# Bases de données II

Jef Wijsen

10 juin 2008

Répondez aux neuf questions dans les espaces réservés. Durée : 2 heures et 30 minutes

| Nom et prénom |
| --- |
| Année |

**Question 1** *(5 points)* *Mariage*$(x, y)$ est vrai si les personnes $x$ et $y$ sont mariées. L'ordre des sexes n'est pas fixé : on peut donc avoir *Mariage*(Leopold, Astrid) et *Mariage*(Mathilde, Philippe) dans la même base de données. *Female*$(x)$ est vrai si $x$ est du sexe féminin, par exemple, *Female*(Astrid) et *Female*(Elisabeth). Une *célibatrice* est une personne du sexe féminin qui n'est pas mariée. Donnez un programme datalog$^\neg$ qui rend toutes les célibatrices. Démontrez que votre programme est stratifié. Utilisez des noms "suggestifs" pour les prédicats, par exemple, *Celebatrice*$(x)$ serait préférable à *ans*$(x)$.

**Question 2** *(15 points)* Soit $P$ le programme datalog$^\neg$ suivant :

$$\begin{aligned} R(x,y) &\leftarrow E(x), E(y) \\ Q(x,y) &\leftarrow F(x,y), \neg R(x,y) \end{aligned}$$

Soit $I = \{E(1), E(2), F(1,2), F(1,3)\}$.

1. Quel est le modèle minimal qui contient $I$, selon la sémantique stratifiée?
2. Soit $\tau$ la traduction de $P$ en logique du premier ordre :

   $$\forall x \forall y((E(x) \wedge E(y)) \rightarrow R(x,y))$$
   $$\forall x \forall y((F(x,y) \wedge \neg R(x,y)) \rightarrow Q(x,y))$$

   Donnez **tous** les modèles minimaux de $\tau$ qui contiennent $I$.
3. Choisissez un modèle logique minimal (i.e. un modèle parmi les réponses à 2) qui n'est pas un point fixe pour l'opérateur $T_P$ et montrez l'application de l'opérateur $T_P$ jusqu'au point fixe.

**Question 3** *(10 points)* Un graphe non-orienté est une paire $(V, E)$ où

1. $V$ un ensemble de *nœuds*; et
2. $E$ est un ensemble d'*arêtes*. Chaque arête est une paire $\{x, y\}$ avec $x \neq y$ et $x, y \in V$.

Un nœud $x \in V$ est *monolien* si $x$ appartient à exactement une arête de $E$. Un nœud $x \in V$ est *adjacent* à un autre nœud $y \in V$ si $\{x, y\} \in E$. Un nœud n'est jamais adjacent à lui-même. Par exemple, si $V = \{a, b, c, d, e\}$ et $E = \{\{a, b\}, \{b, c\}\}$, alors les nœuds monoliens sont $a$ et $c$.

Un graphe est stocké dans une base de données de manière évidente :

1. si $x \in V$, alors $V(x)$ est un fait de la base de données;
2. si $\{x, y\} \in E$, alors $E(x, y)$ ou $E(y, x)$ (le choix est libre) est un fait de la base de données.

**Donnez un programme** datalog$^\neg$ **qui rend tout nœud qui n'est pas adjacent à un nœud monolien.** Démontrez que votre programme est stratifié. Utilisez des noms "suggestifs" pour les prédicats, par exemple, $\mathit{Monolien}(x)$ serait préférable à $M(x)$. Expliquez votre démarche si elle n'est pas claire à partir des noms des prédicats.

*Aide :* Rappelez-vous que $\sigma_{A \neq B} R$ peut être écrit comme $R - (\sigma_{A=B} R)$. Donc, l'inégalité peut être exprimée à l'aide de la négation.

**Question 4** *(15 points)* Soient $x_0, x_1, x_2, \ldots$ des variables distinctes. Pour chaque $n \in \{1, 2, 3, \ldots\}$, soit $P_n$ la requête conjonctive :

$$P_n : \quad ans(\text{“yes”}) \quad \leftarrow \quad \{R(x_i, x_j) \mid i, j \in \{0, 1, \ldots, n\}, i \neq j\} \ .$$

Donc,

$$\begin{array}{lllll}
P_1 : & ans(\text{“yes”}) & \leftarrow & R(x_0, x_1), R(x_1, x_0) \\
P_2 : & ans(\text{“yes”}) & \leftarrow & R(x_0, x_1), R(x_0, x_2), R(x_1, x_2), R(x_1, x_0), R(x_2, x_0), R(x_2, x_1) \\
P_3 : & ans(\text{“yes”}) & \leftarrow & R(x_0, x_1), R(x_0, x_2), R(x_0, x_3), R(x_1, x_2), R(x_1, x_3), R(x_2, x_3), \\
 & & & R(x_1, x_0), R(x_2, x_0), R(x_3, x_0), R(x_2, x_1), R(x_3, x_1), R(x_3, x_2) \\
 & & \vdots &
\end{array}$$

Démontrez que pour tout $n \in \{1, 2, 3, \ldots\}$ :

1. $P_{n+1} \subseteq P_n$
2. $P_n \not\subseteq P_{n+1}$
3. $P_n$ est minimal

Le DTD suivant exprime que chaque pièce peut être composée de zéro ou plusieurs autres pièces. On peut supposer que chaque pièce a un nom qui est unique dans le document XML.

```
<!-- This file is called parts.dtd -->
<!ELEMENT composition (part)*>
<!ELEMENT part (part)*>
<!ATTLIST part name CDATA #REQUIRED>
<!ATTLIST part manufacturer CDATA #REQUIRED>
```

Les questions qui suivent concernent le document valide qui suit :

```
<?xml version="1.0"?>
<!DOCTYPE composition SYSTEM "parts.dtd">
<composition>
<part name="voiture" manufacturer="renault">
   <part name="moteur" manufacturer="renault">
      <part name="bloc-moteur" manufacturer="renault"/>
      <part name="boite-de-vitesses" manufacturer="shimano"/>
      <part name="systeme-de-refroidissement" manufacturer="bosch"/>
   </part>
   <part name="roues" manufacturer="renault">
      <part name="jante" manufacturer="renault"/>
      <part name="pneu" manufacturer="michelin">
         <part name="soupape" manufacturer="bosch"/>
      </part>
   </part>
</part>
<part name="ordinateur" manufacturer="dell">
   <part name="processeur" manufacturer="intel"/>
   <part name="ecran" manufacturer="iiyama"/>
</part>
</composition>
```

**Question 5** *(5 points)* Écrivez une expression XPath qui rend tous les nœuds attributs "`name`" liés à une pièce fabriquée par `renault`. Il y en a cinq : `name="voiture"`, `name="moteur"`, `name="bloc-moteur"`, `name="roues"`, `name="jante"`.

**Question 6** *(5 points)* Écrivez une expression XPath qui rend tous les nœuds attributs "`name`" liés à une pièce qui ne contient aucune pièce fabriquée par michelin. Il y en a neuf : `name="moteur"`, `name="bloc-moteur"`, `name="boite-de-vitesses"`, `name="systeme-de-refroidissement"`, `name="jante"`, `name="soupape"`, `name="ordinateur"`, `name="processeur"`, `name="ecran"`. Notez que `name="voiture"` n'est pas dans cette liste, parce qu'une voiture contient la "sous sous pièce" pneu fabriquée par michelin.

**Question 7** *(5 points)* Écrivez une expression XPath qui rend tous les nœuds attributs "`name`" liés à une pièce qui elle-même n'est pas fabriquée par renault, mais qui est utilisée dans une pièce fabriquée par renault. Il y en a quatre : `name="boite-de-vitesses"`, `name="systeme-de-refroidissement"`, `name="pneu"`, `name="soupape"`.

**Question 8** *(10 points)*
Écrivez une requête XQuery qui rend toutes les combinaisons

```
<a>  <b>x</b>
     <c>y</c>  </a>
```

où `x` et `y` sont des pièces distinctes fabriquées par une même entreprise. Trouvez une solution telle que si l'information `<a><b>x</b><c>y</c></a>` est affichée, alors l'information "inverse" `<a><b>y</b><c>x</c></a>` n'est plus affichée. Voici un résultat correct :

```
<a><b>voiture</b><c>moteur</c></a>
<a><b>voiture</b><c>bloc-moteur</c></a>
<a><b>voiture</b><c>roues</c></a>
<a><b>voiture</b><c>jante</c></a>
<a><b>moteur</b><c>bloc-moteur</c></a>
<a><b>moteur</b><c>roues</c></a>
<a><b>moteur</b><c>jante</c></a>
<a><b>bloc-moteur</b><c>roues</c></a>
<a><b>bloc-moteur</b><c>jante</c></a>
<a><b>systeme-de-refroidissement</b><c>soupape</c></a>
<a><b>roues</b><c>jante</c></a>
```

**Question 9** *(10 points)*

Donnez une feuille XSLT qui copie les nœuds élément, en remplaçant chaque balise `part` par le nom de la pièce tel qu'indiqué par l'attribut `name`, tout en supprimant les autres attributs. Voici le résultat désiré :

```
<composition-traduite>
   <voiture>
      <moteur>
         <bloc-moteur />
         <boite-de-vitesses />
         <systeme-de-refroidissement />
      </moteur>
      <roues>
         <jante />
         <pneu>
            <soupape />
         </pneu>
      </roues>
   </voiture>
   <ordinateur>
     <processeur />
     <ecran />
   </ordinateur>
</composition-traduite>
```

*Aide :* Voici deux façons pour créer un élément de type `mylabel` en XSLT :

```
<mylabel>
 <!-- Creation du contenu -->
</mylabel>
```

a le même effet que

```
<xsl:element name="mylabel">
 <!-- Creation du contenu -->
</xsl:element>
```

Néanmoins, l'usage de `xsl:element` permet la création d'un élément dont le nom est calculé à l'aide d'une expression, à condition de l'encadrer par des accolades ({}). Par exemple,

```
<xsl:element name="{@name}">
   <!-- Creation du contenu -->
</xsl:element>
```