# Philip Hale

## CS3019 Assessment - 25% - due 07/12/2012

**Your report should include:**

- Description of how you extended the provided ontology in different steps.
- Description of how you extended the provided rule set.
- Functionalities of your rule sets. Please provide step by step instructions and detailed examples to illustrate the functionalities of your rule sets.
- Your findings during the assessment.

---

I completed the assessment using a combination of an XML text editor and Protégé. I found it helpful to make the initial extensions to the ontology by directly editing the OWL document, since this gave me a full understanding of how Protégé is operating the data behind its interface. As the tasks become more complicated and required the use of JESS, I switched to using Protégé fully.

Despite editing the ontology directly to complete the tasks, I was sure to check the resulting ontology functioned correctly when imported into Protégé. In addition, my text editor provided XML linting and checked for ontology consistency after each save.

The following is an explanation of what was done at each stage to successfully complete the set tasks. The work is broken down by CAS mark and the tasks within each bracket.

---

## Between CAS 9 and CAS 11

**Classify the ontology using an ontology reasoner**

The Pellet reasoner's CLI shows the following output:

```
$ java -jar pellet-2.3.0/lib/pellet-cli.jar classify bookstore.owl
Classifying 8 elements
Classifying:  100% complete in 00:00
Classifying finished in 00:00

  owl:Thing
     bookstore:Book
     bookstore:Person
        bookstore:Author
        bookstore:Customer
     bookstore:Purchase
     bookstore:Recommendation
```

**Edit these two classes, making them direct subclasses of Person**

This can be achieved by explicitly specifying that Customer and Author are subclasses of Person:

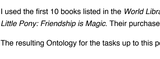
```
<owl:Class rdf:ID="Customer">
    <rdfs:subClassOf rdf:resource="#Person"/>
    [...]
</owl:Class>
<owl:Class rdf:ID="Author">
    <rdfs:subClassOf rdf:resource="#Person"/>
    [...]
</owl:Class>
```

We can use the HermiT reasoner to check this was successful:

```
$ java -jar HermiT/HermiT.jar -ds :Person bookstore.owl
Direct sub-classes of ':Person':
    :Author
    :Customer
```

**Change the asserted type of the 4 existing instances of Person appropriately to be either Author or Customer.**

A screenshot from Protégé shows that all instances of Person are now either Authors or Customers:




Protégé Class Browser

**Add some more books, authors and customers to allow for testing later**

I used the first 10 books listed in the *World Library*. The names of the customers are taken from the children's TV show *My Little Pony: Friendship is Magic*. Their purchases are imagined.

The resulting Ontology for the tasks up to this point can be found in the file `bookstore_cas11.owl`.

## Between CAS 12 and CAS 14

**Manually add recommendations into the ontology based on the newly created instances.**

The following recommendation was added, based on the *Moby Dick* example.

Note that the number of purchases is different in this ontology from that in the previous section. This is in order to comply with the rule that every possible recommendation should be listed.

```
  <Recommendation rdf:ID="recommendation_1_1">
    <hasBuyer rdf:resource="#twilight_sparkle"/>
    <hasBook rdf:resource="#the_divine_comedy"/>
  </Recommendation>
```

**Extend the ontology by adding a BookSubject class, and build up a small subject hierarchy beneath it.**

Given the nature of the books listed, it didn't make sense to split the books in modern categories such as Thriller, Romance etc. Instead, I created a subject hierarchy which was designed to support the rest of the *World Library* collection, were they to be added at some later point in time.

```
$ java -jar pellet-2.3.0/lib/pellet-cli.jar classify bookstore.owl
Classifying 31 elements
Classifying:  100% complete in 00:00
Classifying finished in 00:00

  owl:Thing
     bookstore:BookSubject
        bookstore:English
        bookstore:Fiction
           bookstore:Novel
              bookstore:NovelC19
              bookstore:NovelC20
              bookstore:NovelC21
           bookstore:Play
           bookstore:Poetry
        bookstore:NonEnglish
           bookstore:LanguageArabic
           bookstore:LanguageFrench
           bookstore:LanguageGerman
           bookstore:LanguageGreek
           bookstore:LanguageHebrew
           bookstore:LanguageLatin
           bookstore:LanguageRussian
        bookstore:NonFiction
           bookstore:CategoryBusiness
           bookstore:CategoryCalendars
           bookstore:CategoryComputing
           bookstore:CategoryLanguages
           bookstore:CategoryTravel
     bookstore:Book
     bookstore:Person
        bookstore:Author
        bookstore:Customer
     bookstore:Purchase
     bookstore:Recommendation
```

**Create a hasSubject property and use it to link the books with the subject individuals.**

A new ObjectProperty links books with subjects:

```
  <!--BOOKS HAVE SUBJECTS-->
  <owl:ObjectProperty rdf:ID="hasSubject">
    <rdfs:domain rdf:resource="#Book"/>
    <rdfs:range rdf:resource="#BookSubject"/>
  </owl:ObjectProperty>
```

You can see in Protégé that books have been linked with subjects by way of the mirror hierarchy of subject instances.

The resulting ontology for this section can be found in the file `bookstore_cas14.owl`

## Between CAS 15 and CAS 17

**Enable Jesstab and run bookstore-rules.jess**

The `bookstore_rules.txt` file was renamed `bookstore_rules.jess` and edited to work correctly with the Ontology in Protégé. This meant prefixing references to objects with the full pathname (URL) of the OWL classes like so:

```
(maptclass http://www.cs4021/bookstore.owl#Customer)

(maptclass http://www.cs4021/bookstore.owl#Book)

(maptclass http://www.cs4021/bookstore.owl#Purchase)

(maptclass http://www.cs4021/bookstore.owl#Recommendation)

(defrule recommend-1
   [...]
)
```

Importing these rules and running them in JessTab resulted in the following recommendation being given:

```
Jess> (batch "/Users/philiphale/code/kbs-assessment/bookstore_rules.jess")
TRUE
Jess> (run)
Recommendation for Twilight Sparkle
    : other people that bought "Njal's Saga
    " also bought "The Divine Comedy
    "

Recommendation for Jeff: other people that bought "Moby Dick" also bought "Fly Fishing"
2
```

Protégé

Note that although the formatting is slightly off, the recommendation given is the same as the one added manually in the previous section.

**Create a new Jess rule that recommends books for a customer based on his or her previous purchases.**

*If the customer has bought a book by a particular author, add recommendations for books written by the same author.*

Following the pattern of the example rule, this rule was created by gradually honing in on the correct objects. I imported the rule into Protégé and ran it each time I made a change to the rule, in order to spot any errors with the code as quickly as possible. Here we can see the recommendations, which are generated based on the 'Anonymous' author.

```
Jess> (batch "/Users/philiphale/code/kbs-assessment/bookstore_rules.jess")
TRUE
Jess> (run)
Recommendation for Twilight Sparkle: other people that bought "Njal's Saga" also bought "The Di
Recommendation for Jeff: other people that bought "Moby Dick" also bought "Fly Fishing"
Recommendation for Fluttershy: the author of: "Njal's Saga has also written: "On Thousand and O
Recommendation for Fluttershy: the author of: "Njal's Saga has also written: "Book of Job"
Recommendation for Fluttershy: the author of: "Njal's Saga has also written: "Epic of Gilgamesh
Recommendation for Twilight Sparkle: the author of: "Njal's Saga has also written: "On Thousand
Recommendation for Twilight Sparkle: the author of: "Njal's Saga has also written: "Book of Job
Recommendation for Twilight Sparkle: the author of: "Njal's Saga has also written: "Epic of Gil
Recommendation for Spike: the author of: "On Thousand and One Nights has also written: "Njal's
Recommendation for Spike: the author of: "On Thousand and One Nights has also written: "Book of
Recommendation for Spike: the author of: "On Thousand and One Nights has also written: "Epic of
Recommendation for Rainbow Dash: the author of: "Book of Job has also written: "Njal's Saga"
Recommendation for Rainbow Dash: the author of: "Book of Job has also written: "On Thousand and
Recommendation for Rainbow Dash: the author of: "Book of Job has also written: "Epic of Gilgame
14
```

**Create a new Jess rule that recommends books for a customer based on his or her previous purchases.**

*If the customer has bought a book on a particular subject, add recommendations for books of the same subject (and subclasses of that subject).*

## CAS 18 - 20

**Extend the explanation functionality**

**Extend the first recommendation rule**