Correction — Examen Web Sémantique — Sujet A

CONFIDENTIEL
Ne pas diffuser

Cours #2, #3 et #4 — Réservé à l'enseignant

Exercice 1 — Vrai ou Faux (6 pts)

0,5 pt par bonne réponse, –0,25 par erreur.

#	Réponse	Explication
1	**V**	Définition même de RDF : (sujet, prédicat, objet).
2	**V**	X est unifié avec paul (pos. 1), Y avec jean (pos. 2).
3	**F**	C'est le **chaînage arrière** qui part du but. Le chaînage avant part des faits.
4	**F**	∃ signifie « au moins un ». ∀ signifie « seulement des ». ∃ a_mangé.Viande = « a mangé **au moins un** aliment carné ».
5	**V**	L'incohérence est sémantique, pas syntaxique. Exemple : une classe déclarée équivalente à sa négation.
6	**F**	SPARQL a **CONSTRUCT**, **INSERT**, **DELETE** pour créer/modifier des triplets.
7	**V**	SELECT ?s WHERE { ?s rdf:type :Homme } retourne toutes les instances de Homme.
8	**V**	a est un alias de `rdf:type` en Turtle.
9	**V**	HermiT est un raisonneur OWL 2 complet, écrit en Java.
10	**F**	∀ signifie « tous les », pas « au moins un ». ∀ a_ingrédient.Fromage = « tous les ingrédients sont du fromage » (une pizza 100 % fromage).
11	**F**	Fuseki est écrit en **Java** (Apache Jena). Notre mini_fuseki.py est une réimplémentation en Python.
12	**V**	Les objets RDF peuvent être des URIs (ressources) ou des littéraux (chaînes, nombres, dates).

Total : 12 × 0,5 pt = 6 pts

Exercice 2 — Triplets, Prolog, raisonnement (5 pts)

Q 2.1 — Traduction en triplets (1,5 pt)

0,5 pt par bonne traduction.

2.1.1 `:Montpellier rdf:type :Ville .`

2.1.2 `:Montpellier :situéeDans :Sud .`

2.1.3 `a_du_soleil(X) ← est_dans(X, Sud) ∧ est_une(X, Ville)`
Variante : `soleil(X) ← ville(X) ∧ dans(X, sud)`

Q 2.2 — Chaînage avant (2 pts)

Q 2.2.1 (1 pt) :

Règle `amitie(X) ← aime(X, Y) ∧ est_un(Y, animal)` :
aime(medor, felix) et est_un(felix, animal) → **amitie(medor)**.
aime(medor, felix) et est_un(medor, animal) → **amitie(medor)** (déjà déduit, même résultat).
Soit un fait déduit : amitie(medor).

Q 2.2.2 (0,5 pt) :

`copain(X, Y) ← aime(X, Y) ∧ aime(Y, X)`

Q 2.2.3 (0,5 pt) :

Oui : `¬aime(chien_du_voisin, X)` est une **négation par échec**. Si on ne sait pas si le chien du voisin aime X, on ne peut pas conclure : le système doit supposer que tout ce qui n'est pas prouvé est faux (hypothèse du monde clos). Sans cela, la règle ne peut pas se déclencher.

Q 2.3 — Unification (1,5 pt)

2.3.1 `aime(medor, X)` et `aime(Y, felix)` → ✓ {Y/medor, X/felix} **(0,5 pt)**

2.3.2 `est_un(Z, animal)` et `est_un(medor, Z)` → ✗ **Échec**. Z = medor (pos. 1) ET Z = animal (pos. 2) → conflit. **(0,5 pt)**

2.3.3 `pere(X, mere(X))` et `pere(jean, mere(paul))` → ✗ **Échec**. X = jean (pos. 1), mais mere(jean) ≠ mere(paul) car jean ≠ paul. **(0,5 pt)**

Remarque pédagogique : La Q 2.3.2 est un piège classique. Les étudiants confondent souvent variable positionnelle et valeur. Insistez sur le fait qu'une variable a la même valeur partout où elle apparaît.

Exercice 3 — OWL (5 pts)

Q 3.1 — Hiérarchie et héritage (1,5 pt)

```
Animal
├── Mammifère
│   └── ChauveSouris  (⊓ ∃ sait.Voler)
└── Oiseau
    └── Rapace
        └── Aigle
    └── Moineau
```

Barème : 0,5 pt pour la structure, 0,5 pt pour ChauveSouris sous Mammifère et pas Oiseau, 0,5 pt pour Rapace correctement placé.

Q 3.2 — Restrictions (2 pts)

3.2.1 (0,5 pt) : $\text{ChauveSouris} \sqsubseteq \text{Mammifère} \sqcap \exists\ \text{sait.Voler}$

3.2.2 (0,5 pt) : $\text{Moineau} \sqsubseteq \text{Oiseau} \sqcap \forall\ \text{mange.Graine}$

3.2.3 (0,5 pt) : $\text{Aigle} \sqsubseteq \text{Rapace} \sqcap \exists\ \text{chasse.(Mammifère} \sqcap \text{TaillePetite)}$

3.2.4 (0,5 pt) : $\text{RapaceDiurne} \sqsubseteq \text{Rapace} \sqcap \forall\ \text{chasseAuMoment.Jour}$

Barème : Pénaliser de 0,25 pt si ∃ et ∀ sont inversés.

Q 3.3 — Cohérence (1,5 pt)

L'ontologie est incohérente. Trois problèmes :

- **titi n'a pas d'amis :** $\text{titi} \sqsubseteq \neg(\exists\ \text{a\_pour\_amie.Chose})$ signifie que titi n'a strictement aucun ami (même pas lui-même). C'est une classe très solitaire.
- **titi a un ami non-moineau :** $\text{titi} \sqsubseteq \exists\ \text{a\_pour\_amie.}\neg\text{Moineau}$ exige l'existence d'au moins un ami. C'est la **contradiction directe** avec la ligne précédente (∃ vs $\neg\exists \rightarrow \bot$).
- **L'Aigle :** Rapace ⊑ ∃ chasse.Mammifère, mais Aigle n'a pas de restriction de chasse — ce n'est pas une contradiction en soi (héritage non bloqué), mais c'est un manque de précision.

Barème : 1 pt pour la détection de la contradiction principale (∃ vs ¬∃), 0,5 pt pour l'explication claire.

Note : Le piège « ¬(∃ a_pour_amie.Chose) » vs « ∃ a_pour_amie.¬Moineau » est volontaire. Les étudiants doivent reconnaître ≤ 0 amis $\cap \geq 1$ ami $= \bot$.

Exercice 4 — SPARQL (4 pts)

Q 4.1 — SELECT (1 pt)

```
PREFIX : <http://ex.org/>
SELECT ?ville WHERE {
  ?ville rdf:type :Ville .
  ?ville :dans :Sud .
}
```

Barème : 0,5 pt pour le pattern, 0,5 pt pour le préfixe.

Q 4.2 — FILTER (1 pt)

```
PREFIX : <http://ex.org/>
SELECT ?ville ?population WHERE {
  ?ville rdf:type :Ville .
  ?ville :population ?population .
  FILTER(xsd:integer(?population) > 500000)
} ORDER BY DESC(?population)
```

Barème : 0,3 pt FILTER, 0,3 pt conversion xsd:integer, 0,2 pt ORDER BY DESC, 0,2 pt syntaxe.

Accepté sans xsd:integer si l'étudiant compare des entiers.

Q 4.3 — CONSTRUCT (1 pt)

```
PREFIX : <http://ex.org/>
CONSTRUCT {
  ?ville :a_du_soleil true
} WHERE {
  ?ville :dans ?region .
  ?region :soleil true .
}
```

Barème : 0,5 pt pour la partie CONSTRUCT, 0,5 pt pour le pattern WHERE avec jointure.

Variante acceptée : deux requêtes séparées (une pour true, une pour false).

Q 4.4 — Culture (1 pt)

Deux différences possibles (0,5 pt chacune) :

1. SQL travaille sur des **tableaux** (lignes/colonnes), SPARQL sur des **graphes** (triplets).
2. SPARQL utilise des **URIs** comme identifiants (pas de clés primaires numériques).
3. SPARQL permet les **property paths** (chemins d'arêtes avec expressions régulières).
4. SPARQL gère le **vide** (OPTIONAL) de manière explicite.
5. SPARQL distingue **pas de résultat** vs **NULL** (absence de triplet ≠ valeur nulle).