TP1 - Un traducteur de RDF/Turtle vers RDF/Ntriples

Vous devez rendre ce TP (ainsi que la partie Bonus si vous l'avez faite) au plus tard pour le début de la troisième séance de TP (semaine du 7 au 11 octobre).

1 Cadre et préparation du TP

Le Web sémantique est basé sur RDF, un modèle pour la représentation de données interopérables. L'unité de base en RDF est le triplet, correspondant à une phrase simple de la forme "sujet verbe complément" (le verbe est appelé *prédicat* et le complément est appelé *objet*). Le sujet et le prédicat sont des *entités* (noms entre chevrons <...>) et l'objet est soit une entité soit un *texte* (entre guillemets "..."). Il existe plusieurs notations pour RDF et on souhaite ici effectuer la conversion depuis (un fragment de) Turtle (extension `.ttl`) vers Ntriples (extension `.nt`). Le problème va être présenté par un exemple et vous devrez proposer une solution générale.

Voici un texte Turtle décrivant l'entité `<poly117>` comme étant un poly (type), ayant deux auteurs (Ridoux et Ferre) et ayant pour titre "Compilation". L'entité `<Ridoux>` est défini comme étant une personne et un professeur.

```
<poly117>
  <type> <poly> ;
  <auteur> <Ridoux>, <Ferre> ;
  <titre> "Compilation" .
<Ridoux> <type> <personne>, <professeur> .
```

Ntriples n'autorise qu'une représentation à plat, un triplet par ligne. Le même exemple peut ainsi être représenté comme suit.

```
<poly117> <type> <poly> .
<poly117> <auteur> <Ridoux> .
<poly117> <auteur> <Ferre> .
<poly117> <titre> "Compilation" .
<Ridoux> <type> <personne>.
<Ridoux> <type> <professeur> .
```

On voit donc qu'en Turtle, le point-virgule (;) permet de faire une liste de prédicats pour un même sujet, et la virgule (,) permet de faire une liste d'objets pour un même prédicat. Ces séparateurs jouent le même rôle que la conjonction de coordination 'et' en français.

> **Pour démarrer ce travail, nous vous demandons de proposer une ASD définissant la syntaxe abstraite du sous-ensemble du langage Turtle utilisé ici. Veillez à ce que l'ASD reflète la structure sémantique de Turtle. Faites valider votre ASD par votre enseignant avant de poursuivre.**

2 Réalisation du traducteur de Turtle vers Ntriples

Pendant les deux premières séances de TP PDS, vous allez réaliser un traducteur de fichiers Turtle vers Ntriples. En vous appuyant sur votre ASD de Turtle (voir section 1), vous allez construire le traducteur demandé en utilisant ANTLR/Java ou OCaml. Nous recommandons de suivre les étapes suivantes dans la réalisation du traducteur.

2.1 Implémentation de l'ASD

Il s'agit d'implémenter l'ASD de Turtle que vous avez définie comme un ensemble de classes/types du langage de programmation choisi. Vous suivrez le schéma de traduction donné en cours. En Java, lorsqu'un type a un seul variant, on pourra fusionner la classe abstraite et la classe concrète par souci de concision du code.

2.2 ASD attribuée pour la génération du Ntriples

Il s'agit de définir une ASD attribuée définissant le calcul du Ntriples à partir d'un AST Turtle. Vous définirez l'ASD attribuée sur papier avant de l'implémenter. Pour rappel, tous les calculs devront passer par les attributs et aucune variable globale ne devra être utilisée. Vous pourrez écrire quelques AST manuellement dans le fichier `Main.java` sous forme d'objets Java pour tester votre ASD, en utilisant par exemple les fichiers de tests comme modèles.

2.3 Analyse lexicale et syntaxique

Définissez la grammaire de Turtle, en distinguant bien les règles lexicales et les règles syntaxiques. Implémentez-la, soit en ANTLR si vous avez choisi Java, soit avec ocamllex et les *stream parsers* si vous avez choisi OCaml.

2.4 Production d'AST par l'analyseur syntaxique

En attribuant votre grammaire, produisez un AST qui suive votre ASD de Turtle.

2.5 Test de votre traducteur

Vous testerez soigneusement l'ensemble de la chaîne de traduction. Vous disposez de deux fichiers Turtle exemples, `test1.ttl` et `test2.ttl`, pour tester vos analyseurs.

3 Environnement

3.1 Environnement ANTLR/Java

Commencez par copier le dossier de sources `/share/m1info/PDS/tp/TP1/java/` dans votre répertoire personnel.

Vous allez utiliser le moteur de production Gradle.

Pour construire le projet, lancez la commande `./gradlew build` dans le dossier du projet. Gradle télécharge alors automatiquement les dépendances du projet et construit tous les fichiers nécessaires. Pour lancer le résultat de la compilation, exécutez `java -jar build/libs/TP1.jar`.

La configuration de Gradle est dans le fichier `build.gradle`. Lorsque vous commencerez à utiliser ANTLR, décommentez toutes les lignes commentées dans ce fichier pour indiquer à Gradle que votre projet dépend maintenant d'ANTLR4.

Les sources java se trouvent dans le dossier `src/main/java/TP1` et les grammaires dans le dossier `src/main/antlr/TP1/`. Les fichiers `.java` générés à partir des grammaires par ANTLR se trouvent dans le dossier `build/generated-src/antlr/main/`.

3.2 Environnement OCaml

Commencez par copier le dossier de sources `/share/m1info/PDS/tp/TP1/ocaml/` dans votre répertoire personnel.

Pour construire le projet, lancez la commande `make` dans le dossier du projet[1]. Pour lancer le résultat de la compilation, exécutez `./main.native`.

Les fichiers `.ml` générés par ocamllex se trouvent dans le dossier `_build/`.

4 Extensions possibles

S'il vous reste du temps, voici deux extensions possibles intéressantes : traiter les noeuds anonymes de Turtle et produire une version XML de Turtle. Ces deux extensions sont indépendantes.

4.1 Noeuds anonymes

À la place d'un sujet ou d'un objet, on peut avoir un noeud anonyme, donc sans nom mais avec éventuellement des couples propriété-valeurs associés. Cela permet par exemple de décrire les 2 versions du poly de compilation, sans avoir à nommer chaque version.

```
<poly117> <version>
   [ <annee> "2007" ;
     <nbpages> "147" ] ,
   [ <annee> "2011" ;
```

1. Si l'exécutable `ocamlbuild` est manquant, exécutez ces commandes : `opam init --use-internal-solver; eval \`opam config env\`; opam install ocamlbuild`

```
    <nbpages> "163" ] .
```

La traduction en Ntriples suppose de générer des identificateurs uniques de noeuds anonymes, de la forme _: *id*. La traduction Ntriples de l'exemple ci-dessus est la suivante.

```
<poly117> <version> _:v1 .
_:v1 <annee> "2007" .
_:v1 <nbpages> "147" .
<poly117> <version> _:v2 .
_:v2 <annee> "2011" .
_:v2 <nbpages> "163" .
```

4.2 Production de RDF/XML

Il existe aussi un format XML pour RDF. On propose de définir une nouvelle ASD attribuée pour le produire. La version XML du premier exemple Turtle est comme suit.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xml:base="http://mydomain.org/myrdf/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description rdf:about="poly117">
    <rdf:type rdf:resource="poly"/>
    <auteur rdf:resource="Ridoux"/>
    <auteur rdf:resource="Ferre"/>
    <titre>Compilation</titre>
</rdf:Description>
<rdf:Description rdf:about="Ridoux">
    <rdf:type rdf:resource="personne"/>
    <rdf:type rdf:resource="professeur"/>
</rdf:Description>
</rdf:RDF>
```

La version XML du second exemple avec noeuds anonymes est comme suit.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xml:base="http://mydomain.org/myrdf/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description rdf:about="poly117">
    <version rdf:parseType="Resource">
        <annee>2007</annee>
        <nbpages>147</annee>
    </version>
```

```
    <version rdf:parseType="Resource">
        <annee>2011</annee>
        <nbpages>163</annee>
    </version>
</rdf:Description>
</rdf:RDF>
```