Reasoning in Protégé

Nicolò Pratelli

Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche – Pisa

A.A. 2024-2025

In the OWL seminars

In the last seminars you learned how to:

- create an OWL ontology in Protégé
- set the ontology IRI and the prefixes
- create classes and their subclasses
- create properties and their subproperties
- set property domain and range
- create individuals and set their class
- define OWL axioms in Protégé:
  - using class expressions
  - using property restrictions
  - defining class expression axioms
  - defining property axioms

Reasoning in OWL

A **reasoner** is a piece of software that takes as input a knowledge base and performs several operations to achieve:

- **consistency checking**, i.e. whether the KB contains any contradictions
- **inference generation**, i.e. obtaining new knowledge that can be inferred from the KB

Several OWL reasoners exist: some support all axioms of OWL DL (e.g. Pellet, HermiT), while others are tailored to specific profiles of OWL (e.g. ELK for OWL EL)

Reasoning Tasks

The main reasoning tasks performed by the reasoner on the TBox of an ontology are:

- **subsumption checking**: class expression A is subsumed by B if, in each model of the ontology, each instance of A is also an instance of B
- **satisfiability checking**: class expression A is satisfiable if an instance of A exists in at least one model of the ontology, otherwise A is unsatisfiable

On the ABox, the reasoner performs instance checking to find out whether an individual is instance of a class

Open World Assumption

When working with OWL, you have to keep in mind the open world assumption

This constitutes a significant difference between an OWL KB and a traditional relational database (e.g. MySQL)

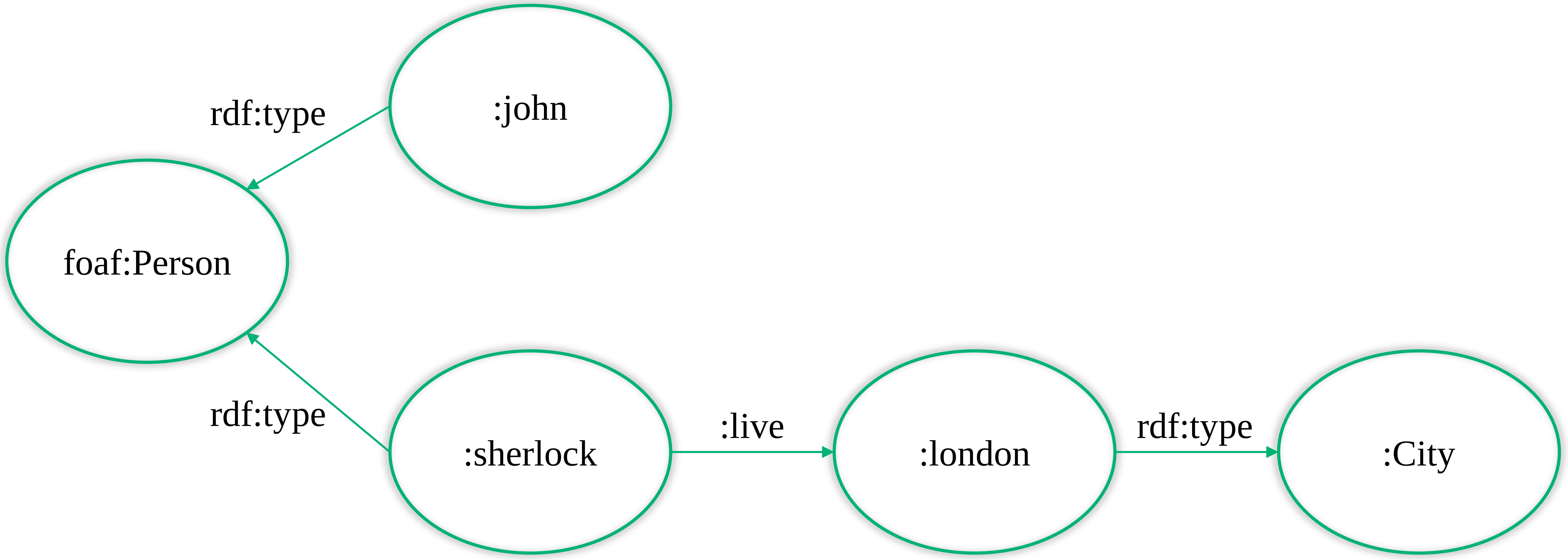
In a relational DB, anything that is not explicitly stated in the DB is considered false → Closed world assumption

In an OWL KB, anything that is not explicitly stated in the KB is considered possible → Open world assumption

Open World Assumption Example

Does Sherlock live in London? Yes

Does John live in London? We don't know



Inconsistencies

OWL reasoners follow the open world assumption, therefore everything that is not stated explicitly in the knowledge base is considered possible

Furthermore, the reasoner will always try to find an interpretation of the knowledge that avoids contradictions

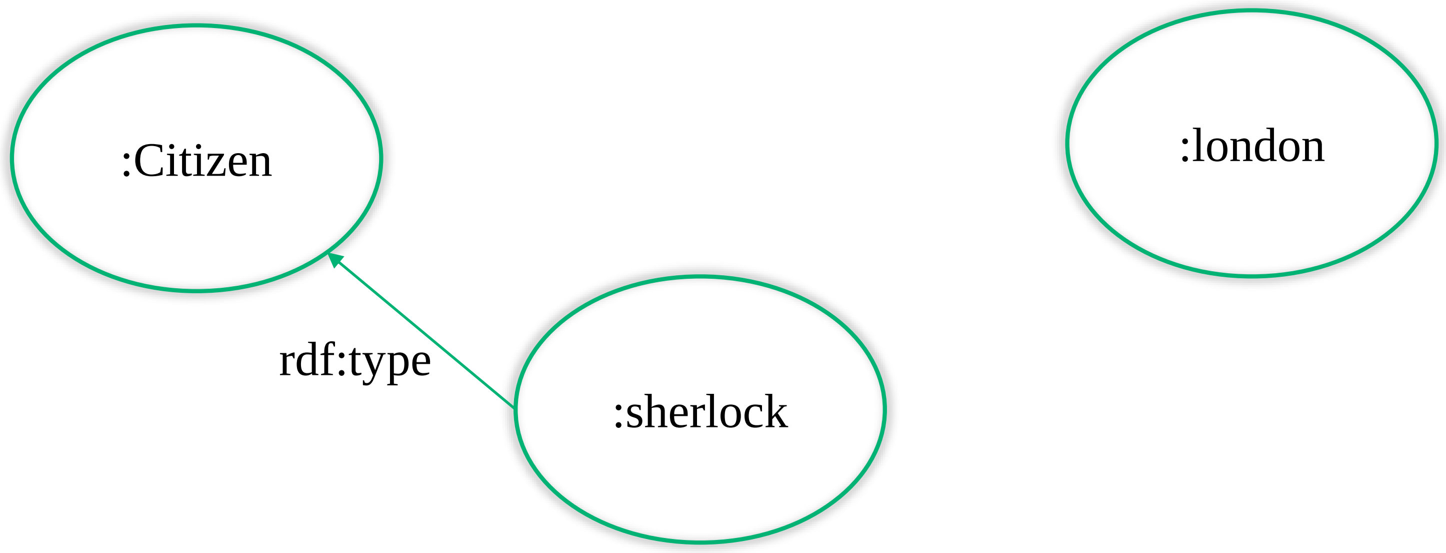
For these reasons, in some cases you may expect the reasoner to generate an inconsistency, but in fact it doesn’t

In the next few slides we will see some examples

Consistent or Inconsistent? (1)

Suppose that Sherlock has type Citizen, and you have stated as an axiom that a person must be citizen of 1 country

But Sherlock isn't citizen of any city in the KB… what happens?

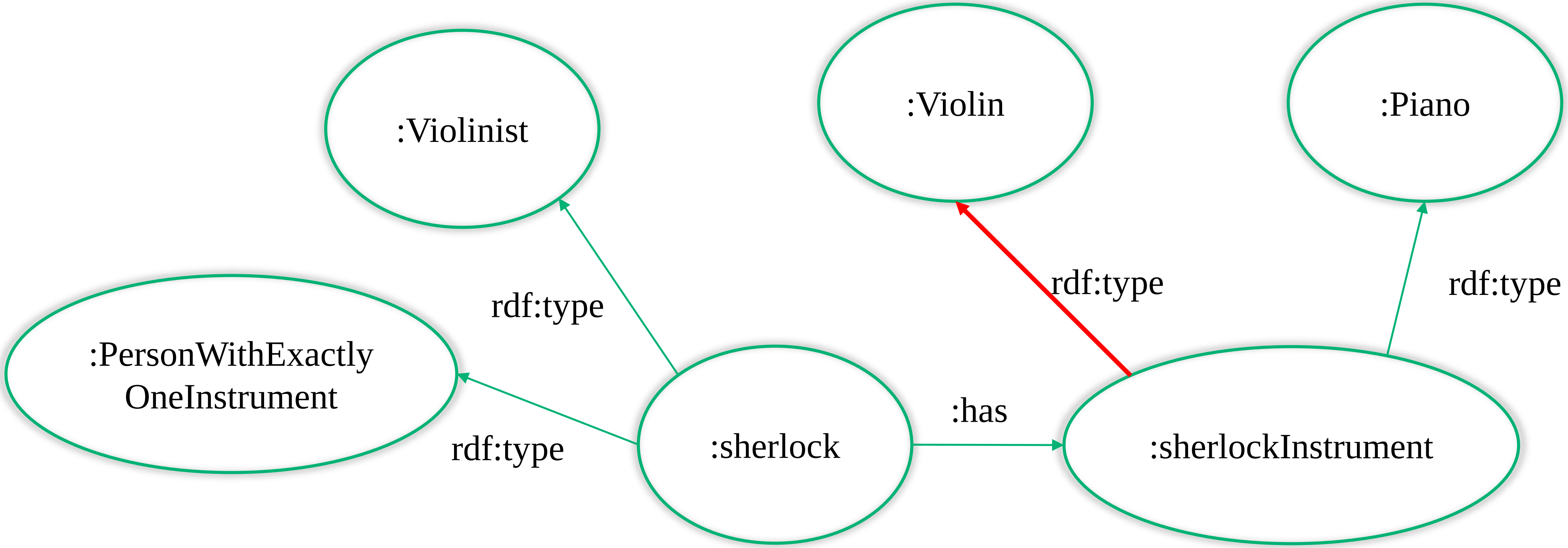


Nothing! Sherlock is citizen of one city, we just don't know about it!

Consistent or Inconsistent? (2)

Suppose that Sherlock is violinist and also a person with exactly one instrument, and all violinists must have at least 1 violin. … but Sherlock has a piano! What happens?

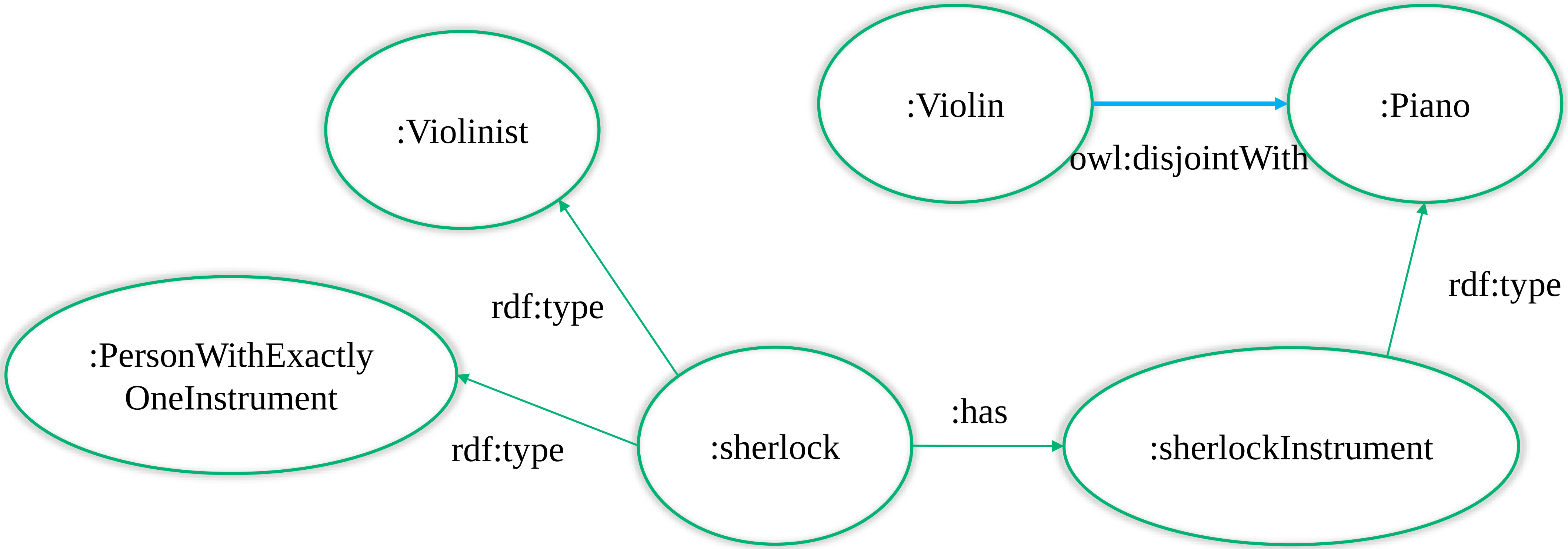
Sherlock’s instrument is both a violin and a piano!



Consistent or Inconsistent? (2)

This is clearly wrong, but the reasoner cannot know about it unless you define the :Violin and :Piano classes as disjoint

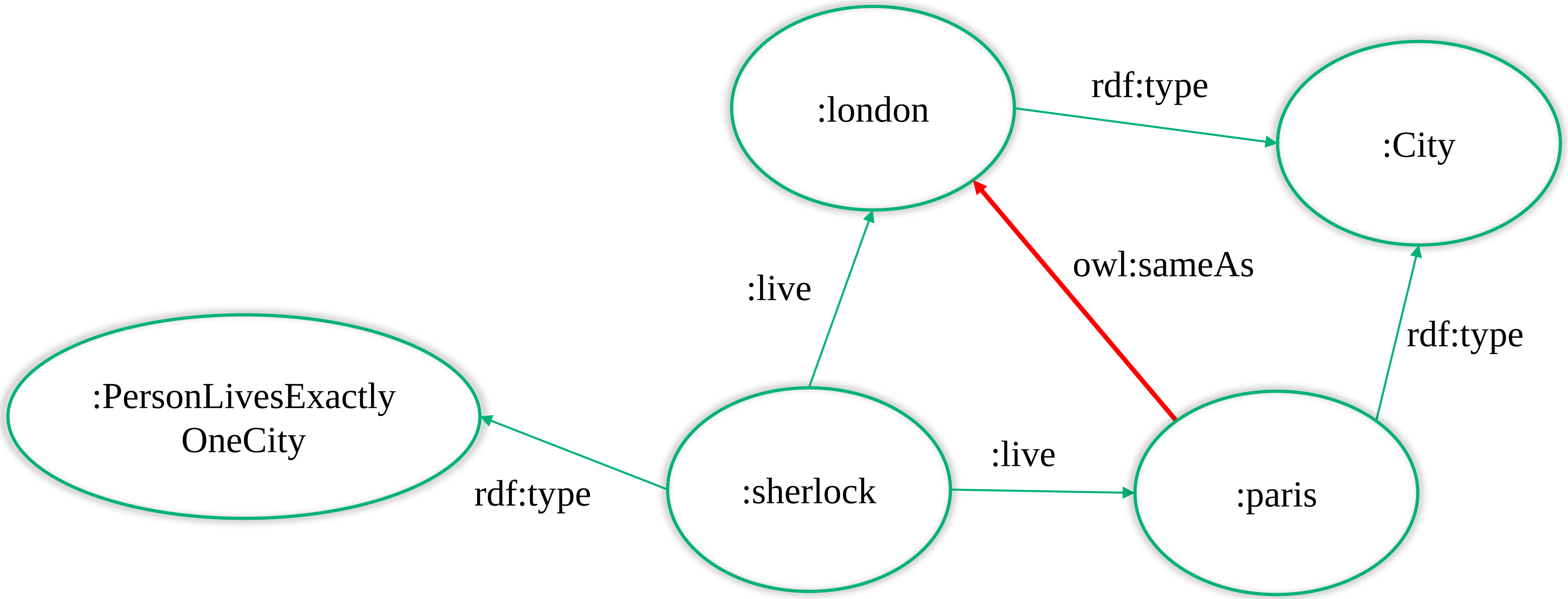
Only in that case it will report an inconsistency



Consistent or Inconsistent? (3)

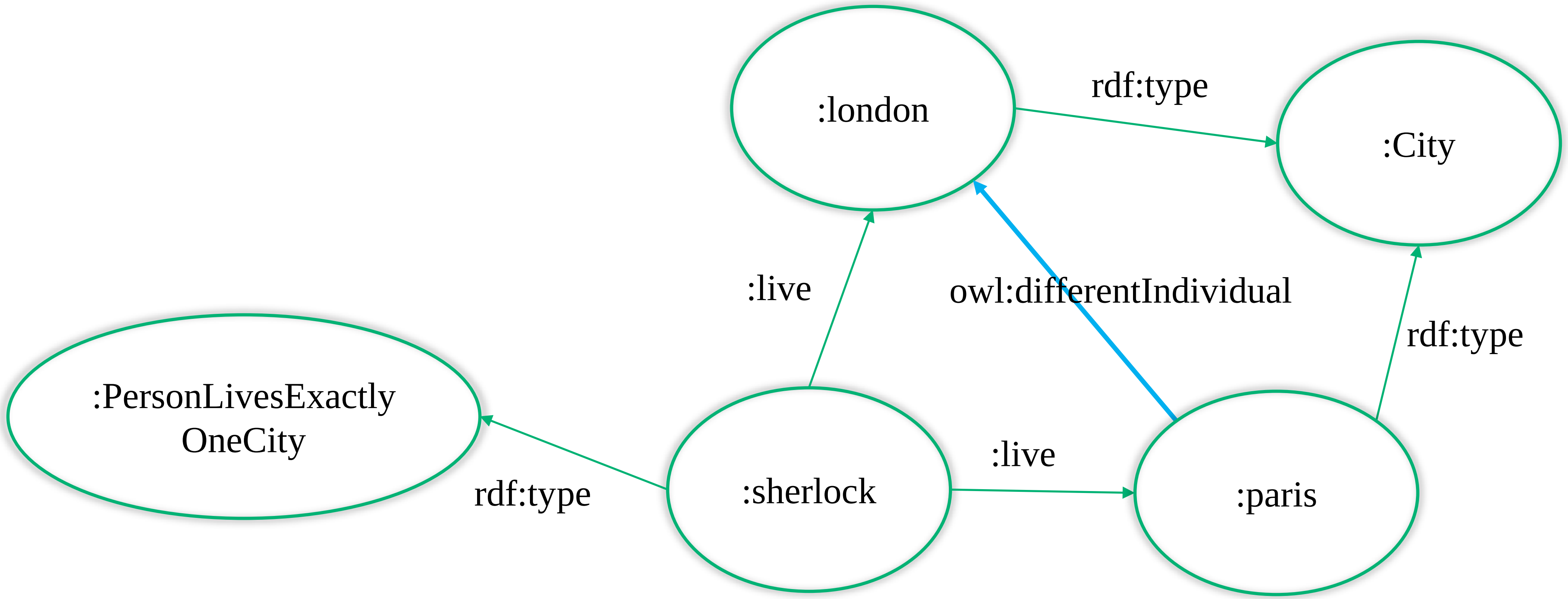
Now suppose that Sherlock lives in London and Sherlock must live exactly in one city… but Sherlock also lives in Paris! What happens?

London and Paris are inferred to be the same city!



Consistent or Inconsistent? (3)

To get an inconsistency here, use owl:differentIndividual



Generating Inconsistencies

Some common ways to generate inconsistencies include:

- putting an individual in two disjoint classes
- declaring two equivalent classes as disjoint
- violating a maximum or exact cardinality constraint using different individuals

When running the reasoner, Protégé will report the inconsistency and also offer one or more explanations for it

Installing the Reasoner

Today we will see how reasoning works in Protégé

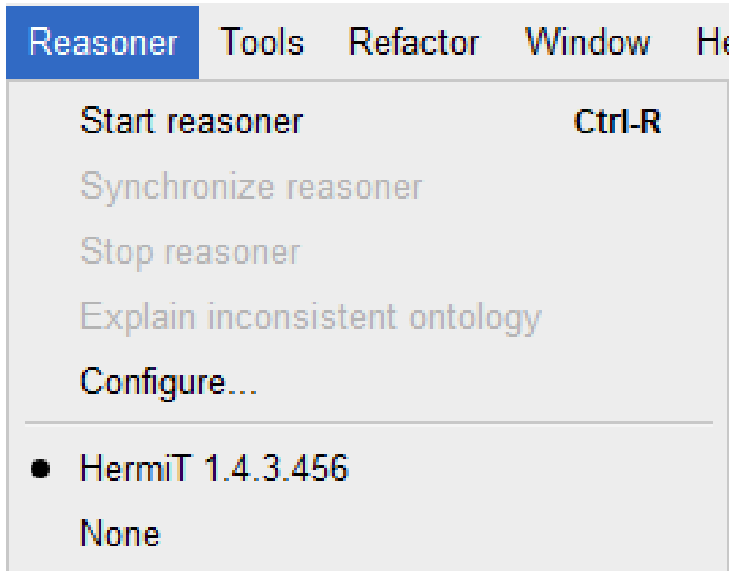
First of all, reasoners are installed through plugins, and Protégé supports several of them

To use a reasoner, go to the Reasoner menu and check if you see HermiT in the list

If you don’t see it, go to File > Check for Plugins, look for HermiT and select it, then click Install

Running the Reasoner

- To run the reasoner, simply click on “**Start reasoner**”
- Always **save** your ontology before starting the reasoner, because it may change its structure
- To view the statements inferred by the reasoner, select the “**Show Inferences**” checkbox (bottom right)
- Inferences are highlighted in yellow



Complex Role Inclusion Axiom

The complex role inclusion axiom allows you to state that a property is the composition of other properties

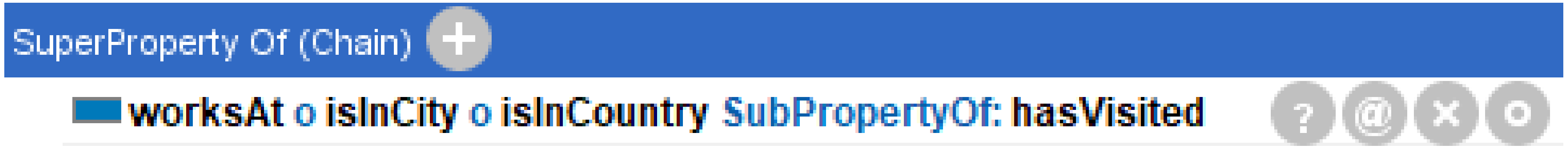
You use properties p_1, p_2, … p_{n-1} to form a chain from individual i_1 to individual i_n, and this chain implies that property q connects directly the first individual i_1 to the last individual i_n

Example: If Sherlock visits Louvre Museum, and Louvre Museum is in Paris, and Paris is in France, then we can infer that Sherlock has visited France

Complex Role Inclusion in Protégé

In Protégé, the complex role inclusion axiom can be expressed through the section of the property description that is called SuperObjectPropertyOf (Chain)

You start from the consequence (in this case hasVisited), and insert the composition of the properties found in the premise (for the composition symbol, simply type "o")


SuperProperty Of (Chain)

worksAt o isInCity o isInCountry SubPropertyOf: hasVisited

Instructions for the Exercise

- Today we will complete the exercise from the last seminar, adding axioms to the ontology about pizza
- In addition, define the complex role inclusion axiom described in the text of the exercise
- Install and run the reasoner as requested in the exercise
- Generate inconsistencies and report them in a text file
- Save the resulting knowledge base in Turtle format and ask us to verify its correctness after you have finished

Exercise (1/4)

- Define a new **subclass** of Pizza: VeganPizza
- Define these two new **subclasses** of NamedPizza: SausagePizza and NapoletanaPizza
- Define the following OWL axioms:
  - **SausagePizza** has a RedBase and the following toppings: SausageTopping, MushroomTopping and MozzarellaTopping
  - **NapoletanaPizza** has a RedBase and the following toppings: AnchoviesTopper and CaperTopping
  - **VeganPizza** is equivalent to a Pizza with minimum 1 VegetableTopping and only VegetableTopping

Exercise (2/4)

- Define a new object property called hasVisited, with domain Person and range Location
- Define the following complex role inclusion axiom:
    - If a person has bought a pizza and that pizza was made by a pizzeria and that pizzeria is located in a city, then that person has visited that city

Exercise (3/4)

- Now define the following individuals and put them in the appropriate classes:
  - instances of *FourCheesePizza*: fourCheese1
  - instances of *FourSeasonPizza*: fourSeason1
- Now use the individuals above to define assertions (if necessary, create further individuals of *PizzaTopping* and *PizzaBase* classes):
  - fourCheese1 has topping provola, gorgonzola, parmigiano and mozzarella
  - fourCheese1 has a white base
  - fourSeason1 has topping olive, ham, mushroom, artichoke and mozzarella
  - fourSeason1 has a red base
- <u>Save the ontology to a Turtle file</u>

Exercise (4/4)

- After saving the ontology, run the reasoner to check that there are no inconsistencies
- Finally, think of two different ways to generate an inconsistency in this KB, and verify them
- Report the inconsistencies in a text file
- Send us both the ontology and the text file via Moodle