Practical Examples for Knowledge Representation and Reasoning (KRR)

Task 1: Construct RDF Triples

Objective: Practice creating RDF triples to represent simple statements.

Example 1

Represent the following statements in RDF triples using Turtle syntax:

- "Alice is a person."
- "Bob knows Alice."
- "Alice works at Acme Corporation."

Solution:

```
@prefix ex: <http://example.org/> .

ex:Alice a ex:Person.                          # Alice is a person
ex:Bob ex:knows ex:Alice.                      # Bob knows Alice
ex:Alice ex:worksAt ex:AcmeCorporation.        # Alice works at Acme Corporation
```

Explanation:

- `ex:Alice` is the subject.
- `ex:Person` is the class representing "Person."
- `ex:knows` and `ex:worksAt` are predicates describing relationships.
- `ex:AcmeCorporation` is the object for "worksAt."

Task 2: Define an Ontology

Objective: Create classes and properties in RDFS for a university domain.

Example 2

Define an ontology for the university domain that includes:

- Classes: `Student`, `Professor`, `Course`, and `Department`.
- Properties: `teaches`, `enrolledIn`, `memberOf`.
- Subclass: `GraduateStudent` is a subclass of `Student`.

Solution:

```
@prefix ex: <http://example.org/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

# Class definitions
ex:Student a rdfs:Class.
ex:GraduateStudent a rdfs:Class;
    rdfs:subClassOf ex:Student.  # GraduateStudent is a subclass of Student
ex:Professor a rdfs:Class.
ex:Course a rdfs:Class.
ex:Department a rdfs:Class.

# Property definitions
ex:teaches a rdf:Property;
    rdfs:domain ex:Professor;
    rdfs:range ex:Course.          # Professors teach courses

ex:enrolledIn a rdf:Property;
    rdfs:domain ex:Student;
    rdfs:range ex:Course.          # Students enroll in courses

ex:memberOf a rdf:Property;
    rdfs:domain ex:Person;
    rdfs:range ex:Department.      # Persons are members of departments
```

Explanation:

- `rdfs:Class` defines new classes such as `Student`, `GraduateStudent`, etc.
- `rdfs:subClassOf` establishes subclass relationships.
- The properties (`teaches`, `enrolledIn`, and `memberOf`) define relationships between classes, with domains and ranges specifying the expected types of subject and object.

Task 3: Perform Logical Inference

Objective: Apply logical reasoning over RDF data.

Example 3

Given the following RDF triples:

- `ex:Planet rdfs:subClassOf ex:CelestialBody.`
- `ex:Jupiter rdf:type ex:Planet.`

Question: What can be inferred about `ex:Jupiter`?
Solution:

```
ex:Jupiter rdf:type ex:CelestialBody.
```

Explanation:

- Since `Jupiter` is a `Planet` and `Planet` is a subclass of `CelestialBody`, we can infer that `Jupiter` is a `CelestialBody`.

Task 4: Write SPARQL Queries

Objective: Practice querying RDF data using SPARQL.

Example 4

Given the following RDF data about books and authors, write SPARQL queries to:

- Retrieve all books written by a specific author.
- List all authors who have written more than three books.

```
@prefix ex: <http://example.org/> .

ex:Book1 a ex:Book;
    ex:writtenBy ex:Author1.

ex:Book2 a ex:Book;
    ex:writtenBy ex:Author1.

ex:Book3 a ex:Book;
    ex:writtenBy ex:Author2.

ex:Book4 a ex:Book;
    ex:writtenBy ex:Author2.

ex:Book5 a ex:Book;
    ex:writtenBy ex:Author1.
```

1. Retrieve all books written by `ex:Author1`:

```
PREFIX ex: <http://example.org/>

SELECT ?book
WHERE {
  ?book ex:writtenBy ex:Author1.
}
```

Result:

- `ex:Book1`
- `ex:Book2`

- `ex:Book5`

This query selects all books where the `writtenBy` property has `ex:Author1` as the value.

2. List all authors who have written more than three books:

```
PREFIX ex: <http://example.org/>

SELECT ?author (COUNT(?book) AS ?bookCount)
WHERE {
  ?book ex:writtenBy ?author.
}
GROUP BY ?author
HAVING (COUNT(?book) > 3)
```

Result:

- No authors meet the condition since `ex:Author1` has written 3 books and `ex:Author2` has written 2 books.

Explanation:

- `COUNT(?book)` counts how many books are associated with each author.
- `HAVING (COUNT(?book) > 3)` filters to only include authors with more than three books.

Task 5: Explore Linked Data

Objective: Understand how linked data integrates information from different sources.

Example 5

Consider DBpedia, which extracts structured information from Wikipedia and represents it as linked data. Here's how linked data works:

Each Wikipedia entity (e.g., a person, place, or thing) has a URI (e.g., `http://dbpedia.org/resource/Albert_Einstein`). Properties in DBpedia link data points, such as:

- `dbpedia:Albert_Einstein dbpedia-owl:birthPlace dbpedia:Ulm.`
- `dbpedia:Albert_Einstein dbpedia-owl:knownFor dbpedia:Theory_of_relativity.`

Research Task:

- Explore how DBpedia extracts and structures this information.
- Explain how URIs are used to link resources across datasets.

Solution:

- **Linked Data**: URIs like `http://dbpedia.org/resource/Albert_Einstein` are used to uniquely identify entities. These URIs can be referenced across datasets, allowing different datasets to be linked and queried together.
- **DBpedia Example**: You can retrieve structured data about Albert Einstein's birthplace, field of work, etc., by querying DBpedia using SPARQL.