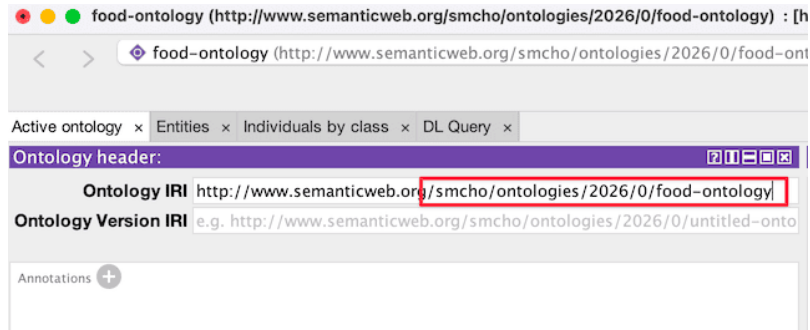


1. Protégé

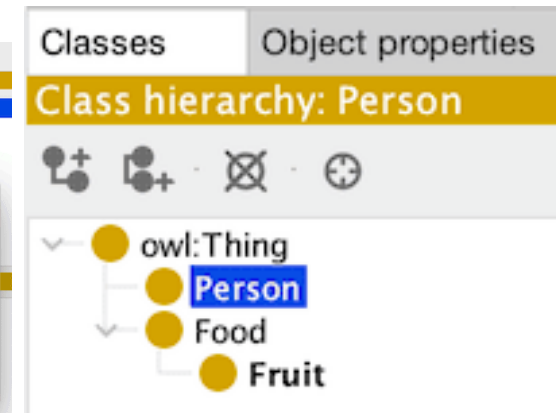
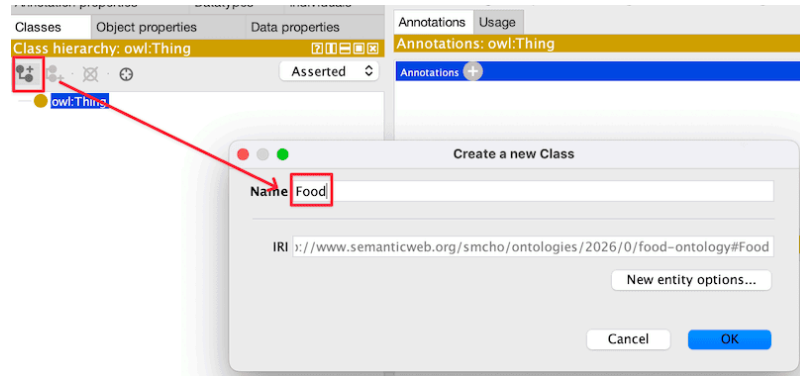
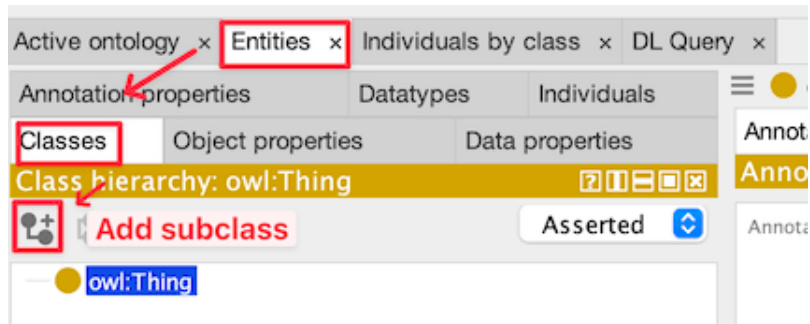
Create a new ontology project

File -> New



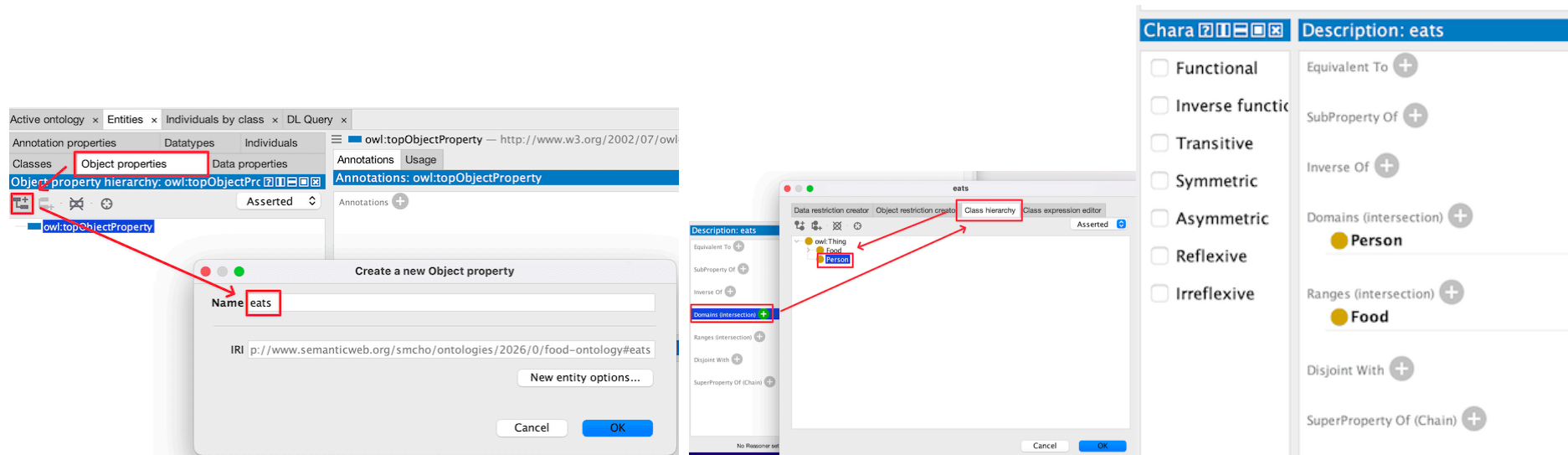
- Ontology project is created & file is loaded
- You are ready to start modeling
- Change the IRI (Identifier) of the ontology

Create Classes



Create Relationships (Properties)

Makes the relationship that: Person (Domain) eats Food (Range)



Up to this point, we have created the following ontology:

1. semantic rule: eats (Person \rightarrow Food)

- Automatic type inference
- Data validation
- Logical reasoning

2. If we assert: Alice eats Apple

Progege can infer that:

Alice is a Person
Apple is Food

Create Individuals (Instances)

The screenshot displays the Protégé ontology editor interface. At the top, the 'Active ontology' tab is active, showing the 'Class hierarchy: Person' view. The hierarchy includes 'owl:Thing' as the root, with 'Food' as a subclass, and 'Person' as a subclass of 'Food'. The 'Person' class is highlighted with a red box. A red arrow points from this box to a 'Create a new Named individual' dialog box. The dialog box has a 'Name' field containing 'Alice' and an 'IRI' field containing 'iri:./www.semanticweb.org/smcho/ontologies/2026/0/food-ontology#Alice'. The 'OK' button is highlighted. Another red arrow points from the 'Person' class in the hierarchy to the 'Direct instances: Apple' section at the bottom. This section shows 'Apple' as a direct instance of the 'Fruit' class. The 'Fruit' class hierarchy is also visible in the top right, showing 'owl:Thing' as the root, with 'Food' as a subclass, and 'Fruit' and 'Person' as subclasses of 'Food'. The 'Fruit' class is highlighted with a blue box. The 'Direct instances: Apple' section is also highlighted with a blue box.

Class hierarchy: Person

- owl:Thing
 - Food
 - Person

Class hierarchy: Fruit

- owl:Thing
 - Food
 - Fruit
 - Person

Create a new Named individual

Name: Alice

IRI: ./www.semanticweb.org/smcho/ontologies/2026/0/food-ontology#Alice

New entity options...

Cancel OK

Individuals Individuals (Inferred)

Direct instances: Apple

For: Fruit

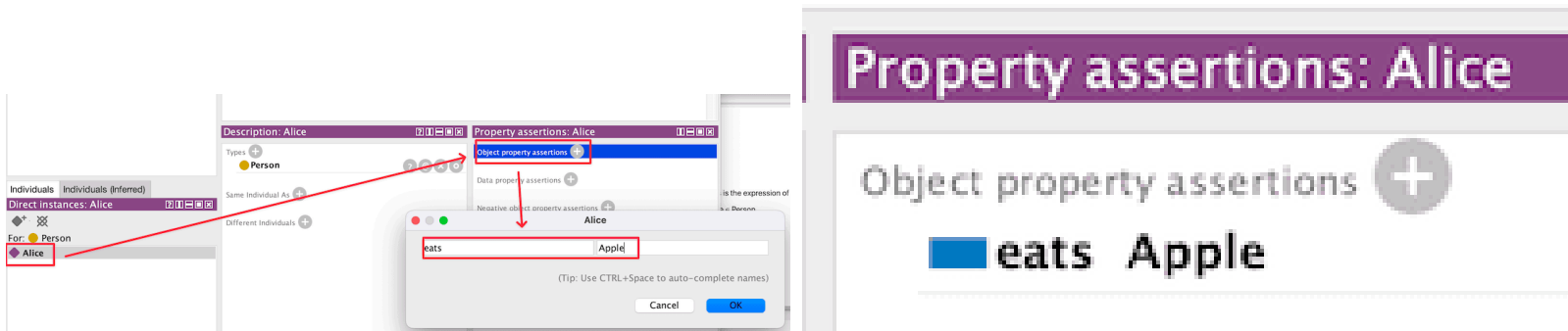
Apple

This is the expression of the individuals we just created:

Alice \in Person

Apple \in Fruit

Assert Relationships between Individuals



- Assert that Alice eats Apple
- Assert that Alice eats Apple

We just created a semantic triple:

Subject		Predicate		Object
Alice	–	eats	→	Apple

- Now we have:
 - $\text{Alice} \in \text{Person}$
 - $\text{Apple} \in \text{Food}$
 - $\text{eats}(\text{Alice}, \text{Apple})$

Run the Reasonor - Automatic Inference in Protege

Make sure this is checked at the bottom right corner:

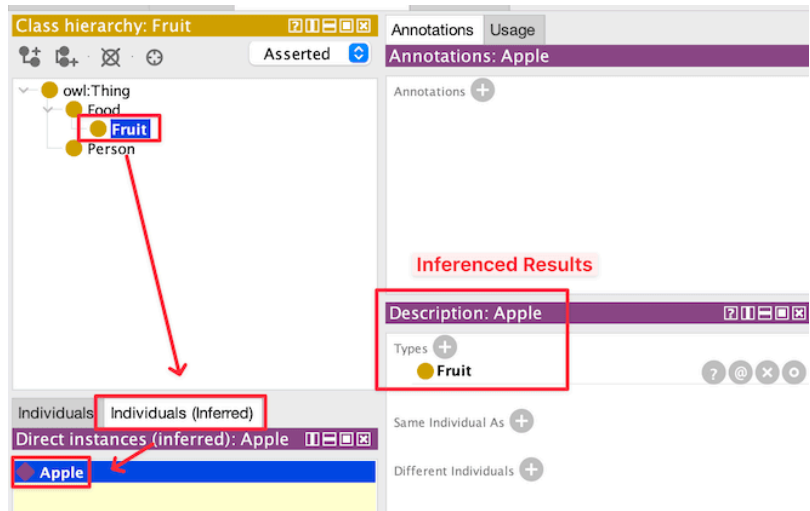


Show Inferences

- Click on the "Reasoner" menu
- Select "Start Reasoner"



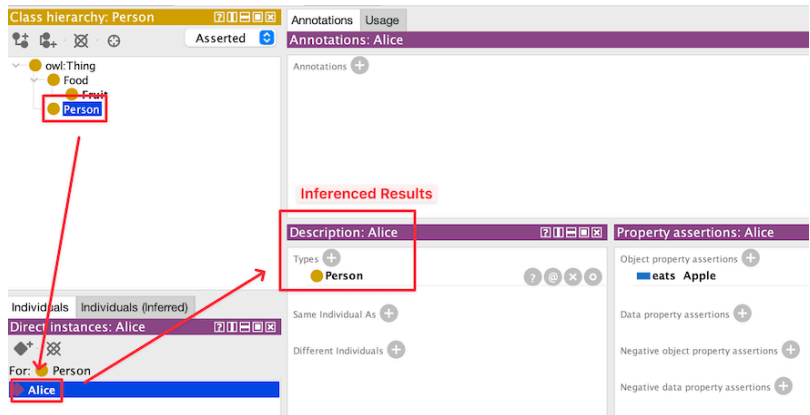
Verify Instance Inference



Even if you never manually assigned Food.

Protege infers that Apple is Food because Alice (a Person) eats Apple.

```
Fruit  $\subseteq$  Food  
Apple  $\in$  Fruit  
 $\Rightarrow$  Apple  $\in$  Food
```

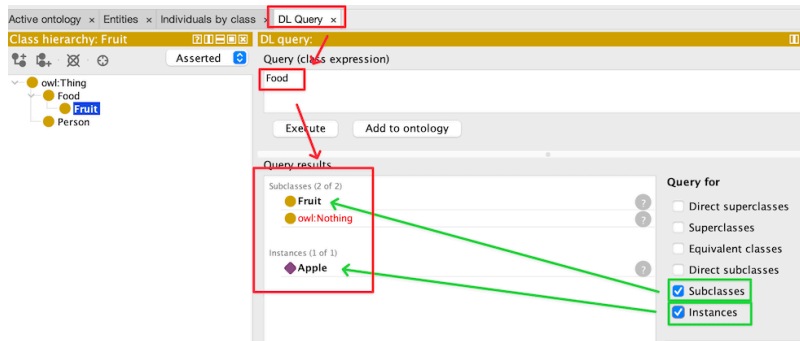


Even if you never manually assigned Person.

Protege infers that Alice is Person because Alice eats Apple (a Food).

```
Domain(eats) = Person  
Alice eats Apple  
⇒ Alice ∈ Person
```

DL Query (Semantic Querying)



Entering "Food" in the DL Query tab and clicking "Execute" will return all individuals inferred to be of type Food.

$\text{Fruit} \subseteq \text{Food}$
 $\Rightarrow \text{Apple} \in \text{Food}$

Query (class expression)

Food **and** (inverse eats **some** Person)

Execute

Add to ontology

Query results

Subclasses (1 of 1)

 owl:Nothing

Instances (1 of 1)

 Apple

Food and (inverse eats some Person)

Alice eats Apple

⇒ Apple is eaten by a Person

⇒ Apple ∈ Food


DL query:

Query (class expression)


Person and (eats some Food)

Query results

Subclasses (1 of 1)

 owl:Nothing

Instances (1 of 1)

 Alice

Person and (eats some Fruit)

Fruit \subseteq Food
Alice eats Apple
 \Rightarrow Alice eats some Fruit
 \Rightarrow Alice \in Person

Interpretation

This is not explicitly asserted:

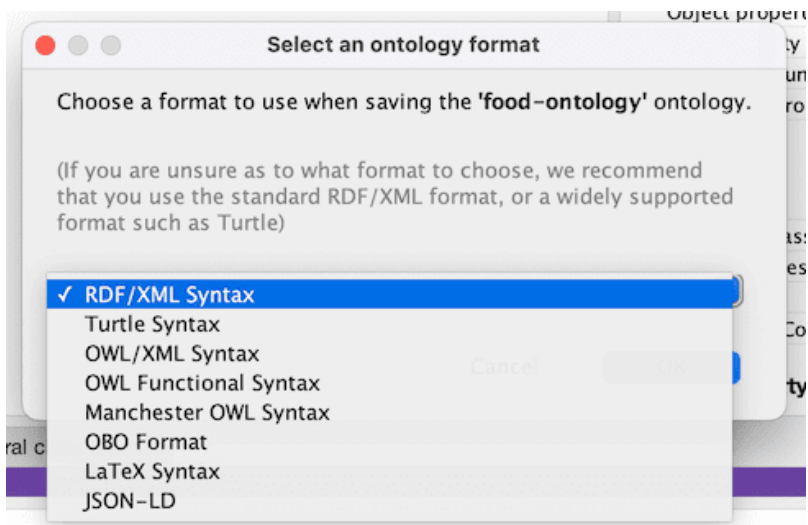
But the system figured it out automatically.

Apple is Food.

That's what makes ontologies powerful!

- Knowledge graphs
- AI reasoning
- Semantic search
- Rule-based systems
- RAG grounding

Save ontology file



- Save & Load as RDF/XML format file

What is saved? Persistent Knowledge

ontology assertions are saved:

```
Fruit  $\subseteq$  Food  
Alice eats Apple  
Apple  $\in$  Fruit  
Domain(eats) = Person  
Range(eats) = Food
```

What is NOT saved?

Reasoner inferences are NOT saved:

```
Apple ∈ Food  
Alice ∈ Person
```

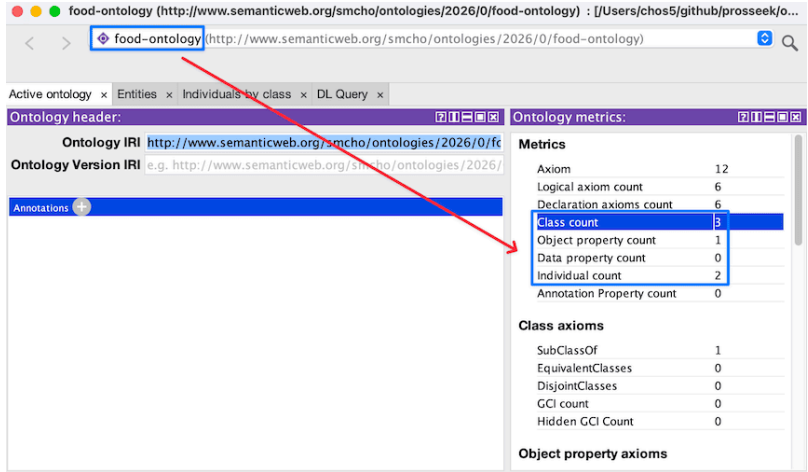
DL Queries are NOT saved.

```
Food and (inverse eats some Person)
```

- This is "Query", not Ontology Assertion.
- Inferences are not saved, only explicit assertions are saved.
- When you close and reopen the ontology file, the inferred types will be lost.

Load ontology file

- Open the saved RDF/XML file in Protégé
 - Choose File -> Open
 - Select the saved RDF/XML file



- All the explicit assertions are loaded
- Run the Reasoner again to re-infer types

File Formats

- Turtle file: `.ttl`
- RDF/XML file: `.owl` or `.rdf`
- OWL Functional Syntax: `.ofn`
- OWL/XML Syntax: `.owlxml`
- Manchester Syntax: `.omn`
- JSON-LD: `.jsonld`

Install Plugins

1. File -> Check for plugins...
2. Select and install desired plugins (e.g., SHACL, SPARQL Query)
3. Restart Protégé to activate plugins
4. Window -> Tabs -> Select installed plugin tab

