Mathis Degryse <mathis.degryse@proton.me>
Balthazar Patiachvili <balthazar.patiachvili@crans.org>

Composition 2023

Partie I. Provenance en bases de données

Question 1.1

```sql
CREATE TABLE Croisiere(
  cid VARCHAR(255) PRIMARY KEY,
  depart VARCHAR(255),
  arrivee VARCHAR(255),
  distance INT
);

CREATE TABLE Bateau(
  bid VARCHAR(255) PRIMARY KEY,
  type VARCHAR(255),
  autonomie INT
);

CREATE TABLE Employe(
  eid VARCHAR(255) PRIMARY KEY,
  nom VARCHAR(255),
  age INT
);

CREATE TABLE Navigable(
  bid VARCHAR(255) FOREIGN KEY REFERENCES Bateau(bid),
  eid VARCHAR(255) FOREIGN KEY REFERENCES Employe(eid),
  PRIMARY KEY (bid, eid)
);
```

Question 1.2

Voici la requête SQL :

```sql
SELECT eid
FROM Navigable
NATURAL JOIN Bateau
WHERE type = "Catamaran";
```

et la requête en algèbre relationnelle correspondante :

$$\pi_{\text{eid}}\left(\sigma_{\text{type} = \text{Catamaran}}(\texttt{Navigable} \bowtie \texttt{Bateau})\right)$$

Question 1.3

Voici la requête SQL :

```sql
SELECT
  bid,
```

```
3    cid
4  FROM Bateau
5  CROSS JOIN Croisiere
6  WHERE distance <= autonomie;
```

et la requête en algèbre relationnelle correspondante :

$$\pi_{\text{bid, cid}}(\sigma_{\text{distance} \leq \text{autonomie}}(\texttt{Bateau} \times \texttt{Croisiere}))$$

Question 1.4

Voici la requête SQL :

```sql
1   SELECT bid
2   FROM Navigable
3   NATURAL JOIN Employe
4   WHERE age > 40
5   GROUP BY bid
6   HAVING COUNT(*) = (
7     SELECT COUNT(*)
8     FROM Employe
9     WHERE age > 40
10  );
```

et la requête en algèbre relationnelle correspondante :

$$\pi_{\text{bid}}(\texttt{Navigable} \div \pi_{\text{eid}}(\sigma_{\text{age} > 40}(\texttt{Employe})))$$

Question 1.5

Voici la requête SQL :

```sql
1  SELECT eid
2  FROM Navigable
3  NATURAL JOIN Bateau
4  WHERE autonomie > 3000
5  EXCEPT
6  SELECT eid
7  FROM Navigable
8  NATURAL JOIN Bateau
9  WHERE type = "Catamaran";
```

et la requête en algèbre relationnelle correspondante :

$$\pi_{\text{eid}}(\sigma_{\text{autonomie} > 3000}(\texttt{Navigable} \bowtie \texttt{Bateau})) \setminus \pi_{\text{eid}}(\sigma_{\text{type} = \text{Catamaran}}(\texttt{Navigable} \bowtie \texttt{Bateau}))$$

Question 1.6

Voici la requête SQL :

```sql
1  SELECT eid
2  FROM Employe
3  WHERE age = (
```

```sql
    SELECT MAX(age)
    FROM Employe
);
```

Remarque 1

Cette requête SQL est relativement simple, mais ne se traduit pas directement en algèbre relationnelle. Voici une requête SQL équivalente qui peut se traduire directement :

```sql
SELECT eid
FROM Employe
EXCEPT
SELECT e1.eid
FROM Employe e1
JOIN Employe e2
ON e1.age < e2.age;
```

et la requête en algèbre relationnelle correspondante :

$$\pi_{\text{eid}}(\texttt{Employe}) \setminus \pi_{\text{e1id}}\left(\sigma_{\text{e1\_age} < \text{e2\_age}}\left(\rho_{\text{eid} \to \text{e1id},\ \text{age} \to \text{e1\_age}}(\texttt{Employe}) \times \rho_{\text{age} \to \text{e2\_age}}(\texttt{Employe})\right)\right)$$

Question 1.7

```sql
SELECT COUNT(*)
FROM Bateau;
```

Question 1.8

```sql
SELECT bid, COUNT(*)
FROM Navigable
GROUP BY bid;
```

Question 1.9

```sql
SELECT AVG(age) as moyenne
FROM (
    SELECT DISTINCT eid, age
    FROM Employe
    NATURAL JOIN Navigable
);
```

Question 1.10

À partir d'une requête Q, il est possible d'obtenir une requête Q' telle que Q' renvoie un enregistrement contenant uniquement un 0-uplet sur la base de données D si et seulement Q renvoie au moins un enregistrement lorsqu'elle est évaluée sur D de la manière suivante :

- dans le cadre d'une requête SQL, on crée la requête Q' suivate :

```sql
SELECT DISTINCT FROM Q
```

En effet, l'opérateur `SELECT` assure que chaque ligne renvoyée par la requête Q est transformée en 0-uplet, et l'opérateur `DISTINCT` fait en sorte que s'il y a au moins un résultat par la requête Q, alors il n'y en aura bien qu'un seul en sortie.

- dans le cadre d'une requête en algèbre relationnelle, on crée la requête Q' suivante : $Q' = \pi(Q)$. En effet, l'opérateur π projeté sur aucune colonne permet d'obtenir en sortie des 0-uplets. Or, les objets de l'alèbre relationnelle sont les ensembles, il n'y a donc aucun doublon, et il n'y a qu'un seul 0-uplet possible : $()$. Donc Q' renvoie bien $()$ si et seulement si Q renvoie au moins un résultat.

Pour les requêtes de la question 1.2, les requêtes booléennes induites sont les suivantes :

```sql
SELECT DISTINCT
FROM (
  SELECT eid
  FROM Navigable
  NATURAL JOIN Bateau
  WHERE type = "Catamaran"
);
```

et $\pi\big(\pi_{\text{eid}}\big(\sigma_{\text{type} = \text{Catamaran}}(\texttt{Navigable} \bowtie \texttt{Bateau})\big)\big)$.

Question 1.11

Sur la requête de la question 1.2, le résultat sur cette base de données est :

eid
e2

La provenance de cette requête est l'ensemble des bases de données composées de `Navigable`$(\text{b1}, \text{e2})$, de `Bateau`$(\text{b1}, \text{Catamaran}, 2000)$, de `Employe`$(\text{e2}, \text{Alice}, 42)$, et de n'importe quel ensemble d'autres enregistrements de la base initiale, tel que les clefs étrangères des enregistrement `Navigable` apparaîssent par ailleurs dans la provenance.

Sur la requête de la question 1.9, le résultat sur cette base de données est :

moyenne
31

La provenance de cette requête est l'ensemble des bases de données composées de `Employe`$(\text{e1}, \text{Bob}, 20)$, `Employe`$(\text{e2}, \text{Alice}, 42)$, `Navigable`$(\text{b2}, \text{e2})$, de `Navigable`$(\text{b1}, \text{e1})$ ou de `Navigable`$(\text{b3}, \text{e1})$, et de n'importe quel ensemble d'autres enregistrements de la base initiale, tel que les clefs étrangères des enregistrements `Navigable` apparaîssent dans la provenance.

Question 1.12

1. L'ensemble $\{\{e_1, e_2\}\}$ peut être représenté par la formule $x_1 \wedge x_2$.
2. L'ensemble $\{\{e_1\}, \{e_2\}\}$ peut être représenté par la formule $(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$.

Question 1.13

Montrons par récurrence sur $n \in \mathbb{N}^*$ que $|\phi| = 2^n$.

- Pour $n = 1$, on a $E = \{e_1\} \cup \{f_1\}$ et $\phi = \{\{e_1\}, \{f_1\}\}$, donc $2^1 = |\phi|$.
- Soit $n \in \mathbb{N}$ tel que $|\phi| = 2^n$. On a alors $E = \{e_1, ..., e_n\} \cup \{f_1, ..., f_n\}$, posons $E' = E \cup \{e_{n+1}, f_{n+1}\}$, et le ϕ' correspondant. Alors, soit $P \in \phi$: on a $P \cup \{e_{n+1}\} \in \phi'$, et $P \cup \{f_{n+1}\} \in \phi'$. Or, ces deux éléments sont distincts, et pour $P, P' \in \phi$, on a également immédiatement $P \neq P' \implies P \cup \{e_{n+1}\} \neq P' \cup \{e_{n+1}\}$, et de même pour f_{n+1}, car e_{n+1} et f_{n+1} n'apparaissent pas dans E et donc dans ϕ. Il n'y a pas d'autres éléments dans ϕ' que ceux décrits de cette manière, on obtient donc : $|\phi'| = |\phi| \times 2$, soit $|\phi'| = 2^{n+1}$. La propriété est donc vérifiée au rang $n + 1$.

. Donc, $\forall n \in \mathbb{N}^*, |\phi| = 2^n$, d'où ϕ est un ensemble de taille exponentielle.

On pose la formule $\psi = (e_1 \vee f_1) \wedge (\neg e_1 \vee \neg f_1) \wedge ... \wedge (e_n \vee f_n) \wedge (\neg e_n \vee \neg f_n)$ (en confondant éléments de E et variables booléennes pour ne pas surcharger de notations). Celle-ci est bien de taille linéaire : il y a précisément $2n$ symboles $\vee$, $2n - 1$ symboles $\wedge$, $4n$ variables et $4n$ parenthèses, donc la taille de cette formule est $12n - 1$, ce qui est linéaire en n. Montrons maintenant la correction de cette formule : on veut montrer que pour tout $P \in \phi$, la valuation induite par P valide ψ, et réciproquement pour toute valuation validant ψ, l'ensemble induit appartient à ϕ. On étend ici le domaine des valuations à toutes les formules booléennes pour ne pas surcharger de notations.

- Soit $P \in \phi$ et ν_P la valuation induite (i.e $x \in P \iff \nu_P(x) = \top$). Par définition de ϕ, pour chaque $i \in [\![1, n]\!]$, on a $e_i \in P$ ou $f_i \in P$ mais pas les deux. On suppose sans perdre de généralité que $e_i \in P$, et donc $f_i \notin P$: on a alors $\nu_P(e_i) = \top$ et $\nu_P(f_i) = \bot$, et donc $\nu_P((e_i \vee f_i) \wedge (\neg e_i \vee \neg f_i)) = \top$. Or, cela est vrai pour tout i, donc $\nu_P(\psi) = \top$.
- Soit ν une valuation de X_E telle que $\nu(\psi) = \top$, et P le sous-ensemble de E induit (i.e $x \in P \iff \nu_P(x) = \top$). Considérons $i \in [\![1, n]\!]$ et la partie de ψ concernant les variables d'indice i : on a $\nu((e_i \vee f_i) \wedge (\neg e_i \vee \neg f_i)) = \top$. On en déduit immédiatement que $\nu(e_i) = \top$ et $\nu(f_i) = \bot$ ou inversement, et donc $e_i \in P$ et $f_i \notin P$, ou inversement. Ce résultat est vrai quel que soit i, donc dans tous les cas on a e_i ou f_i qui appartient à P, mais pas les deux. Donc $P \in \phi$ par définition de ϕ.

La formule ψ est donc linéaire et est bien une représentation en formule booléenne de ϕ.

Question 1.14

Pour les enregistrements `Croisiere`, `Bateau`, et `Employe` on confond ici clef primaire et variable booléenne associée. Pour les enregistrements `Navigable`, on les note sous la forme n_{b_i,e_j} où b_i est la clef étrangère associée à la table `Bateau`, et e_j à la table `Employe`. En réutilisant les résultats de la question 1.11, on voit qu'il faut que les variables n_{b_1,e_2}, b_1 et e_2 soient toutes vraies, et également que les variables n_{b_i,e_j} vérifient $n_{b_i,e_j} \Rightarrow b_i \wedge e_j$ (pour que les clefs étrangères fassent référence à des clefs primaires existantes). La formule suivante décrit ainsi la provenance booléenne de la requête de la question 1.2 appliquée à la base de données présentée en 2.3 :

$$n_{b_1,e_2} \wedge b_1 \wedge e_2 \wedge \left(\neg n_{b_2,e_2} \vee (b_2 \wedge e_2)\right) \wedge \left(\neg n_{b_3,e_1} \vee (b_3 \wedge e_1)\right)$$

Question 1.15

> **Remarque 2**
>
> La provenance au travers de l'algèbre relationnelle étant mal définie, je ne suis pas certain de ces réponses.

La requête 1.3 sous forme d'algèbre relationnelle est $Q = \pi_{\text{bid, cid}}(\sigma_{\text{distance} \leq \text{autonomie}}(\texttt{Bateau} \times \texttt{Croisiere}))$, donc :

$$\begin{aligned}
\Pr((\text{bid}, \text{cid}), Q, D) &= \bigvee_{\substack{\text{ty, au, de, ar, di}}} \Pr((\text{bid}, \text{ty}, \text{au}, \text{cid}, \text{de}, \text{ar}, \text{di}), \sigma_{\text{di} \leq \text{au}}(\texttt{Bateau} \times \texttt{Croisiere}), D) \\
&= \bigvee_{\substack{\text{ty, au, de, ar, di} \\ \text{di} \leq \text{au}}} \Pr((\text{bid}, \text{ty}, \text{au}, \text{cid}, \text{de}, \text{ar}, \text{di}), \texttt{Bateau} \times \texttt{Croisiere}, D) \\
&= \bigvee_{\substack{\text{ty, au, de, ar, di} \\ \text{di} \leq \text{au}}} \Pr((\text{bid}, \text{ty}, \text{au}), \texttt{Bateau}, D) \wedge \Pr((\text{cid}, \text{de}, \text{ar}, \text{di}), \texttt{Croisiere}, D) \\
&= \bigvee_{\substack{\text{ty, au, de, ar, di} \\ \text{di} \leq \text{au}}} \texttt{Bateau}(\text{bid}, \text{ty}, \text{au}) \wedge \texttt{Croisiere}(\text{cid}, \text{de}, \text{ar}, \text{di})
\end{aligned}$$

La requête 1.6 sous forme d'algèbre relationnelle est $Q = \pi_{\text{eid}}(\texttt{Employe}) \setminus \pi_{\text{e1id}}\big(\sigma_{\text{e1\_age} < \text{e2\_age}}(\rho_1(\texttt{Employe}) \times \rho_2(\texttt{Employe}))\big)$, donc :

$$\begin{aligned}
\Pr((\text{eid}), Q, D) &= \Pr((\text{eid}), \pi_{\text{eid}}(\texttt{Employe}), D) \wedge \\
&\quad \neg \Pr((\text{ei1d}), \pi_{\text{e1}}(\sigma_{\text{a1} < \text{a2}}(\texttt{Employe} \times \texttt{Employe})), D) \\
&= \bigvee_{\substack{\text{nom, age}}} \Pr((\text{eid}, \text{nom}, \text{age}), \texttt{Employe}, D) \wedge \\
&\quad \neg \left(\bigvee_{\substack{\text{e1, n1, a1, e2, n2, a2} \\ \text{a1} < \text{a2}}} \Pr((\text{e1}, \text{n1}, \text{a1}, \text{e2}, \text{n2}, \text{a2}), \texttt{Employe} \times \texttt{Employe}) \right) \\
&= \bigvee_{\substack{\text{nom, age}}} \Pr((\text{eid}, \text{nom}, \text{age}), \texttt{Employe}, D) \wedge \\
&\quad \neg \left(\bigvee_{\substack{\text{e1, n1, a1, e2, n2, a2} \\ \text{a1} < \text{a2}}} \Pr((\text{e1id}, \text{n1}, \text{a1}), \texttt{Employe}, D) \wedge \Pr((\text{e2id}, \text{n2}, \text{a2}), \texttt{Employe}, D) \right) \\
&= \left(\bigvee_{\substack{\text{nom, age}}} \Pr((\text{eid}, \text{nom}, \text{age}), \texttt{Employe}, D) \right) \wedge \neg \left(\bigvee_{\substack{\text{e1, n1, a1, e2, n2, a2} \\ \text{a1} < \text{a2}}} \texttt{Employe}(\text{e1}, \text{n1}, \text{a1}) \wedge \texttt{Employe}(\text{e2}, \text{n2}, \text{a2}) \right)
\end{aligned}$$

Question 1.16

- La requête 1.2 est croissante : si $D \leq D'$, alors D' contient au moins les mêmes marins pouvant piloter des catamarans, donc ajouter des marins, des catamarans et/ou des enregistrements `Navigable` ne peut qu'augmenter la sortie de la requête, donc $Q(D) \leq Q(D')$.
- La requête 1.3 est croissante pour les mêmes raisons.
- La requête 1.4 n'est pas croissante : si l'on prend un bateau avec `bid` pour identifiant renvoyé par une requête sur une base de données D, ajouter un nouvel employé d'identifiant `eid`, d'âge 50

ans et un enregistrement `Navigable(bid, eid)` donnera une nouvelle base de données D' telle que $D \leq D'$ et $(\texttt{bid}) \notin Q(D')$.

- La requête 1.5 n'est pas croissante : si l'on prend une base de données D et $(\texttt{eid}) \in Q(D)$, alors en ajoutant un catamaran d'identifiant `bid` et un enregistrement `Navigable(bid, eid)` à D, on obtient une nouvelle base de données D' telle que $D \leq D'$ et $(\texttt{eid}) \notin Q(D')$.
- La requête 1.6 n'est pas croissante : si l'on prend une base de données D et $(\texttt{eid}) \in Q(D)$, alors on considère l'enregistrement `Employe(eid, nom age)` associé, et on ajoute un enregistrement `Employe(eid', nom, age')` avec `eid'` un nouvel identifiant unique, et $\texttt{age'} = \texttt{age} + 1$. On a alors $(\texttt{eid}) \notin Q(D')$ avec D' la nouvelle base de données, le seul employé ayant l'âge maximal parmi les employés est celui dont l'identifiant est `eid'`.
- La requête 1.7 n'est pas croissante : en ajoutant un bateau à la base de données, on change la sortie qui ne comporte qu'un seul élément.
- La requête 1.8 n'est pas croissante : en ajouant un marin qui peut naviguer sur un bateau déjà existant, on change l'un des tuples de la sortie.
- La requête 1.9 n'est pas croissante : en ajoutant un employé dont l'âge est différent de la moyenne calculée sur une base de données D, on obtient une base de données D' où le seul tuple de la sortie est différent que celui obtenu sur D.

Question 1.17

Soient D une base de données et Q une requête croissante. Montrons que $\Pr(Q, D)$, vue comme une formule booléenne φ, est croissante. Soient ν_1 et ν_2 deux valuations telles que $\nu_1 \leq \nu_2$ et $\nu_1(\varphi) = \top$: montrons que $\nu_2(\varphi) = \top$. On associe ν_1 au sous-ensemble E_1 de D, et de même pour $E_2 \subseteq D$. On a donc $E_1 \subseteq E_2$ car $\nu_1 \leq \nu_2$. Or, Q est croissante, donc $Q(E_1) \leq Q(E_2)$, d'où $Q(E_2)$ non vide (car $\nu_1(\varphi) = \top$, et donc $Q(E_1)$ non vide). On a donc $\nu_2(\varphi) = \top$, d'où $\Pr(Q, D)$ croissante.

Question 1.18

Pour ajouter une colonne `VAR` à la relation `Croisiere`, on altère tout d'abord la table de cette manière :

```sql
ALTER TABLE Croisiere
ADD VAR VARCHAR(255);
```

On peut ensuite affecter à toutes les lignes la valeur de la colonne `VAR` :

```sql
UPDATE Croisiere
SET VAR = cid;
```

Question 1.19

Remarque 3

Je ne suis vraiment pas des nouvelles requêtes, n'hésitez pas à me prévenir si vous pensez qu'il y a une erreur.

La requête de la question 1.2 se réécrit de la manière suivante :

```sql
SELECT eid, ADDAND(AGGREG_OR(n.VAR), AGGREG_OR(b.VAR)) AS provenance
FROM Navigable n
```

```
3 NATURAL JOIN Bateau b
4 WHERE type = "Catamaran"
5 GROUP BY eid;
```

La requête de la question 1.6 se réécrit de la manière suivante :

```sql
SELECT eid, ADDAND(AGGREG_OR(VAR), CONCAT("not", ADDAND(AGGREG_OR(prov1),
AGGREG_OR(prov2)))) as provenance
FROM Employe
WHERE age NOT IN (
  SELECT e1.age, e1.VAR AS prov_1, e2.VAR AS prov_2
  FROM Employe e1
  JOIN Employe e2
  ON e1.age < e2.age
);
```

La requête de la question 1.9 se réécrit de la manière suivante :

```sql
SELECT AVG(age) as moyenne, AGGREG_AND(prov) as provenance
FROM (
  SELECT DISTINCT e.eid, age, AGREG_OR(e.VAR) as prov
  FROM Employe e
  JOIN Navigable n
  ON e.eid = n.eid
  GROUP BY e.eid
);
```

Partie II. Ponts d'un graphe

1. Ponts et blocs dans un graphe non orienté

Question 2.1

Les ponts du graphe de la figure 4 sont les arêtes reliant les paires de sommets $(0,1)$, $(1,2)$ et $(4,6)$. On en déduit que le graphe est composé des 4 blocs $\{0\}$, $\{1\}$, $\{2,3,4\}$ et $\{5,6,7\}$.

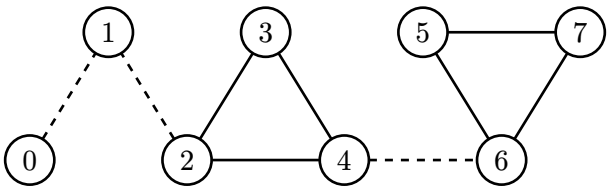

Fig. 1. – Blocs et Ponts du graphe de la figure 4

Question 2.2

On choisit comme représentants des blocs les sommets $0, 1, 4$ et 7, représentés par des cercles épais en Fig. 2

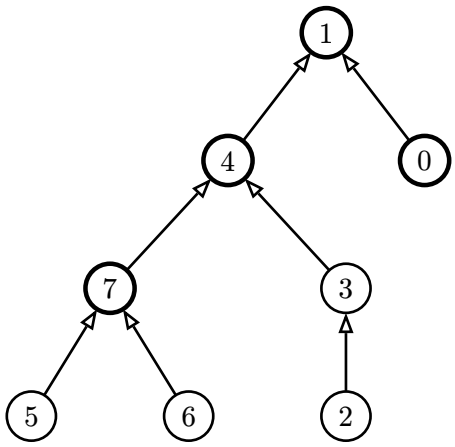

Fig. 2. – Structure *buf* représentant le graphe en figure 4

Question 2.3

```ocaml
let init (n: int): buf = {
  parent = Array.init n (fun x -> x);
  repr = Array.make n true;
  rang = Array.make n 0;
}
```

Question 2.4

```ocaml
let rec find (b: buf) (i: int): int =
  if b.repr.(i) then i else begin
    let r = find b b.parent.(i) in
    b.parent.(i) <- r;
    r
  end
```

Question 2.5

On construit un tableau de listes qui contient, pour chaque indice, s'il correspond à un représentant, à la liste des éléments du bloc correspondant.

On convertit ensuite ce tableau en la forme attendue. La complexité totale correspondante est linéaire (car `find` est en coût amorti constant).

```ocaml
 let blocs (b : buf) : int list list =
   let n = Array.length b.parent in
   let b_arr = Array.make n [] in

   for i = 0 to n - 1 do
     let r = find b i in
     b_arr.(r) <- i :: b_arr.(r)
   done;

   Array.fold_left
     (fun acc l -> if l = [] then acc else l :: acc)
```

```
12      [] b_arr
```

Question 2.6

Si on utilise directement une fonction récursive, on obtiendra une liste dans le mauvais sens, on utilise donc une fonction auxiliaire remplissant un accumulateur le long du parcours des ancêtres.

```ocaml
let chaine_racine (b: buf) (s: int) : int list =
  let rec chaine_acc r acc =
    if parent.(r) = r then
      r :: acc
    else
      chaine_acc (find b parent.(r)) r :: acc
  in
  chaine_aux (find b s) []
```

Question 2.7

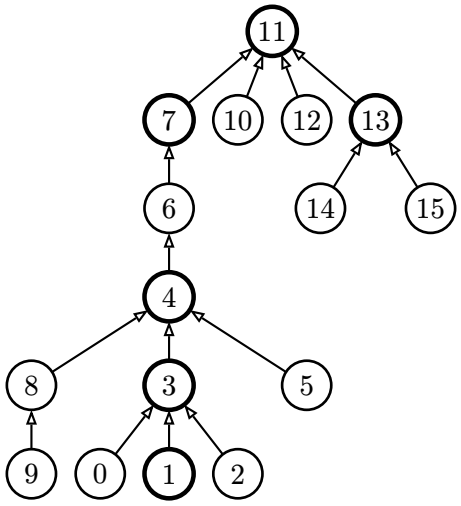

Fig. 3. – Structure *buf* représentant le *buf* en figure 3 après l'ajout de l'arête $(7, 12)$

Question 2.8

Il faut renverser l'ordre *uniquement entre représentants*, sinon, on risque de changer les éléments de bloc. Par ailleurs, il ne faut pas oublier de marquer la nouvelle racine comme telle[1].

```ocaml
let rec retourner_chaine b = function
  | [] -> ()
  | [t] -> b.parent.(t) <- t
  | t :: s :: q ->
    b.parent.(t) <- s;
    retourner_chaine b (s :: q)
```

[1]En pratique on utilise cette fonction seulement pour pouvoir joindre la structure à une autre, si bien que la nouvelle racine ne reste jamais racine par la suite.

Question 2.9

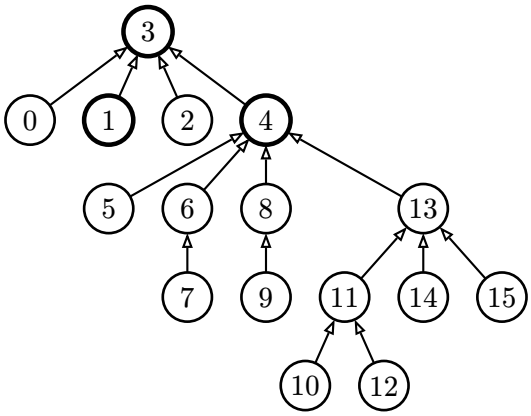

Fig. 4. – Structure *buf* représentant le *buf* en figure 5 après l'ajout de l'arête $(7, 12)$

Question 2.10

```ocaml
let union (b: buf) (r1: int) (r2: int) =
  let rg1 = b.rang.(r1) and rg2 = b.rang.(r2) in
  let (grand, petit, rang') = if rg1 < rg2
    then (r2, r1, rg2)
    else if rg1 = rg2 then (r1, r2, rg1 + 1)
    else (r1, r2, rg1)
  in
  b.parent.(petit) <- grand;
  b.repr.(petit) <- false;
  b.rang.(grand) <- rang'
```

Question 2.11

Il s'agit de réaliser l'union successive.

Partie III. Architecture des ordinateurs

Question 3.1

a. La table de vérité de l'opérateur NOR se trouve en **Tableau 1**

x	y	$\overline{x+y}$
0	0	1
0	1	0
1	0	0
1	1	0

Tableau 1. – Table de vérité de l'opérateur NOR

b. On a premièrement $\mathrm{NOT}(x) = \mathrm{NOR}(x, x)$, ce qu'on vérifie facilement en regardant les cas $x = y$ dans le **Tableau 1**. On en déduit alors que

$$\begin{aligned}\mathrm{OR}(x, y) &= \mathrm{NOT}(\mathrm{NOR}(x, y)) \\ &= \mathrm{NOR}(\mathrm{NOR}(x, y), \mathrm{NOR}(x, y))\end{aligned}$$

D'après les lois de De Morgan, on a

$$\text{NOT}(\text{AND}(x, y)) = \text{OR}(\text{NOT}(x), \text{NOT}(y))$$

On en déduit alors que

$$\begin{aligned}\text{AND}(x, y) &= \text{NOT}(\text{NOT}(\text{AND}(x, y))) \\ &= \text{NOT}(\text{OR}(\text{NOT}(x), \text{NOT}(y))) \\ &= \text{NOR}(\text{NOR}(x, x), \text{NOR}(y, y)) \\ &= \text{NOR}(\overline{x}, \overline{y})\end{aligned}$$

c. Si on construit la table de vérité pour $\text{NOR}(x, \text{NOR}(y, z))$ et $\text{NOR}(\text{NOR}(x, y), z)$, on peut remarquer que ces deux opérations ne sont pas équivalentes (**Tableau 2**). En effet, on a $\text{NOR}(0, \text{NOR}(0, 1)) \neq \text{NOR}(\text{NOR}(0, 0), 1)$

x	y	z	$\overline{x + y}$	$\overline{\overline{x + y} + z}$	$\overline{y + z}$	$\overline{x + \overline{y + z}}$
0	0	0	1	0	1	0
0	0	1	1	0	0	1
0	1	0	0	1	0	1
0	1	1	0	0	0	1
1	0	0	0	1	1	0
1	0	1	0	0	0	0
1	1	0	0	1	0	0
1	1	1	0	0	0	0

Tableau 2. – Table de vérité pour l'associativité de NOR

On a

$$\begin{aligned}A(x, y, z) &= x.y.z \\ &= (x.y).z \\ &= \text{NOR}(\overline{x}, \overline{y}).z \\ &= \text{NOR}\left(\overline{\text{NOR}(\overline{x}, \overline{y})}, \overline{z}\right) \\ &= \text{NOR}(\text{OR}(\overline{x}, \overline{y}), \overline{z}) \\ &= \text{NOR}(\text{NOR}(\text{NOR}(\overline{x}, \overline{y}), \text{NOR}(\overline{x}, \overline{y})), \overline{z})\end{aligned}$$

Question 3.2

a. On peut voir un parallèle entre l'égalité $x + y = x + \overline{x}.y$ et la notion de OU paresseux : on peut se permettre de passer à l'évaluation de y uniquement lorsque x vaut 0. Cette égalité peut se vérifier, comme on l'a fait précédemment, avec une table de vérité. On peut également en faire une preuve directe en utilisant les règles de calcul sur les algèbres de Boole:

$$x + y = x + (x + \overline{x}).y = x + x.y + \overline{x}.y = x.(1 + y) + \overline{x}.y = x + \overline{x}.y$$

On en déduit que

$$\begin{aligned}f(x, y, z, t) &= x + \overline{x}.\overline{y}.\overline{z}.\overline{t} \\ &= x + \overline{y}.\overline{z}.\overline{t}\end{aligned}$$

b. On a

$$\begin{aligned} f(x,y,z,t) &= x + \overline{y}.\overline{z}.\overline{t} \\ &= x + A(\overline{y},\overline{z},\overline{t}) \\ &= \text{NOR}(\text{NOR}(x, A(\overline{y},\overline{z},\overline{t})), \text{NOR}(x, A(\overline{y},\overline{z},\overline{t}))) \end{aligned}$$

On a ensuite

$$\begin{aligned} g(x,y,z,t) &= \overline{t}.(z + \overline{x}.y) \\ &= \overline{t}.z + \overline{t}.\overline{x}.y \\ &= A(\overline{t}, z, 1) + A(\overline{t}, \overline{x}, y) \\ &= \text{NOR}(\text{NOR}(A(\overline{t}, z, 1), A(\overline{t}, \overline{x}, y)), \text{NOR}(A(\overline{t}, z, 1), A(\overline{t}, \overline{x}, y))) \end{aligned}$$

Et enfin

$$\begin{aligned} h(x,y,z,t) &= x.\overline{y} + \overline{x}.y.\overline{z} \\ &= A(x, \overline{y}, 1) + A(\overline{x}, y, \overline{z}) \\ &= \text{NOR}(\text{NOR}(A(x, \overline{y}, 1), A(\overline{x}, y, \overline{z})), \text{NOR}(A(x, \overline{y}, 1), A(\overline{x}, y, \overline{z}))) \end{aligned}$$

Question 3.3

a. Lorsqu'on a $R = S = 0$, on a $Q_1 = \text{NOR}(Q_2, R) = \text{NOR}(Q_2, 0) = \overline{Q_2}$. L'autre porte nous donne la relation $Q_2 = \overline{Q_1}$, qui est équivalente. Donc, lorsque les entrées sont nulles et stables, les sorties sont opposées l'une à l'autre.

b. Si R passe à 1 depuis l'état E_1, on a $R = 1$ et $S = 0$. On a donc $Q_1 = \text{NOR}(Q_2, R) = 0$, et $Q_2 = \text{NOR}(Q_1, S) = \overline{Q_1} = 1$

c. En vérité, la bascule repasse toujours par un état stable $R = S = 0$, on suppose donc qu'on passe de $R = 1, S = 0$ à $R = S = 0$, puis à $R = 0, S = 1$. Ce cas est symétrique à celui de la **Sous-question b**, on a $Q_1 = 1, Q_2 = 0$

Si jamais on passe de $R = 1, S = 0$ à $R = 1, S = 1$, alors on a un état non défini pour la bascule, car la relation $Q_1 = \overline{Q_2}$ n'est plus vérifiée. On obtient en effet $Q_1 = \text{NOR}(Q_2, R) = 0$, et $Q_2 = \text{NOR}(Q_1, S) = 0$

d. La table de vérité est donnée dans le **Tableau 3**

e. L'égalité est donnée dans le **Tableau 3**

R	S	Q_1	Q_2	$\overline{R}(Q_1' + S)$	$\overline{S}(Q_2' + R)$
0	0	Q_1'	Q_2'	Q_1'	Q_2'
0	1	1	0	1	0
1	0	0	1	0	1
1	1	0	0	0	0

Tableau 3. – Table de vérité pour la bascule RS

f. Lors de l'état $R = S = 0$, on a $Q_1 = \overline{Q_2}$ d'après la **Sous-question a**. Cette relation est aussi vraie dans les états $R = 1, S = 0$ et $R = 0, S = 1$. Il reste le cas où on passe d'un de ces deux états à l'état $R = S = 0$, auquel cas les valeurs pour Q_1, Q_2 ne changent pas (1^e ligne du tableau **Tableau 3**). On en déduit que cette relation reste toujours vraie, si on ne passe jamais par le 4^e état.

Question 3.4

a. Pour chacune des bascules RS, on donne l'équation donnée par la **Sous-question e**:

$$
\begin{aligned}
Q_1 &= \overline{R_1}(Q_1' + S_1) \\
\overline{Q_1} &= \overline{S_1}\left(\overline{Q_1'} + R_1\right) \\
Q_2 &= \overline{R_2}(Q_2' + S_2) \\
\overline{Q_2} &= \overline{S_2}\left(\overline{Q_2'} + R_2\right) \\
Q_3 &= \overline{R_3}(Q_3' + S_3) \\
\overline{Q_3} &= \overline{S_3}\left(\overline{Q_3'} + R_3\right)
\end{aligned}
$$

b. Hors, on a aussi les relations $S_1 = H$, ainsi que $S_2 = D$, $R_3 = \overline{Q_1}$, $S_3 = Q_2$, $R_2 = H + \overline{Q_1}$ et $R_1 = \overline{Q_2}$. Notons de plus que Q_1 n'est pas utilisé. Finalement, on a les 5 relations

$$
\begin{aligned}
\overline{Q_1} &= \overline{H}\left(\overline{Q_1'} + \overline{Q_2}\right) \\
Q_2 &= \overline{H + \overline{Q_1}}(Q_2' + D) \\
\overline{Q_2} &= \overline{D}\left(\overline{Q_2'} + H + \overline{Q_1}\right) \\
Q_3 &= Q_1(Q_3' + Q_2) \\
\overline{Q_3} &= \overline{Q_2}\left(\overline{Q_3'} + \overline{Q_1}\right)
\end{aligned}
$$

Question 3.5

Remarque 4

L'énoncé suppose $D = 0$ pour l'entièreté de la question, puis dit « On suppose maintenant qu'à t, $D = 0$ » pour la **Sous-question b**, nous supposerons alors que $D = 0$ pour toute la question.

a. On se place avant le front descendant, donc $H = 1$ et on suppose $D = 0$. On a alors

$$
\begin{aligned}
\overline{Q_1} &= 0 \cdot \left(\overline{Q_1'} + \overline{Q_2}\right) = 0 \\
Q_2 &= \overline{1 + \overline{Q_1}}(Q_2' + D) = 0 \cdot (Q_2' + D) = 0 \\
\overline{Q_2} &= 1 \cdot \left(\overline{Q_2'} + 1 + \overline{Q_1}\right) = 1 \\
Q_3 &= 1 \cdot (Q_3' + 0) = Q_3' \\
\overline{Q_3} &= 1 \cdot \left(\overline{Q_3'} + 0\right) = \overline{Q_3'}
\end{aligned}
$$

On en déduit que $Q_3, \overline{Q_3}$ sont stables.

b. Maintenant, on a $H = 0$, mais on a déjà calculé les valeurs des Q_i avant le front, donc on connaît maintenant les valeurs des Q_i'

On a

$$
\begin{aligned}
Q_2 &= \overline{0 + \overline{Q_1}}(0 + 0) = 0 \\
\overline{Q_2} &= 1 \cdot \left(1 + 0 + \overline{Q_1}\right) = 1 \\
\overline{Q_1} &= 1 \cdot \left(0 + \overline{Q_2}\right) = \overline{Q_2} = 1 \\
Q_3 &= 0 \cdot (Q_3' + Q_2) = 0 \\
\overline{Q_3} &= 1 \cdot \left(\overline{Q_3'} + 1\right) = 1
\end{aligned}
$$

Finalement, on a bien $Q_3 = 0, \overline{Q_3} = 1$

c. Comme le montre les calculs précédents, toutes les valeurs sont stables, donc Q_3 et $\overline{Q_3}$ ne changeront pas tant que H, D gardent leur valeur.

d. On suppose maintenant $H = 1$, et on se réfère pour les valeurs Q'_i aux valeurs Q_i de la **Sous-question b**. On a alors

$$\overline{Q_1} = 0 \cdot \left(\overline{Q'_1} + \overline{Q_2}\right) = 0$$
$$Q_2 = \overline{1 + \overline{Q_1}}(Q'_2 + D) = 0 \cdot (Q'_2 + D) = 0$$
$$\overline{Q_2} = 1 \cdot \left(\overline{Q'_2} + 1 + \overline{Q_1}\right) = 1$$
$$Q_3 = 1 \cdot (Q'_3 + 0) = Q'_3$$
$$\overline{Q_3} = 1 \cdot \left(\overline{Q'_3} + 0\right) = \overline{Q'_3}$$

On en déduit que les valeurs $Q_3, \overline{Q_3}$ ne changent pas lors du front montant

Question 3.6

a. On se place avant le front descendant, donc $H = 1$ et on suppose dorénavant $D = 1$. On a alors

$$\overline{Q_1} = 0 \cdot \left(\overline{Q'_1} + \overline{Q_2}\right) = 0$$
$$Q_2 = \overline{1 + \overline{Q_1}}(Q'_2 + 1) = 0$$
$$\overline{Q_2} = 0 \cdot \left(\overline{Q'_2} + H + \overline{Q_1}\right) = 0$$
$$Q_3 = 1 \cdot (Q'_3 + 0) = Q'_3$$
$$\overline{Q_3} = 0 \cdot \left(\overline{Q'_3} + \overline{Q_1}\right) = 0$$

Notons que la bascule RS numéro 2 n'est pas dans un état stable, la valeur $\overline{Q_2}$ ne peut donc plus être interprétée comme la négation de Q_2.

On regarde maintenant ce qu'il se passe après le front descendant, on a $H = 0$ et $D = 1$, mais on a déjà calculé les valeurs des Q_i avant le front, donc on connaît maintenant les valeurs des Q'_i

On a

$$\overline{Q_2} = 0 \cdot \left(\overline{Q'_2} + H + \overline{Q_1}\right) = 0$$
$$\overline{Q_1} = 1 \cdot \left(0 + \overline{Q_2}\right) = 0$$
$$Q_2 = \overline{0 + \overline{Q_1}}(0 + 1) = Q_1 = 1$$
$$Q_3 = 1 \cdot (Q'_3 + 1) = 1$$
$$\overline{Q_3} = 0 \cdot \left(\overline{Q'_3} + 0\right) = 0$$

Finalement, on a bien $Q_3 = 1, \overline{Q_3} = 0$, ce qui correspond au comportement voulu.

Comme le montre les calculs précédents, toutes les valeurs sont stables, donc Q_3 et $\overline{Q_3}$ ne changeront pas tant que H, D gardent leur valeur.

On suppose maintenant $H = 1$, et on se réfère pour les valeurs Q'_i aux valeurs Q_i calculées précédemment. On a alors

$$\overline{Q_1} = 0 \cdot \left(\overline{Q_1'} + \overline{Q_2}\right) = 0$$
$$Q_2 = \overline{1 + \overline{Q_1}}(Q_2' + D) = 0$$
$$\overline{Q_2} = 0 \cdot \left(\overline{Q_2'} + H + \overline{Q_1}\right) = 0$$
$$Q_3 = 1 \cdot (Q_3' + 0) = Q_3' = 1$$
$$\overline{Q_3} = 0 \cdot \left(\overline{Q_3'} + \overline{Q_1}\right) = 0$$

On en déduit que les valeurs $Q_3, \overline{Q_3}$ ne changent pas lors du front montant

b. On a $\overline{Q} = \overline{Q_3}$, et on a montré que dans les deux cas ($D \in \{0, 1\}$), on a juste après le front descendant $\overline{Q_3} = \overline{D}$, on en déduit que la propriété P_1 est vraie.

On a de plus montré dans les deux cas (**Sous-question d**), que $\overline{Q_3}$ ne changeait pas lors du front montant. On en déduit que la propriété P_2 est vraie.

Question 3.7

a. On démarre dans l'état q_0, et à chaque transition prise (correspondant à la prochaine lettre lue), la lettre renvoyée est celle de l'état d'arrivée. La suite d'état obtenue avec l'entrée 1000110111 est $q_0, q_1, q_2, q_4, q_4, q_1, q_3, q_2, q_1, q_3, q_3$. La sortie associée est donc 1100101100

b. Nous allons tout de même donner une intuition, même si aucune justification n'est nécessaire pour cette question.

On peut remarquer que si on répète une même lettre, on termine dans les états q_3, q_4, dans lesquels on renvoie systématiquement 0. Dès lors qu'on change de lettre, on passe dans q_1, q_2, et on renvoie un 1.

On obtient alors que la sortie $s_0 s_1 ... s_n$ vérifie $s_0 = 1$ et pour tout $i \geq 1$ on a $s_i = (e_i \neq e_{i-1})$. La sortie possède un 1 quand la lettre lue et différente de celle lue juste avant.

Question 3.8

a. On a 5 états, on a donc besoin de 3 bits pour stocker des valeurs allant de 0 à 4.

b. Avant de faire notre table de Karnaugh, commençons par déterminer quels triplets (Q_0, Q_1, Q_2) correspondent à quel état. On obtient alors la **Fig. 5**

Q_2 \ Q_1Q_0	00	01	11	10
0	q_0	q_1	q_3	q_2
1	q_4	?	?	?

Fig. 5. – Correspondance Bits - état

Maintenant on peut remplacer dans cette table les états par leur sortie pour obtenir notre table de Karnaugh en **Fig. 6**

Q_2 \ Q_1Q_0	00	01	11	10
0	0	1	0	1
1	0	?	?	?

Fig. 6. – Table de Karnaugh de S en fonction des Q_i

Remarque 5

Commes les valeurs des « ? » nous importent peu, on peut les choisir comme elles nous arrangent. En particulier ici, pour obtenir des zones plus grandes, et donc une formule plus simple, on choisit une valeur de 1 pour le premier et troisième « ? ».

On en déduit que $S = Q_1.\overline{Q_0} + \overline{Q_1}.Q_0$

c. On a

$$\begin{aligned} S &= Q_1.\overline{Q_0} + \overline{Q_1}.Q_0 \\ &= \text{NOR}\Big(\text{NOR}\Big(Q_1.\overline{Q_0}, \overline{Q_1}.Q_0\Big), \text{NOR}\Big(Q_1.\overline{Q_0}, \overline{Q_1}.Q_0\Big)\Big) \\ &= \text{NOR}\Big(\text{NOR}\Big(\text{NOR}\Big(\overline{Q_1}, Q_0\Big), \text{NOR}\Big(Q_1, \overline{Q_0}\Big)\Big), \text{NOR}\Big(\text{NOR}\Big(\overline{Q_1}, Q_0\Big), \text{NOR}\Big(Q_1, \overline{Q_0}\Big)\Big)\Big) \end{aligned}$$

Question 3.9

a. on donne en Fig. 7 la table des états d'arrivée dépendamment de l'entrée et des valeurs Q_0, Q_1, Q_2.

EQ_2 \ Q_1Q_0	00	01	11	10
00	q_2	q_2	q_2	q_4
01	q_4	?	?	?
11	q_1	?	?	?
10	q_1	q_3	q_3	q_1

Fig. 7. – Table des états d'arrivée en fonction des Q_i et E

On en déduit les trois tables suivantes, pour D_1, D_2, D_3

EQ_2 \ Q_1Q_0	00	01	11	10
00	0	0	0	0
01	0	?	?	?
11	1	?	?	?
10	1	1	1	1

(a) Table de Karnaugh de D_0

EQ_2 \ Q_1Q_0	00	01	11	10
00	1	1	1	0
01	0	?	?	?
11	0	?	?	?
10	0	1	1	0

(b) Table de Karnaugh de D_1

EQ_2 \ Q_1Q_0	00	01	11	10
00	0	0	0	1
01	1	?	?	?
11	0	?	?	?
10	0	0	0	0

Fig. 9. – Table de Karnaugh de D_2

b. On en déduit les trois formules suivantes :

$$D_0 = E$$
$$D_1 = Q_0 + \overline{E}.\overline{Q_2}.\overline{Q_1}$$
$$D_2 = \overline{E}.Q_2 + \overline{E}.Q_1.\overline{Q_0}$$

c. On se réfère aux fonctions de la **Question 3.2**, qui n'étaient finalement pas choisies au hasard !

$$\begin{aligned} D_0 &= E \\ D_1 &= f(Q_0, Q_1, Q_2, E) \\ D_2 &= \overline{E} \cdot \left(Q_2 + \overline{Q_0} \cdot Q_1\right) \\ &= g(Q_0, Q_1, Q_2, E) \end{aligned}$$