CSE 579: Knowledge Representation & Reasoning

Module 7:
Ontology Languages

Ali Altunkaya
Spring 2023

Outline

1. Definition of Ontology
2. Syntactic web vs Semantic Web
3. RDF (Resource Description Framework)
4. RDFS (Schema)
5. Knowledge Graph
6. Description Logics (another ontology language)
7. Reasoning Problems
8. OWL:Web Ontology Language (another ontology language)

1. Ontology

What is Ontology?

- Taxonomy is the science of classification.
- Ontology is a more general concept than taxonomy. For a specific domain, ontology try to
  - explain all entities,
  - classify them and
  - describe relations between them.

- Sometimes we use different names for the same thing:
  - “Heart Attack” = “Myocardial Infarction”
  - “Location” = “Place”
- This is a problem in Syntactic Web.
- Semantic web can overcome this problem using Ontologies.

2. Syntactic Web

Syntactic Web

- The current state of the web.
- All data is written in HTML markup language. Syntax is ok, but we cannot get "meaning" from data.
  - HTML tags is for syntax (Bold, Font, Size, Color, Indentation, Paragraph)
  - But there is no tag for meaning, example
    - Some pages use "location: COOR170"
    - Other pages use "place: COOR170"
- Because of that, we cannot make automated reasoning from the web.
- Google Pagerank, usually assigns ranking to web pages for "keywords".
  - (Ok, probably their algorithm is more complex, but this is the essence)
  - For a keyword e.g. "knowledge representation", there is a list of web pages in a specific ranking.
- But why Google cannot answer complex questions? Because there is no semantic web...
  - "Where is the location of CSE 579 at ASU in Fall 2022?"

2. Semantic Web

Semantic Web (XHTML + Ontology)

- Solution: XML markup with “meaningful” tags such as:
  - “location”
  - “time”
  - “founder”
- What happens if people use different tags for the same “meaning”?
- We also need to add ontologies: agreement on the meaning of annotations.
  - All web pages use “location”.
  - Or you need to add relation between them, eg: “location” = “place”.
- Once we created our Ontology, we need to convert it into FOL or ASP for reasoning. There is a trade-off between:
  - Automated Reasoning (SAT is an NP-Complete problem)
  - Representation (More expressive is better)
- Therefore, we should use “sufficient” expressive representation to model the domain of interest.

3. RDF (Resource Description Framework)

RDF is an Ontology language.

- A data model for representing information about entities (or resources) on the web.
  - Everything on the web is a resource.
- RDF provides a graph like data model:
  - Subject —Predicate→ Object or
  - Subject —Property→ Object
- In RDF, everything is a resource.
  - All of these are also a resource: Subject, Object, Property.
  - And each resource has a unique identifier: IRI (Intern. Resource Identifier)

3. RDF (Resource Description Framework)

Three Basic Ideas of **RDF**:

1-) Resources: Anything on the web is a resource, such as

- Objects: authors, people, hotels, individuals, etc
- Properties or relations: written by, age, title, sibling of, etc

- Every resource has a global, worldwide unique identifier called IRI. Like URL or a unique ID number.

2-) Properties: A special kind of resources, which describe the properties of a resource or relations between resources (objects, subjects etc.), such as: “founded by”, “written by” etc. or “age”, “title”, “”.

3-) Statements: We combine “resources” and “properties” in the statements. Within the graph, nodes are resources, edges are properties. FOL atoms.

- Statements assert the properties of resources:
  - “Computer23” —“owned by”→ “Craig”
- A statement is a triplet:
  - Subject —Property→ Object (objects can be resource or value)
  - Resource —Property→ Resource or Value

3. RDF (Resource Description Framework)

Reify: make into an object.

Reification: RDF has two arguments. But still we can represent complex statements using reification in 3 steps:

- We convert the whole statement into an object.
- Make an abstract thing concrete.
- Then change its properties one by one.

For example, we can represent a table using reification:

- Convert whole table into an object.
- Target only a single item from the table.
- Change its value one by one.

4. RDF Schema

RDF is an ontology language to describe resources and relations between them. But it does not define semantics within the domain. So non-sense statements are allowed in RDF. For example:

- “CSE579” → “is taught by” → ”CSE259”

RDF Schema is an extension of RDF to represent ontological information better via restrictions. It allow users to define semantics using:

- Classes and Properties
- Class Hierarchies and Inheritance (ex: Professor → Academics)
- Property Hierarchies (ex: taughtBy → involvedIn, rents→residesAt)

Classes and Instances: RDFS can distinguish between classes and instances using “rdf:type”. (But RDF can't)

- Class: Sets of individuals sharing properties: instructors, courses, etc.
- Instance: Individual objects belong to a class: CSE579, Dr. Lee, etc.

4. RDF Schema

Restrictions: We can define semantic rules by using the restriction on the resources.

Subject —Property→ Object

- Range restriction: restriction on the values of object.
- Domain restriction: restriction on the values of subjects.

We need class & property hierarchies for restrictions.

Class Hierarchies: Classes can be organized in hierarchies. “Professor” is a subclass of “Academic Staff”.

- A class can have multiple subclasses (or children) and also can have multiple superclasses (or parents).
- Instances will inherit abilities from its class and all of its superclasses.

Property Hierarchies: Properties can be organized in hierarchies too. For ex:

- “isTaughtBy” is a sub-property of “involves”.
- “residesAt” is a sub-property of “rents”.

4. RDF Schema

Core Properties: There are many classes in RDFS, but these are just the most important ones:

We can define classify our entities by using them:

- rdfs:Resource
- rdfs:Class
- rdfs:Literal
- rdfs:Property
- rdfs:Statement

We can define types and hierarchies by using them:

- rdfs:type
- rdfs:subClassOf
- rdfs:subPropertyOf

We can put restriction by using them:

- rdfs:domain
- rdfs:range

4. RDF Schema

Limitations of RDF Schema:

- Does not support negation.
  - We can define “Dr. Lee is a Professor.”
  - But we cannot define “Dr. Lee is not a Professor.”
- Does not provide constructors to define Classes.
- Does not support instance-level relations (although it supports class-level relations via subClassOf and subPropertyOf)

Next chapter, **Description Logic** (another Ontology Language) will overcome of these limitations.

Axiomatic Semantics of RDF Schema:

- First we model our domain using an Ontology Language such as RDFS.
- Then we need to formalize these Statements, by translating them into FOL.
- So we make the semantics unambiguous and make automated-reasoning.
- Slides show how to convert some common RDFS Statements into FOL rules or programs.

5. Knowledge Graph

Knowledge Graph is an ontology where:

- Nodes describes real-world entities/objects
- Edges defines relations between those entities
- Reasoner can query it and derive new knowledge
- Inheritance, classes, instances might be possible.

How to create Knowledge Graph?

- Some of them are created and organized by ontology engineers.
- Some of them are automated from websites + manual creation.

6. Description Logics (DL)

Description Logics is another Ontology Language, more powerful than RDFS.

- It is used to Represent Knowledge of a domain, but cannot do Reasoning.
- We need to convert DL to FOL, so we can make Reasoning.
- FOL is more expressive but computing its Satisfiability is undecidable.
  - DL is a decidable fragment (or subset) of FOL.
  - They apply some constraints to make DL decidable.

Description Logics is similar to RDF, but more expressive.

ALC is a simple version of Description Logics.

There are more expressive DL variations too.

6. Description Logics (DL)

ALC Syntax: ALC describes a domain using these 3 items.

- Concepts (aka classes) → unary predicate constants in FOL
- Roles (aka relations or properties) → binary predicate constants in FOL
- Individuals (aka instances) → object constants in FOL

Terminological Axioms: We can define new rules/concepts/relations in our Ontology using existing concepts. For instance: Woman = Person Π Female

Concepts are defined recursively.

i-) Primitive concepts: occur only on the right side of definitions, which are simple and fundamental concepts (set of individuals in a class).

- For example “Person” is primitive, not derived from others.

ii-) Defined concepts: occur on the left side of definitions, which are complex concepts(union/intersection of two classes, etc.)

- For example “Woman” is defined, because derived from other concepts.

6. Description Logics (DL)

ALC Syntax: Necessary & Sufficient Conditions:

- C ⊑ D means “C subsumed by D” or like “C subset of D”
- C is a sufficient condition for D
- D is a necessary condition for C
- If C ≡ D then C is a sufficient & necessary condition for D, and vice versa.

Assertions: We can make assertions on individuals, using concepts or roles:

- Concept assertions: C(a) where C is a concept and a is an individual. For example: Student (John)
- Role assertions: R(a,b) where R is a role and a,b are individuals. For example: enrolled (John, CSE415)

In DL, a knowledge base K is a pair (T, A) which contains Tbox and Abox.

- Tbox is a set of terminological axioms, concept definitions, general relation between concepts/classes. No individuals/instances.
- Abox is a set of concept & role assertions. These are individual or instance level assertions.

6. Description Logics (DL)

ALC Semantics:

An interpretation I is a pair which consists

- A non-empty set (the universe or domain of the interpretation)
- A function for mapping (individuals, roles, concepts) from axioms to universe

In other languages T → True and ⊥ → False. In Description Logics:

- T(top) → set of all individuals in the universe, whole universe
- ⊥(bottom) → empty set

6. Description Logics (DL)

ALC Semantics:

TBox semantics apply to classes/concepts.

ABox semantics apply to individuals/instances.

A knowledge base K=(T,A) includes both ABox A and TBox T.

Model:

- An interpretation I is a model for TBox T, if it satisfies all statements in T.
- An interpretation I is a model for ABox A, if it satisfies all statements in A.
- An interpretation I is a model for KB K=(T, A), if it satisfies both T and A.

Satisfiability:

- An ABox A is satisfiable if it has at least one model.
- A TBox T is satisfiable if it has at least one model.
- A knowledge base K=(T, A) is satisfiable if it has at least one model.

Entailment: We can infer many things from Knowledge Base. We don't need to write every rule or statement in the Knowledge Base.

- Entailment help us to infer new statements from KB.
- Entailment means, if Knowledge Base is true then this new conclusion must be true too. (not optional, such as it can be or not.) see Example1-revisited.

Validity: A terminological axiom is valid, if every interpretation is a model of it.

6. Description Logics (DL)

DL to FOL: DL is an Ontology Language and actually it is a limited version of FOL or subset of FPL. So we can compute DL satisfiability easily.

We need to convert DL to FOL, so we can make reasoning.

- There are general rules to translate DL into FOL. (see slides)

If there is an individual in the statement, it is Abox, else it is Tbox. There are two extra translating rules for Tbox and Abox:

- For subsumed in Tbox
- For elementOf in Abox

6. Description Logics (DL)

Beyond ALC (simple DL): We can make DL more expressive using some additional constructs. While keeping its computing Satisfiability decidable.

- Number Restrictions: Adding numeric constraints on the number of roles.
- Qualified Number Restrictions: Also adding qualification to the numerics constraints.
- Enumerations – Nominals: Nominal is a concept that contains exactly one individual. Whereas Enumeration can contain many individuals.

UNA (Unique Name Assumption): If we have nominals and we don't use UNA, then we need to explicitly state whether two individuals are same or different. See Example in slides, why we cannot entail Mary as BusyTeacher.

K BusyTeacher(MARY) ?

OWA vs CWA:

- Closed World Assumption: Used in Answer-Set Theory. If it is not in the interpretation, we assume it is false.
- Open World Assumption: Used in Description Logics. If it is not in the interpretation, we assume it can be false or true. See the Example in slides, why we cannot entail Mary as ConsciousTeacher.

K ConsciousTeacher(MARY) ?

7. Reasoning Problems

We just reviewed Syntax and Semantics. How can we apply reasoning using a Knowledge Base and DL?

- Concept Satisfiability: Checking whether a concept C is satisfiable in knowledge base K. Or whether there exists a model I of K.
- Subsumption: Checking whether C ⊑ D with respect knowledge base K.
- Taxonomy & Classification: Given a new concept, find the right place in the taxonomy. It is also related with subsumption.
- Satisfiability: Checking whether knowledge base KB is satisfiable, whether it has at least one model that satisfies all statements in KB.
- Instance Checking: Checking whether instance/individual “a” belongs to class/concept C, e.g. C(a).
- Answering Concept Queries: Finding all instances of class C.

8. Web Ontology Language (OWL)

OWL is another Ontology Language:

- OWL is very much related to DL.
- OWL is recommended by W3C, to make it standard.
- OWL has 3 different variants (Lite, DL, Full). We use the OWL-DL version in this course.

OWL Syntax: There are three different styles of syntax (Functional, RDF, Manchester). But since we use Protege, we don't need to write it manually. But in some cases, you may need to change it manually (if not supported by GUI).

OWL Semantics: There are two different methods of defining OWL semantics: RDF-based and DL-based. We use DL-based semantics SROIQ in this course.

See the slides for correspondence of Terminologies and Syntax in these two languages: DL versus OWL.

Thanks

&

Questions