représentant une formule booléenne codant la provenance des réponses de la requête évaluée sur la base de données. L'opérateur $\wedge$ est représenté par le caractère * et l'opérateur $\vee$ est représenté par le caractère U. Par exemple la formule $x \wedge y$ aura comme représentation la chaîne de caractères x * y. De même, la formule $x \vee y$ aura comme représentation la chaîne de caractères x U y.

Comme vu précédemment, nous devons associer à chaque enregistrement une variable qui sera construite à partir des valeurs des clefs primaires des relations. Dans la suite de cette sous-section, nous supposons que les attributs composant les clés primaires des relations sont de type VARCHAR. La variable est obtenue en concaténant les valeurs des différents attributs de la clef primaire. Cette valeur sera stockée dans un nouvel attribut dénoté VAR.

Pour cet exercice, nous utiliserons plusieurs fonctions qui sont admises.
- La fonction ADDOR prend deux chaînes de caractères CH1 et CH2 et renvoie (CH1 U CH2).
- La fonction ADDAND prend deux chaînes de caractères CH1 et CH2 et renvoie (CH1 * CH2).
- La fonction AGGREG-OR prend un ensemble de chaînes de caractères et renvoie une chaîne de caractères représentant la disjonction des formules associées.
- La fonction AGGREG-AND prend un ensemble de chaînes de caractères et renvoie une chaîne de caractères représentant la conjonction des formules associées.

Question 1.18. Créer les commandes SQL qui permettent d'ajouter la colonne VAR à la relation *Croisiere* du schéma de la sous-section 1.2. Créer la requête de mises à jour pour remplir cette colonne dans la base de données.

Exemple 4 *Nous reprenons la requête Q_2 égale à* SELECT R.A FROM R WHERE R.B = 3. *Cette requête est réécrite en la requête* SELECT R.A AGGREG-OR(R.VAR) AS PROVENANCE FROM R WHERE R.B = 3 GROUP BY R.A. *La chaîne de caractères de l'attribut provenance donne une représentation d'une formule booléenne de la provenance de chaque réponse.*

Question 1.19. Réécrire les requêtes des questions 1.2, 1.6 et 1.9 qui renvoient, en plus des réponses, leur provenance avec le schéma modifié pour toutes les relations, i.e. toutes les relations ont un attribut supplémentaire VAR.

Partie II. Ponts d'un graphe

1 Ponts et blocs dans un graphe non orienté

Dans un graphe non orienté, un **pont** est une arête dont la suppression fait croître le nombre de composantes connexes. Lorsque l'on retire tous les ponts d'un graphe, les composantes connexes restantes sont appelées **blocs**. Par exemple, le graphe représenté figure 1 possède 4 ponts, reliant les paires de sommets $(1, 3)$, $(3, 4)$, $(6, 7)$ et $(11, 15)$ et 6 blocs, comme illustré figure 2.

Dans ce problème, nous allons nous intéresser à un algorithme permettant de déterminer les blocs d'un graphe non orienté décrit *en ligne*, c'est-à-dire en ajoutant les arêtes une par une. On suppose que l'on connaît à l'avance le nombre n de sommets du graphe, et que ceux-ci sont numérotés de 0 à $n - 1$.

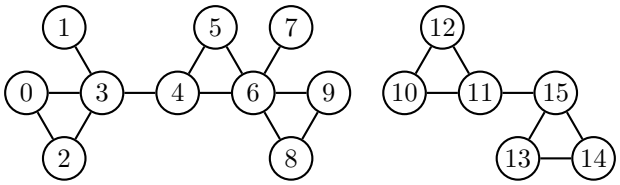


FIGURE 1 – Exemple de graphe.

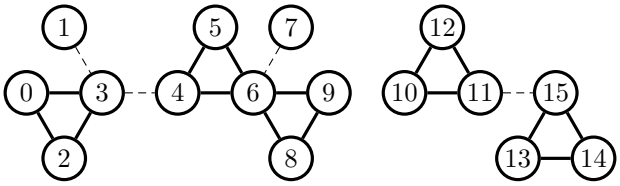


FIGURE 2 – Blocs en gras et ponts en hachuré.

La programmation s'effectuera en OCaml. Le candidat peut, s'il en éprouve le besoin, définir des fonctions auxiliaires pour mieux structurer son code. Il devra alors en préciser le rôle.

Quelques rappels sur le langage OCaml

Une liste est construite à partir de la liste vide `[]` et de la construction x `::` ℓ qui renvoie une nouvelle liste dont la tête est l'élément x et dont la queue est la liste ℓ. L'appel de `List.rev` ℓ renvoie une nouvelle liste, formée des éléments de la liste ℓ en ordre inverse.

On peut créer des tableaux avec les fonctions `Array.make`, `Array.init` et `Array.of_list`.
- L'appel de `Array.make` n x crée un tableau de taille n dont toutes les cases contiennent la valeur x.
- L'appel de `Array.init` n f crée un tableau de taille n dans lequel la valeur de la case d'indice i est égale à $f(i)$.
- L'appel de `Array.of_list` ℓ crée un tableau contenant, dans l'ordre, les éléments d'une liste ℓ.

Tournez la page S.V.P.

Les cases d'un tableau sont numérotées à partir de 0. La fonction `Array.length` renvoie la taille d'un tableau. Pour un tableau t, on accède à l'élément d'indice i avec `t.(i)` et on le modifie avec `t.(i) <-` v.

On mentionne enfin le type polymorphe `'a option` défini par :

```
type 'a option = None | Some of 'a
```

Un élément de la forme `Some` x correspond à la présence d'une valeur x de type `'a`, et un élément de la forme `None` correspond à une absence de valeur.

1.1 Représentation sous forme de forêt

Nous allons représenter le graphe sous forme d'une forêt à l'aide d'une structure de type *union-find* et que nous appellerons *buf* (pour *block union-find*). Cette structure consiste en trois tableaux de taille le nombre n de sommets du graphe :

```
type buf = { parent : int array; repr : bool array; rang : int array; }
```

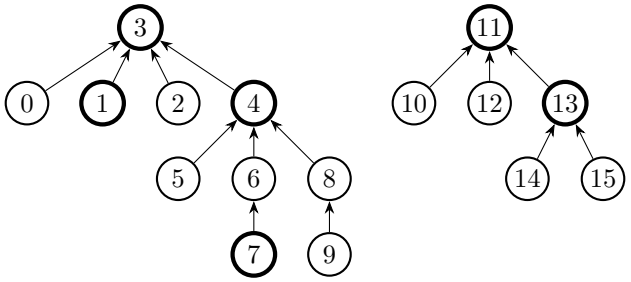
Chaque bloc a un unique *représentant* (défini de façon arbitraire), indiqué par le tableau `repr`. Chaque sommet a un *parent*, représenté par le tableau du même nom et qui a deux fonctions :

— si un sommet s est le représentant d'un bloc, notons-le b, alors soit `parent.(`s`)` $= s$, auquel cas s est racine d'un des arbres de la forêt, soit il indique un sommet d'un bloc b' tel qu'il existe un pont entre un sommet de b et un sommet de b' ;

— sinon, en suivant les parents successifs à partir d'un sommet s, on arrive au représentant de son bloc.

Enfin, le tableau `rang` indique, pour chaque représentant de bloc, une mesure de la taille de ce bloc. Cette valeur interviendra uniquement à la question 2.10 lors de l'implémentation de l'opération de fusion entre blocs.

Ainsi, tous les blocs sont représentés à la manière d'une structure de type *union-find*, à la différence que les différents blocs forment une forêt. On donne en figure 3 une représentation du graphe exemple de la figure 1 comme un élément de type *buf*, ainsi que son illustration graphique.




```
{ parent = [|3; 3; 3; 3; 3; 4; 4; 6; 4; 8; 11; 11; 11; 11; 13; 13|];
  repr = [|false; true; false; true; true; false; false; true; false;
           false; false; true; false; true; false; false|];
  rang = [|0; 0; 0; 1; 2; 0; 0; 0; 0; 0; 0; 1; 0; 1; 0; 0|] }
```

FIGURE 3 – Structure *buf* représentant le graphe exemple.

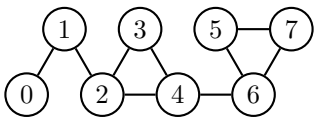


FIGURE 4 – Un autre graphe.

Question 2.1. Indiquer les blocs et les ponts du graphe représenté figure 4.

Question 2.2. Représenter graphiquement une structure *buf* correspondant au graphe de la figure 4.

Question 2.3. Écrire une fonction `init : int -> buf` qui, étant donné un entier n, renvoie une structure *buf* décrivant un graphe de n sommets sans aucune arête : chaque sommet donne lieu à un bloc de rang 0 dont il est le représentant.

Question 2.4. Écrire une fonction `find : buf -> int -> int` qui renvoie le représentant dans la structure *buf* du sommet passé en argument. On mettra en œuvre la *compression de chemin* : on modifie les parents de tous les sommets croisés sur le chemin entre le sommet de départ et son représentant, chaque sommet ayant pour nouveau parent leur représentant commun.

Question 2.5. Écrire une fonction `blocs : buf -> int list list` qui renvoie la liste des blocs correspondant à la structure passée en argument. L'ordre des blocs ainsi que l'ordre des sommets à l'intérieur d'un bloc n'est pas contraint. Ainsi, avec le graphe de la figure 1 (représenté par la forêt de la figure 3), on *peut* avoir :

```
[[15; 13; 14]; [12; 11; 10]; [7]; [9; 8; 6; 5; 4]; [3; 2; 0]; [1]]
```

On utilisera systématiquement la fonction `find` pour déterminer le représentant d'un sommet.

1.2 Ajout d'arêtes

Nous allons maintenant étudier l'effet de l'ajout d'une arête sur notre structure. Plusieurs cas sont possibles, suivant que les extrémités de l'arête ajoutée appartiennent à un même bloc, à des blocs distincts d'une même composante connexe, ou à des blocs de composantes connexes distinctes. Notons que si l'on ajoute une arête entre les sommets d'un même bloc, aucune modification n'est nécessaire. Pour traiter les deux autres cas, nous allons tout d'abord écrire quelques fonctions utilitaires avant d'implémenter la fonction d'ajout d'arête proprement dite.

Dans la suite, on appelle *chaîne de représentants* une suite finie non vide $(s_0, \ldots, s_p)$ de représentants telle que pour tout $i \in \{0, \ldots, p-1\}$, le sommet s_i est le représentant du parent de s_{i+1}. En particulier, les représentants apparaissent de la gauche vers la droite par profondeur croissante. De façon naturelle, une telle chaîne sera représentée par une liste.

Question 2.6. Écrire une fonction `chaine_racine : buf -> int -> int list` qui, étant donné un sommet s du graphe, renvoie la chaîne des représentants reliant le représentant de s à la racine de l'arbre correspondant.

Ainsi, avec l'exemple précédent, la chaîne des représentants pour le sommet 5 est la liste `[3; 4]`. De même, pour le sommet 7 (qui est lui-même un représentant), on doit obtenir `[3; 4; 7]`.

1.2.1 Arêtes entre des sommets de composantes connexes distinctes

Lorsque l'on ajoute une arête entre deux sommets appartenant à des composantes connexes distinctes, cette nouvelle arête est un pont. En terme de structure *buf*, si l'on note u et v les extrémités de l'arête ajoutée, on considère la chaîne des représentants allant de celui de l'un des sommets (supposons que l'on utilise u) vers la racine correspondante et on « retourne » toutes les flèches de la chaîne, faisant du représentant de u la nouvelle racine de son arbre. On change alors le parent du représentant de u pour le faire pointer vers celui de v.

La figure 5 illustre le résultat de l'ajout d'une arête entre les sommets 5 et 14.

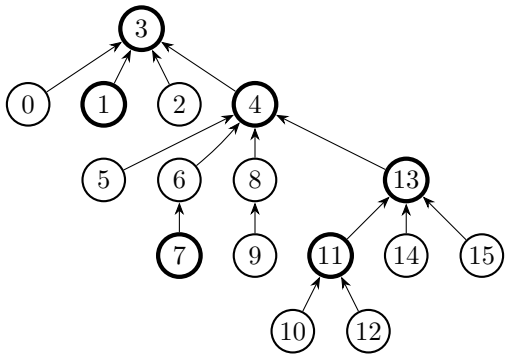


FIGURE 5 – Exemple d'ajout d'arête entre deux composantes connexes distinctes, entre les sommets 5 et 14.

Question 2.7. Représenter la forêt obtenue après avoir ajouté dans la forêt représentée en figure 3 une arête entre les sommets 7 et 12. On supposera que c'est le chemin issu du sommet 7 qui est retourné.

Question 2.8. Écrire une fonction `retourner_chaine : buf -> int list -> unit` qui implémente le retournement de chaîne présenté ci-dessus, sans effectuer le changement de parent de la nouvelle racine. On supposera que la liste passée en argument est une chaîne de représentants, de premier élément la racine d'un arbre.

1.2.2 Arêtes entre blocs distincts d'une même composante connexe

L'ajout d'une arête entre deux blocs d'une même composante connexe va entraîner la fusion de ces deux blocs ainsi que de tous les blocs compris entre les deux. Par exemple, comme illustré sur la figure 6, l'ajout d'une arête entre les sommets 1 et 15 à partir du graphe de la figure 5 conduit à la fusion des blocs de représentants 1, 3, 4 et 13.

Question 2.9. Représenter une forêt que l'on peut obtenir après avoir ajouté une arête entre les sommets 7 et 12 dans la forêt représentée en figure 5.

Question 2.10. Écrire une fonction `union : buf -> int -> int -> unit` qui effectue l'union des blocs dont les représentants sont passés en argument. Concrètement, le parent du représentant du bloc de plus petit rang deviendra le représentant du bloc de plus grand rang. En cas d'égalité des rangs, le choix se fera de façon arbitraire, et le rang du bloc résultant augmentera de 1. Le rang d'un sommet qui n'est pas un représentant ne joue aucun rôle. On ne se souciera pas, pour le moment, de la valeur du parent du bloc obtenu.

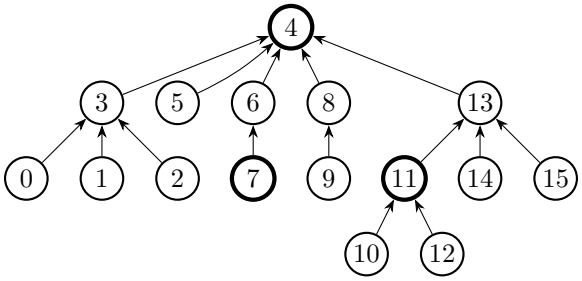


FIGURE 6 – Exemple d'ajout d'arête entre deux sommets d'une même composante connexe.

Question 2.11. Écrire une fonction `fusion_chaine : buf -> int list -> unit` qui effectue la fusion de tous les blocs dont les représentants sont dans la chaîne de représentants passée en argument. On portera une attention particulière, une fois la fusion de la chaîne effectuée, à la valeur du parent du représentant du bloc obtenu.

1.2.3 Fonction d'ajout

Question 2.12. Écrire une fonction `ajout : buf -> int -> int -> unit` qui implémente l'ajout d'un arête entre les deux sommets u et v passés en argument. En notant leurs représentants r_u et r_v, si ceux-ci sont différents, on distingue (à partir des chaînes de ces représentants vers leurs racines respectives) les cas où ceux-ci appartiennent à des composantes connexes distinctes ou à la même composante connexe. Dans le premier cas, on retourne la chaîne d'un des représentants jusqu'à la racine puis on lui attribue comme parent l'autre représentant. Dans le second, on fusionne les blocs des chaînes reliant r_u et r_v à leur plus proche ancêtre commun qui est l'élément commun le plus profond de leurs chaînes respectives vers leur racine.

1.3 Liste des ponts

On désire modifier la structure *buf* afin de pouvoir obtenir, en plus des blocs, la liste des ponts du graphe modélisé.

Question 2.13. Décrire une telle modification, en indiquant précisément les modifications à apporter à la structure *buf* et au code des diverses fonctions pour la manipuler, ainsi que l'implémentation d'une nouvelle fonction

```
ponts : buf -> (int * int) list.
```

Idéalement, les modifications des fonctions précédentes se traduiront par un surcoût de complexité constante et la fonction `ponts` sera de complexité linéaire en le nombre de sommets du graphe.

Partie III. Architecture des ordinateurs

1 Algèbre de Boole et circuits combinatoires

Question 3.1. Dans cette question, x et y sont des valeurs dans $\{0, 1\}$. On considère l'opérateur booléen NOR défini par $\text{NOR}(x, y) = \overline{x + y}$. L'opérateur « ou » considéré ici n'est pas exclusif, soit $1 + 1 = 1$. La porte associée est représentée par la figure 1.

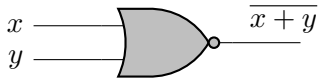


FIGURE 1 – Représentation d'une porte NOR.

a) Donner la table de vérité de l'opérateur NOR.

b) Exprimer les opérateurs de base $\text{NOT}(x) = \overline{x}$, $\text{AND}(x, y) = x.y$ et $\text{OR}(x, y) = x + y$ avec uniquement des opérateurs binaires NOR. En déduire leur réalisation avec des portes NOR à deux entrées.

c) Est-ce que l'opérateur NOR est associatif ? Justifier votre réponse.

d) Calculer $A(x, y, z) = x.y.z$ avec des opérateurs NOR binaires. Pour simplifier la formule, la négation des paramètres $\overline{x}$, $\overline{y}$ et $\overline{z}$ est acceptée.

Question 3.2. Soient les fonctions booléennes $f(x, y, z, t) = x + \overline{x}.\overline{y}.\overline{z}.\overline{t}$, $g(x, y, z, t) = \overline{t}.(z + \overline{x}.y)$ et $h(x, y, z) = x.\overline{y} + \overline{x}.y.\overline{z}$.

a) Démontrer que $x + \overline{x}.y = x + y$. En déduire une (petite) simplification de f.

b) Exprimer les fonctions f, g et h uniquement avec des opérateurs binaires NOR. On peut utiliser la fonction A et la négation des paramètres pour alléger l'expression des résultats.

2 Bascules RS et D

Question 3.3. On considère dans cette question la bascule RS réalisée avec des portes NOR représentée par la figure 2.

a) Quelles sont les valeurs possibles des sorties Q_1 et Q_2 quand les entrées R et S sont stables à 0 ($R = S = 0$) ? On appelle cet état E_1. Justifier votre réponse.

b) Que se passe-t-il quand R passe à 1 à partir de l'état E_1 ? On appelle cet état E_2. Justifier votre réponse.

c) Que se passe-t-il quand à partir de l'état E_2, S passe à 1 ? Justifier votre réponse.

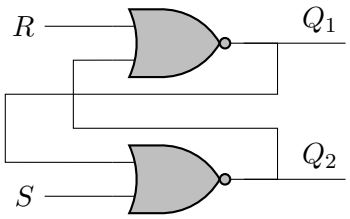


FIGURE 2 – Bascule RS réalisée avec des portes NOR.

d) En déduire la table de vérité de la bascule RS. Ici, les nouvelles valeurs de sortie sont notées Q_1 et Q_2 et sont dépendantes de R, S et des valeurs de sortie précédentes Q'_1 et Q'_2 ;

e) À partir de la table de vérité de la bascule RS, montrer que $Q_1 = \overline{R}(Q'_1+S)$ et $Q_2 = \overline{S}(Q'_2+R)$;

f) On suppose qu'au démarrage $R = S = 0$ et que les entrées varient de sorte qu'à tout instant, $R.S = 0$. Quelle est la relation entre Q_1 et Q_2 ? Justifier votre réponse.

Question 3.4. On souhaite maintenant réaliser une bascule D à front descendant à partir de 3 bascules RS. Pour cela, on considère le schéma présenté par la figure 3. Le signal D est la donnée, H un signal d'horloge, Q et $\overline{Q}$ les signaux de sortie.

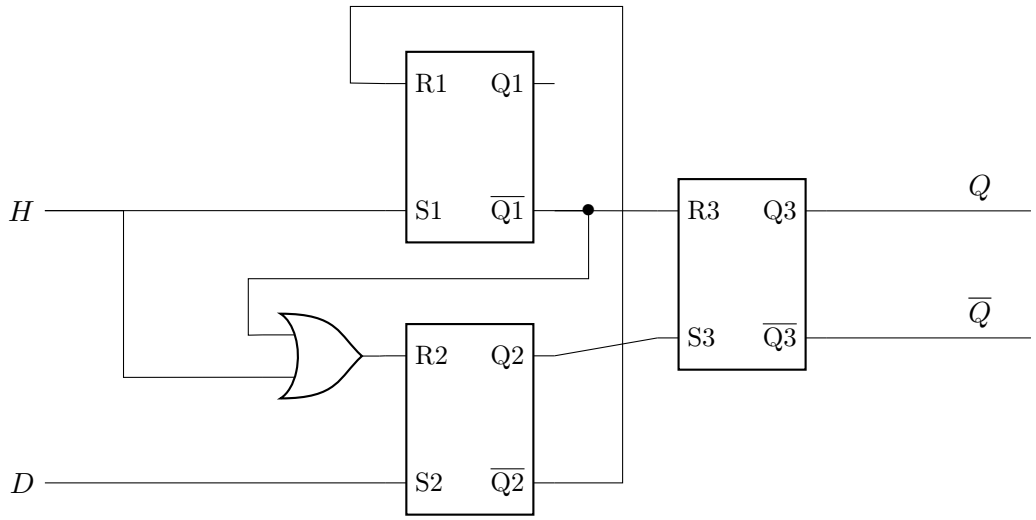


FIGURE 3 – Réalisation d'une bascule D à front descendant avec 3 bascules RS. La porte représentée est une porte OR.

a) Donner les équations vérifiées par les sorties des 3 bascules $\overline{Q1}$, $Q2$, $\overline{Q2}$, $Q3$ et $\overline{Q3}$;

b) Donner les équations vérifiées par les entrées des 3 bascules $R1$, $S1$, $R2$, $S2$, $R3$ et $S3$.

Question 3.5. On suppose dans cette question que la valeur de D est stable quand le signal de l'horloge passe de 1 à 0 (au moment du front descendant). On souhaite démontrer les deux propriétés suivantes :

P1 La valeur de la sortie $\overline{Q}$ est celle de $\overline{D}$ au moment du dernier front descendant.

P2 La valeur de sortie $\overline{Q}$ est stable entre deux fronts descendants.

Soit t, un instant qui coïncide avec un front descendant. On suppose dans cette question que $D = 0$ à t.

a) On considère dans un premier temps l'état du système à $H = 1$ juste avant le front descendant à t. Montrer que les sorties $Q3$ et $\overline{Q3}$ sont stables (ne varient pas) ;

b) On suppose maintenant qu'à t, $D = 0$. Démontrer qu'au moment où H passe de 1 à 0, $Q3 = 0$ et $\overline{Q3} = 1$

c) Démontrer que les sorties $Q3$ et $\overline{Q3}$ restent inchangées tant que H reste à 0 ;

d) Que se passe-t-il quand H repasse à 1 ?

Question 3.6.

a) Reprendre le raisonnement de la question précédente en le modifiant pour traiter le cas $D = 1$;

b) Conclure dans le cas général ($D \in \{0, 1\}$) que les deux propriétés **P1** et **P2** sont vérifiées, en justifiant votre réponse.

3 Synthèse d'un automate de Moore

On considère dans cette partie un automate de Moore $\mathcal{A}$ de 5 états représenté par la figure 4. Les noeuds sont étiquetés par l'état et la sortie associée.

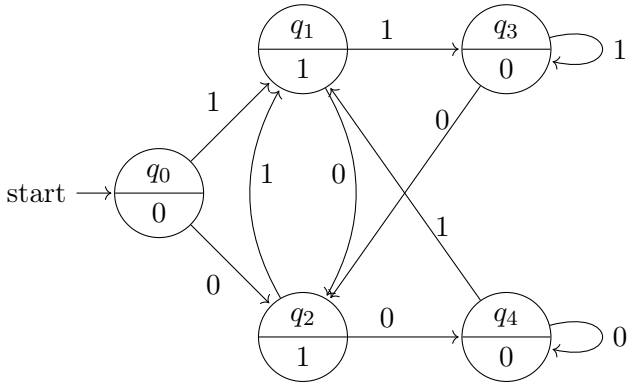


FIGURE 4 – Un automate de Moore $\mathcal{A}$.

Question 3.7.

a) Donner la sortie de l'automate de Moore $\mathcal{A}$ pour l'entrée $e = 1000110111$ et la liste des états visités.

b) Décrire sans justification à quoi correspond la sortie de cet automate.

Question 3.8. On souhaite synthétiser cet automate par une machine de Moore. Les états seront stockés à l'aide de bascule D. On suppose que les états sont codés par des entiers correspondant à leur numérotation, soit l'état q_0 est représenté par l'entier 0, l'état q_1 par l'entier 1, etc.

a) Quel est le nombre β de bascules nécessaires pour mémoriser les états de l'automate ? Justifier votre réponse.

   Par la suite, on note $Q_0, \dots Q_{\beta-1}$ les β valeurs booléennes qui permettent de coder les états de l'automate de sorte que l'entier $i \in \{0, \dots, 5\}$ associé à l'état q_i vérifie $i = \sum_{j=0}^{\beta-1} 2^j Q_j$;

b) Donner le tableau de Karnaugh de la fonction qui associe selon les valeurs Q_j, $j \in \{0, \dots, \beta-1\}$ la valeur de sortie S de l'automate. En déduire l'expression booléenne de S ;

c) Donner l'expression de S uniquement avec des opérateurs binaires NOR et les valeurs Q_j et $\overline{Q}_j$ pour $j \in \{0, \dots, \beta - 1\}$.

Question 3.9.

a) Donner les tables de Karnaugh qui permettent de calculer l'état futur de la machine de Moore en fonction de l'état présent exprimé par les valeurs Q_j, $j \in \{0, \dots, \beta - 1\}$ et la valeur E de l'entrée. On note D_j, $j \in \{0, \dots, \beta-1\}$ les valeurs de signaux booléens correspondant à l'état futur ;

b) En déduire les expressions booléennes des valeurs D_j, $j \in \{0, \dots, \beta - 1\}$ en fonction des valeurs Q_j, $j \in \{0, \dots, \beta - 1\}$ et de la valeur E de l'entrée ;

c) Donner les expressions de D_j, $j \in \{0, \dots, \beta - 1\}$ uniquement avec des opérateurs binaires NOR et les valeurs E, $\overline{E}$ et Q_j et $\overline{Q}_j$ pour $j \in \{0, \dots, \beta - 1\}$.

* *
*