

Test 4a - Ontologies

Started: 2 Oct at 12:57

Quiz instructions

PLEASE READ CAREFULLY: this Test is different from the previous tests, and closely related to the [assignments 4b](https://canvas.vu.nl/courses/24675/assignments/2914) (<https://canvas.vu.nl/courses/24675/assignments/2914>) and [assignment 4c](https://canvas.vu.nl/courses/24675/assignments/2915) (<https://canvas.vu.nl/courses/24675/assignments/2915>). It will take the entire module to work on the test this time.

In this test you will build your own OWL ontology in your local installation of Protege and experiment with reasoning. While answering the questions in the test, you will create an ontology, which you will have to hand in as assignment 4b, and extend it, so that it now shows some extra OWL inferences for assignment 4c. This test is about the motivation of your modelling.

You are free to choose the domain (subject) of the ontology you are going to build (e.g. on nutritional value, recipes, supermarkets, food safety, health, restaurants, planes, trains and automobiles, developing countries, modern slavery, political parties, refugees, you name it...)

Please be sure to practice a bit with Protege, and look at the common mistakes slides in the lecture handout.

You can submit the test **only once**. Be absolutely sure not to submit before you're really done.

Make sure to **save your answers**. You can then leave the test, and come back to it whenever you want. Just **make sure not to submit before you are ready!**

You are allowed to help each other out with the technical details, but make sure that you provide **your own answer** to any 'open questions'.

The ontology from the Pizza tutorial is a good example: <http://protege.stanford.edu/ontologies/pizza/pizza.owl> (<http://protege.stanford.edu/ontologies/pizza/pizza.owl>), but please be creative (toppings on fries instead of pizza's is not creative). The [tutorial itself is worth](http://owl.cs.manchester.ac.uk/publications/talks-and-tutorials/protg-owl-tutorial/) (<http://owl.cs.manchester.ac.uk/publications/talks-and-tutorials/protg-owl-tutorial/>) reading. It guides you through building an ontology step by step, as does the nice screencast by Rinke that is provided in Module 4.

Remember that it would be quite smart to be able to reuse this ontology for the final project.

Question 1

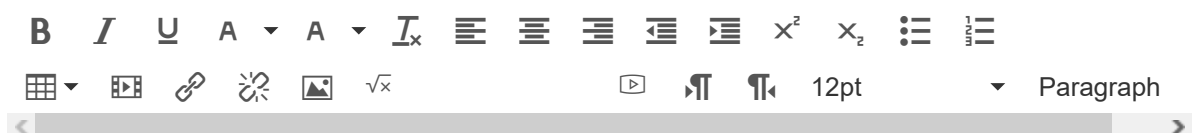
1 pts

First, you will have to create a new ontology in Protege.

Be sure to choose your own unique **ontology URI**, and specify a **namespace** and a **prefix**

Give the ontology URI and namespace + prefix

[HTML Editor](#)



@prefix : <http://clockman.com/ontologies/scientific-research> .

(The default prefix ":" is assigned to the URI in order to make it easier to work with it. This is accomplished in Protege by simply leaving the 'prefix' cell empty, and filling only the URI. It is also confirmed that this approach works by checking the .ttl file of the project in a text editor.)

p

Question 2

3 pts

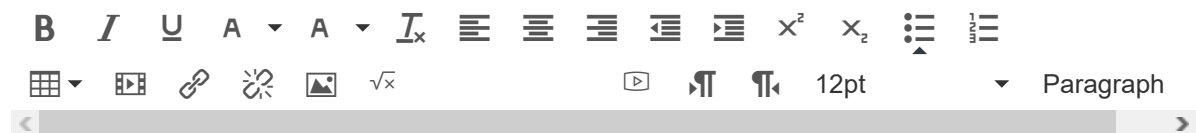
Create **six** classes, **four** properties, and an example instance for **each** class in the ontology you created before.

Try not to assert class membership, but do use the properties to relate the instances to each other.

Run the reasoner

Which classes did you create and which instances?

[HTML Editor](#)



- The following **classes** were created
 - :Document
 - *instance: Semantic_Web_Primer_3rd_edition*
 - :ConferenceProceedings
 - *instance: International_Joint_Conference_on_Artificial_Intelligence*
 - :Person
 - *instance: Can_John_Lokman*
 - :Project
 - *instance: Knowledge_Flows_In_Interdisciplinary_Research*
 - :ResearchPaper
 - *instance: Non-Standard_Reasoning_Services_for_the_Debugging_of_Description_Logics_Terminologies*
 - :Researcher
 - *instance: Stefan_Schlobach*

- Also, the following object **properties** were created in order to relate the instances to each other:
 - :hasTopic
 - :hasCreator
 - :hasProjectLeader
 - :hasAuthor
 - :hasCoAuthor
 - :hasEditor
 - :hasWorkedOn
 - :leadsProject
 - :isAuthorOf
 - :isCoAuthorOf
 - :isEditorOf
 - :isPublishedOn
 - :hasFeatured
 - :publishesInTheFieldOf
 - :hasFieldOfResearch

ul » li » ul » li



Question 3

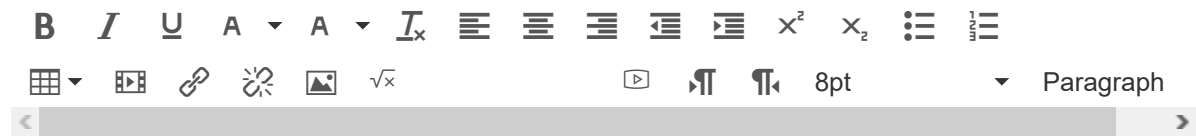
2 pts

Create necessary and sufficient conditions on **three** classes. For example, an Hawaiian pizza always has pineapple topping, a vegetarian dish never contains meat and/or fish, narcissists always like themselves, etc.

Run the reasoner

Question: for which classes did you define the conditions? What were these conditions, and why?

[HTML Editor](#)



Subclass relationships were defined for numerous classes. The two most relevant ones of these (as they play a role during reasoning) are as following:

Class of journal articles:



Description: :JournalArticle

Equivalent To +

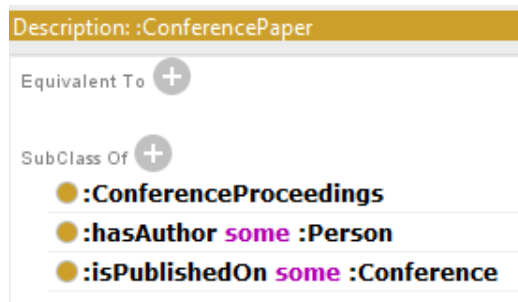
SubClass Of +

- :hasAuthor some :Author
- :isPublishedOn some :Journal
- :Journal

(<https://i.imgur.com/EkNDdWX.png>)

The superclass assignments above were made as a Journal article could be described as something that has an author, and is published on journal.

Class of conference papers:



Description: :ConferencePaper

Equivalent To +

SubClass Of +

- :ConferenceProceedings
- :hasAuthor some :Person
- :isPublishedOn some :Conference

(<https://i.imgur.com/EmakEXc.png>)

A similar reasoning was followed assigning superclasses to the conference papers.

To retrieve a superclass of all published papers, and also for retrieving a superclass of all publications, the following necessity + sufficiency definitions were entered

Class of all publications:



Description: :Publications

Equivalent To +

- :featuresArticle some (:ConferencePaper or :JournalArticle)

SubClass Of +

- owl:Thing

<https://i.imgur.com/CR6gA6Q.png>

Class of all published papers:



Description: :PublishedPapers

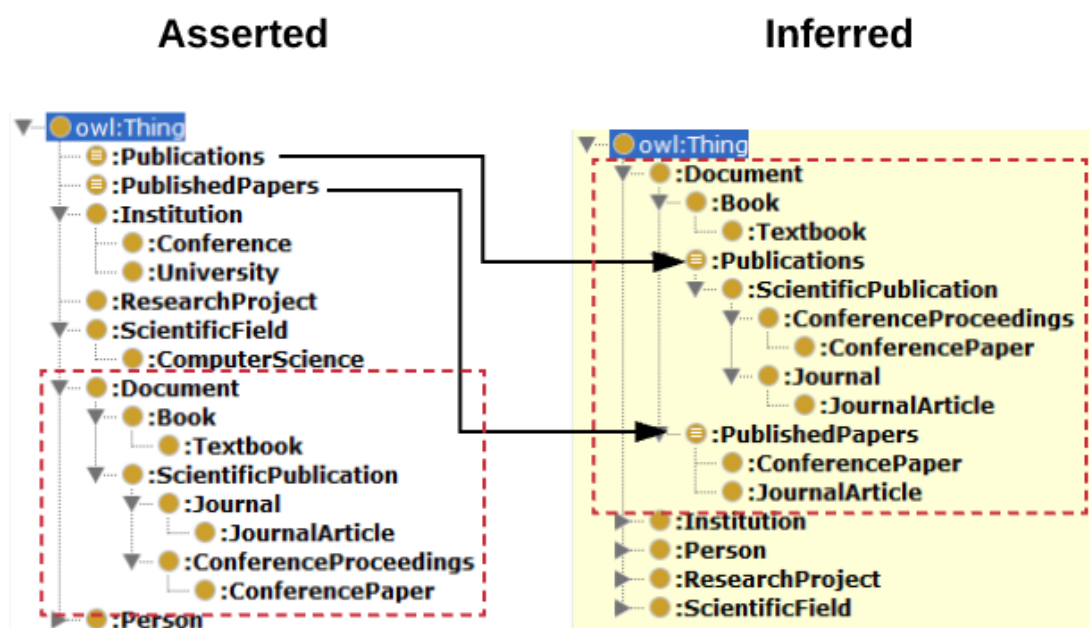
Equivalent To +

- :isPublishedOn some (:Conference or :Journal)

SubClass Of  owl:Thing

(<https://i.imgur.com/WTUBfsa.png>)

After these descriptions (i.e., subclass relationships) and definitions (i.e., equivalentOf relationships), the reasoner returned the following results:



(<https://i.imgur.com/LqqoyQI.png>)

p » span

Question 4

3 pts

Define at least **three** more, different class restriction types (someValuesFrom, allValuesFrom, hasValue and cardinality, local reflexivity) for the classes from your ontology. They need not be necessary & sufficient, but can be merely necessary as well.

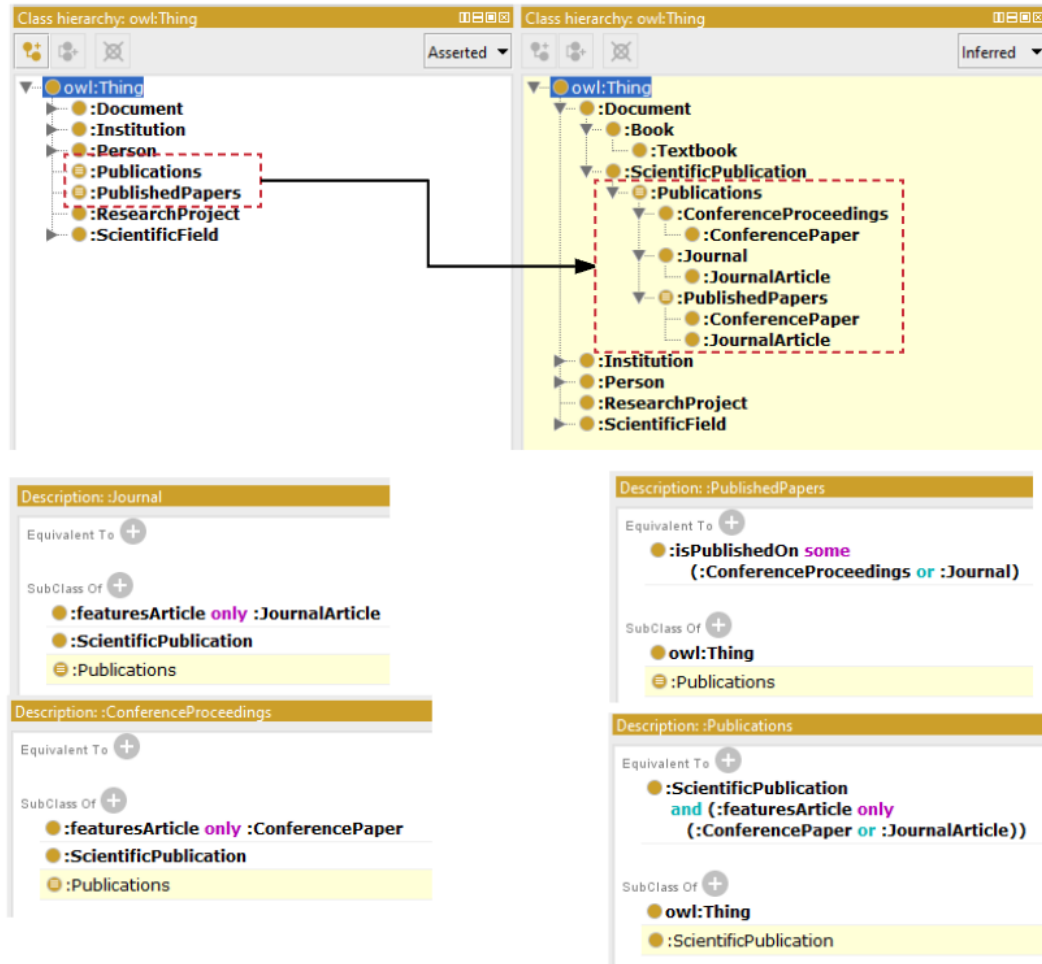
Run the reasoner.

Which restrictions did you define, and for which classes? And why?

[HTML Editor](#)

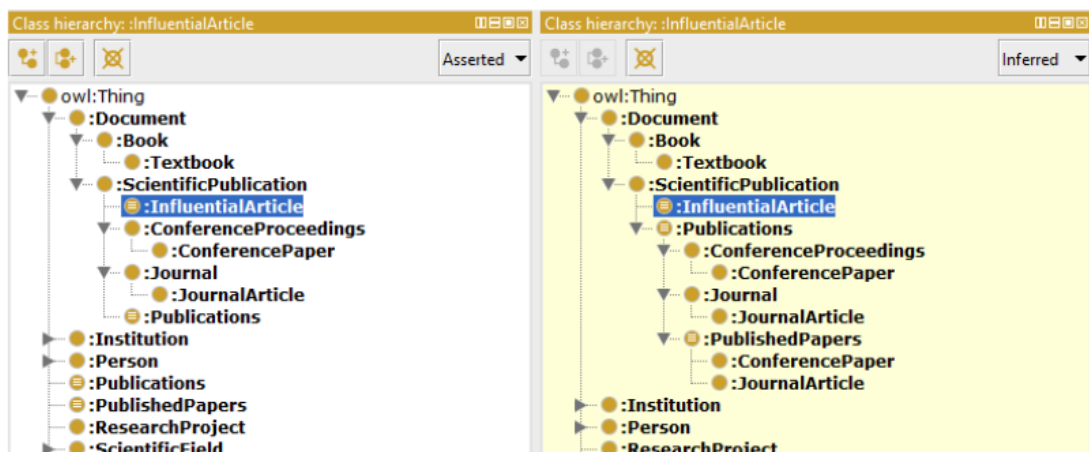
Rich text editor toolbar with icons for bold, italic, underline, text color, background color, bulleted list, numbered list, link, unlink, image, source code, undo, redo, and font size (12pt). The text 'Paragraph' is visible on the right.

1. 'someValuesFrom' was already present, so '**allValuesFrom**' construct is tried by converting most someValuesFrom statements (i.e., subclassOf) to allValuesFrom (i.e., equivalentOf) in 3 places where this made sense. Much like it is in the real-world, Journals now **only** allowed to publish JournalArticles, and conferenceProceedings were now only allowed to publish conferencePapers (before, these relationships were defined with 'some', instead of 'only').. Notably, this harsher class restriction did not have a negative effect on the inferences, and in fact, probably made the ontology more stable. This state of the ontology can be viewed below:



[<https://i.imgur.com/ler6H7q.png>]

2. Second, influential articles were classified by using **magnitude restrictions** through hasCitationCount property:



Instances Individuals

Instances: For: **:InfluentialArticle**

Instances (inferred) Property assertions

Instances (inferred):

- :Non-Standard_Reasoning_Services_for_the_Debugging**
- :Ontology_Learning_for_the_Semantic_Web**

Description: **:InfluentialArticle**

Equivalent To +

- :ScientificPublication**
- and (**:hasCitationCount** some xsd:int[>= "1000"^^xsd:int])

SubClass Of +

- :ScientificPublication**

[<https://i.imgur.com/mS9pnpK.png>]

3. Third, cases where collaborations between authors occurred were classified using cardinality restrictions in the following manner:

Asserted Inferred

owl:Thing

- :Document**
 - :Book**
 - :Textbook**
 - :ScientificPublication**
 - :Collaborations**
 - :InfluentialArticle**
 - :ConferenceProceedings**
 - :ConferencePaper**
 - :Journal**
 - :JournalArticle**
 - :Publications**
 - :Institution**
 - :Person**
 - :Publications**
 - :PublishedPapers**
 - :ResearchProject**
 - :ScientificField**

owl:Thing

- :Document**
 - :Book**
 - :ScientificPublication**
 - :Collaborations**
 - :InfluentialArticle**
 - :Publications**
 - :Institution**
 - :Person**
 - :ResearchProject**
 - :ScientificField**

Instances (inferred):

- :Non-Standard_Reasoning_Services_for_the_Debugging**

Description: **:Collaborations**

Equivalent To +

- :ScientificPublication**
- and **:hasAuthor** min 2

SubClass Of +

- :ScientificPublication**

An individual that represents a scientific article

[<https://i.imgur.com/z9u6rvB.png>]

Property assertions: **:Non-Standard_Reasoning_Services_for_the_Debugging**














Object property assertions +

- :isPartOf** **:Non-Standard_Reasoning_Services_for_the_Debugging**
- :hasCreator** **:Stefan_Schlobach**
- :hasCreator** **:Ronald_Cornet**
- :hasAuthor** **:Stefan_Schlobach**
- :hasAuthor** **:Ronald_Cornet**

p

2 pts

Which properties did you select, what is the special feature, and why did you model it like this?

B *I* U A ▾ A ▾ I_x      x^2 x_2      \sqrt{x}    12pt ▾ Paragraph

'hasAuthor' property, which relates documents to their authors, is an asymmetric property due to the fact that it is a one directional relationship: An document may have an author, but an author may not have an author. Similarly, hasAuthor is also considered an irreflexive property, due to the fact that an a document cannot be its own author.

The screenshot displays the Protege ontology editor interface. On the left, the 'Object property hierarchy: owl:topObjectProperty' is shown as a tree. The property **foaf:knows** is highlighted with a red box. To the right, the 'Characteristics: :hasAuthor' panel shows the following checked options: **Asymmetric** and **Irreflexive**. Below this, the 'Description: :hasAuthor' panel shows the following relationships: **Equivalent To** (empty), **SubProperty Of** (**:hasCreator**), **Inverse Of** (**:isAuthorOf**), **Domains (intersection)** (**:Document**), **Ranges (intersection)** (**:Person**), **Disjoint With** (empty), and **SuperProperty Of (Chain)** (empty).

Below the hierarchy, the 'Characteristics: foaf:knows' panel shows the following checked options: **Symmetric**. The 'Description: foaf:knows' panel shows the following relationships: **Equivalent To** (empty), **SubProperty Of** (empty), **Inverse Of** (**foaf:knows**), **Domains (intersection)** (**:Person**), **Ranges (intersection)** (**:Person**), **Disjoint With** (empty), and **SuperProperty Of (Chain)** (empty).

[<https://i.imgur.com/xpHWMJn.png>]

Second (bottom left of the image above), foafKnows has chosen to be a Symmetric property due to the fact that if someone knows another person, this other person would also automatically know the first person. Initially, the 'Reflexive' option was also selected, and this is justified by the fact that a person can be claimed to 'know' himself, but although legitimate, this feature was not kept due to causing errors with the reasoner / destabilizing the ontology (as detected by Protege's

explanation module / debugger) multiple times.

p














Question 6

3 pts

IMPORTANT: Before you answer this question, first finish your work for assignments 4b and 4c!

Describe **what went well, and what went wrong**. What adjustments did you have to make to ensure that the reasoner inferred new knowledge?

[HTML Editor](#)

B *I* U **A** ▾ **A** ▾ *I_x*       x^2 x_2    \sqrt{x}     12pt ▾ Paragraph

It was overall a steep learning curve, as the concepts still do not fully sink in after one week of missing sleep over it. Neither OWL nor Protege is made to be intuitive, and in my opinion, wider adoption of this wonderful thing we are doing will require much better interfaces. Here is my progress and some things I learned:

- Although this happened a bit late due to hurrying for the tasks at hand, discovering how to customize my Protege tabs and workspace has been really helpful. During this assignment, the best setting was to put default and inferred panes side by side.
- It took me a good long few hours to be able to do my first inference. And it was surprising to see that this only happened after I carefully defined all features of all properties. Remembering the inference rules regarding to properties, I learned to check the properties when things seemed to be wrong with no obvious reason.
- I lost few hours on an inference error, which was due to a cardinality requirement that was not being met no matter what. I later realized, almost by accident, that two individuals were not being seen as different entities (thanks/because of the no unique names assumption and open world assumption). After differentiating these two individuals manually, the cardinality restriction worked like a charm, because now finally there were two individuals in the class I specified! After all this pain, I started to doubt whether open world assumption is simply laziness/lack of means to do a more clever reasoner/language; and whether no unique names assumption is simply bad design. Luckily, I found a "differentiate all individuals" option under Edit menu. Now I appreciate the importance of this operation.
- Another error a few hours ago than this one was caused by a similar issue: after making my top-level classes disjoint, I started to get a lot of errors. Although this was frustrating in the beginning, it turned out to be a very good way to root out the inconsistencies in the ontology. As the classes that should be perfectly distinct was not able to separate, I had to check properties, and indeed, realized that a name change, together with two very similar entities that belonged to different classes was the culprit: "Conference" as institute, and "Conference" (proceedings) as a publication. I will be extra careful around such similar entities in future, and will update the occurrences of one of them carefully if the second one is created much later (which was the case). Also, the "Disjoin all siblings" option in the Edit menu —once again— is a great way to check if everything is OK and for fortifying my ontology.
- In case of errors, asking for the explanation to the Protege was unexpectedly helpful. I also used the Protege debugger plugin for extra assistance.
- Overall, it was a good and tiring adventure with many moments of frustration. It is a shame that OWL and Protege is so user-hostile, as popularity of linked data and similar technologies could have been increasing much more rapidly if they

and similar technologies could have been increasing much more rapidly if they were more accessible. Perhaps that's something I can contribute towards as a

- ul » li

Saved at 6:45

Submit quiz