Ontology-oriented programming for biomedical informatics

Jean-Baptiste Lamy

Jean-baptiste.lamy @ univ-paris13.fr

LIMICS, Université Paris 13, Sorbonne Paris Cité, 93017 Bobigny, France, INSERM UMRS 1142, UPMC Université Paris 6, Sorbonne Universités
The biomedical domain is one of the most complex domains. A response has been the use of ontologies. Ontologies structure knowledge and enable inferences with a reasoner.

**Ontology**

- **Universal statements**
  - knowledge:
    - Ticagrelor is contraindicated with Hemorrhagic disorders

- **Terminological statements**
  - translations:
    - EN = Hemorrhagic disorders
    - FR = Maladies hémorragiques

- **Assertional statements**
  - data:
    - Ticagrelor has for daily cost 2.48€
Many tools have been developed for working with ontologies:

- Ontology language: OWL
- Ontology editor: Protégé
- Ontology API: OWLAPI

However, these tools do not integrate well with usual tools:

- Object-oriented programming
- Database...
Java source code for computing the total cost of a drug order:
(considering a single box for each drug)

```java
class Order {
    ...

    public float getCost() {
        float cost = 0.0;
        Iterator drug_iterator = this.getDrugs().iterator();
        while(drug_iterator.hasNext()) {
            Drug drug = (Drug) drug_iterator.next();
            cost = cost + drug.getPrice();
        }
        return cost;
    }
}
```
Same example using an ontology:

```java
public static float getOrderCost(OWLIndividual order) {
    OWLModel model = order.getOWLModel();
    OWLProperty drugProperty = model.getOWLObjectProperty("drug");
    OWLProperty priceProperty = model.getOWLDatatypeProperty("price");

    float cost = 0.0;
    Iterator drugs = order.listPropertyValues(drugProperty);

    while(purchases.hasNext()) {
        OWLIndividual drug = (OWLIndividual) drugs.next();
        Float price = (Float) drug.getPropertyValue(priceProperty);

        cost = cost + price.floatValue();
    }
    return cost;
}
```

Not really object-oriented

 распространенными в языковой модели ontology есть объективно-ориентированные сущности, которые не являются объектами в Java.
Working with ontologies: example

The VIIIP project (integrated visualisation of information about new drugs)

- Objective: to extract, enrich and visualize information about new drugs
- How to perform all three steps in an object-oriented way?

1) populate the ontology
2) produce inferences with a reasoner
3) generate a website
Tools for manipulating ontologies in programming languages are known as “ontology programming interface”

A problem identified by A. Rector in 2008

*Use Cases for Building OWL Ontologies as Modules: Localizing, Ontology and Programming Interfaces & Extensions*

How to access, manipulate and populate an ontology from a computer programming language?

* E.g. how to write a program for populating the ontology from a database and then for generating an HTML website from the inferences produced by the ontology?
Difficulties encountered with ontologies in the biomedical domain

Many medical concepts cannot be represented by *individuals* but need *classes* to represent them

- Disorders and drugs can be described at various levels of granularity
  - NSAID (Non Steroid Anti-Inflammatory Drugs)
  - Aspirin
  - Aspirin 100mg tablet in the antiplatelet indication

=> **Disorders and drugs must be classes in ontologies**

- Disorders and drugs properties are often defined at mid-granularity level
  - E.g. aspirin is contraindicated with hemorrhagic disorders

**But OWL does not allow direct relation between two classes**

- All kinds of aspirin are contraindicated with all kind of hemorrhagic disorders
- $\forall a, \forall d, \text{Aspirin}(a) \cap \text{HemorrhagicDisorder}(d) \Rightarrow \text{contraindication}(a, d)$
Difficulties encountered with ontologies in the biomedical domain

The open-world assumption is not always appropriate for medical reasoning

- Open-world assumption is interesting when reasoning on patient
  - It allows the reasoner to perform hypothesis about unknown patient conditions
- But it is not desirable on drug or disorder knowledge
  - Physicians usually consider that all properties of a given drug are known and described in official reference documents
  - This can lead to stupid alerts:
    - “the patient has renal failure; the drug you are prescribing is not contraindicated with renal failure but the drug might have a yet-unknown contraindication with renal failure.”

=> Even more difficult to integrate ontologies in computer program
Two approaches for ontology programming interface

- **Traditional API (e.g. OWLAPI in Java)**
  - Classes and functions for manipulating OWL constructs
  - An OWL class is an instance of OWLClass

- **Ontology-oriented programming**
  - Many similarities between ontologies and the object-oriented programming paradigm
    - Classes, properties and individuals in ontologies
    - = Classes, attributes and instances in object models

  - Try to unify the ontological model with the object model of the programming language
    - An OWL class is a class in the programming language
  - => shorter source code, faster development and thus fewer errors
## Object and ontology paradigm

<table>
<thead>
<tr>
<th><strong>Object</strong></th>
<th><strong>Ontology</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Closed-world assumption:</strong> Implicit disjoints between classes: two classes are disjoint if they have no common child</td>
<td><strong>Open-world assumption:</strong> Explicit disjoints between classes: two classes can be disjoint or not</td>
</tr>
<tr>
<td>Implicit disjoints between instances: all instances are distinct</td>
<td>Explicit disjoints between instances: two instances can be distinct or not</td>
</tr>
<tr>
<td>An instance belongs to a single class</td>
<td>An instance can belong to several classes</td>
</tr>
<tr>
<td>A property is defined in a given class</td>
<td>Properties are first-class entities, independent from classes</td>
</tr>
</tbody>
</table>
Static vs dynamic

Static ontology-oriented programming:
- Automatic generation of classes source code from an OWL ontology
  - Java [Kalyanpur], Sapphire [Stevenson], C# [Goldman]

Dynamic ontology-oriented programming:
- A dynamic translator between the ontology paradigm and the object paradigm
- Limited to dynamic programming languages
  - Common Lisp [Koide], Python [Babik]
- Allows automatic classification at run time
- Mix Open-world (ontology) and closed-world (object) assumption

=> Dynamic ontology-oriented programming seems the best way for accessing ontology in biomedical informatics
OwlReady

A module for dynamic ontology-oriented programming in Python 3

- https://pypi.python.org/pypi/Owlready (Free Software, GNU LGPL)
  - Dynamic class creation
  - Automatic classification of classes and instances with the Hermit reasoner
  - Addition of Python methods to OWL classes
  - OWL/XML 2 file format

Diagram:

- OwlReady
  - Metaclasse
  - Classes
  - Properties
  - Individuals
- OWL file
  - Classes
  - Properties
  - Individuals
- Hermit reasoner
- OWL/XML parser
- .py file (Python source code)
public static float getOrderCost(OWLIndividual order) {
    OWLModel model = order.getOWLModel();
    OWLProperty drugProperty = model.getOWLObjectProperty("drug");
    OWLProperty priceProperty = model.getOWLDatatypeProperty("price");

    float cost = 0.0;
    Iterator drugs = order.listPropertyValues(drugProperty);

    while(purchases.hasNext()) {
        OWLIndividual drug = (OWLIndividual) drugs.next();
        Float price = (Float) drug.getPropertyValue(priceProperty);

        cost = cost + price.floatValue();
    }
    return cost;
}

class Order(Thing):
    def get_cost(order):
        cost = 0.0
        for drug in order.drugs:
            cost = cost + drug.price
        return cost

A Python / OWL class with a Python method.
Using OwlReady in VIIIP

Architecture of the VIIIP platform with OwlReady

- Include an ontology of drug contraindications

Diagram:
- Drug database
- Clinical trial reports
- SQL request
- Manual data entry
- HermiT Reasoner
- VIIIP Ontology (OWL)
- VIIIP Ontology (Python)
- Website
- Dialog boxes
Automatic dialog boxes generation from the ontology

- OwlReady can generate dialog boxes for editing instances
  - Using the EditObj3 dialog box generator
  - And the definition in the ontology
## Comparison with literature

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>static</td>
<td>static</td>
<td>dynamic</td>
<td>dynamic</td>
<td>semi-dynamic</td>
<td>dynamic</td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td>C#</td>
<td>Java</td>
<td>Common Lisp</td>
<td>Python</td>
<td>Java</td>
<td>Python</td>
</tr>
<tr>
<td><strong>Classification of classes</strong></td>
<td>no</td>
<td>no</td>
<td>?</td>
<td>?</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Classification of instances</strong></td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>?</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Syntax for OWL definitions</strong></td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>?</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Mixed classes with methods</strong></td>
<td>no</td>
<td>no</td>
<td>?</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
Discussion

Limitations

- OwlReady works well for reasoning purpose
- But it is not yet good at loading big ontologies with many imports
  - No support for RDF
- Performance are low, but the time lost is negligible compared to the time taken by the reasoner

Future works: relations on classes

- Export some restrictions of a class as class attributes
  - OWL: Class property_x value 1  <=>  Python: Class.x = 1
- Add support for class-class relations
  - i.e. relation between all instances of a class and all instances of another
Dynamic ontology-oriented programming is an interesting approach for integrating ontologies in computer software

- Simpler and shorter source code
  - While computer programs manipulating biomedical ontologies tend to be more and more complex
  - Especially in the biomedical domain
    - because disorders and drugs must be represented using classes
    - because a mix of open- and closed-world assumption is often needed


