МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Практическая работа № 2
Применение онтологии к решению задач описания информационных систем
Методы и средства проектирования информационно-аналитических систем

(наименование дисциплины (модуля) в соответствии с учебным планом)

Уровень: специалитет
(бакалавриат, магистратура, специалитет)

Форма обучения: очная
(очная, очно-заочная, заочная)

Направление(-я) подготовки: 10.05.04 «Информационно-аналитические системы безопасности»
(код(-ы) и наименование(-я))

Институт: Институт кибербезопасности и цифровых технологий (ИКБ)
(полное и краткое наименование)

Кафедра: Информационно-аналитические системы кибербезопасности (КБ-2)
(полное и краткое наименование кафедры, реализующей дисциплину (модуль))

Используются в данной редакции с учебного года 2023/24
(учебный год цифрами)

Проверено и согласовано «____» _____20___г.

_____ _____
(подпись директора Института/Филиала с расшифровкой)

Москва 2024 г.

Цель работы: получить навык использования методов и инструментов описания баз знаний

Задание на практическую работу: создайте два описания баз знаний по предложенным в методической разработке последовательности и создайте свою база знаний.

В отчет по практической работе включить привести снимки экранов созданных баз знаний по двум представленным вариантам (продемонстрировать срабатывание ризонера). Описание свой базы знаний привести в текстовой форме и дополнить схемой из приложения и файлом.

Теоретические сведения

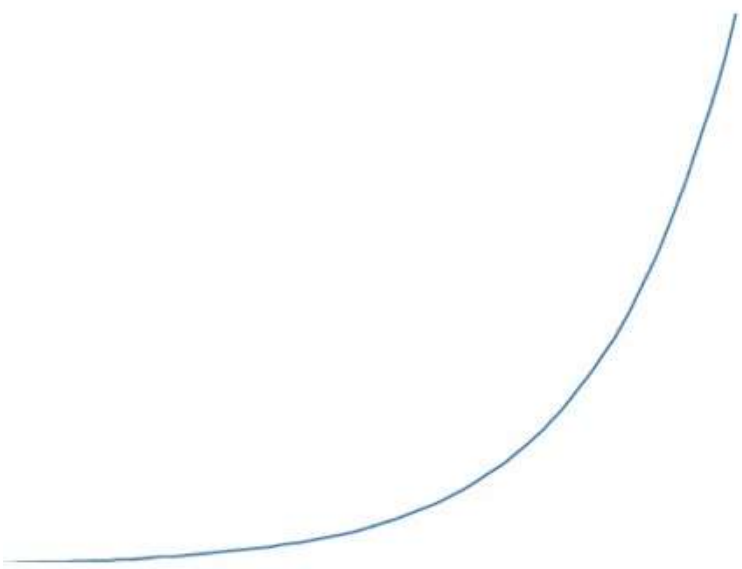
Онтологии служат для систем организации знаний и применяются в тех областях, где требуется обнаружить новые факты, выявить скрытые взаимосвязи между элементами (например, рекомендательные и экспертные системы). Это альтернатива классическим базам данных, которые используют «гипотезу закрытого мира», когда все, чего нет в базе данных – не существует. В противоположность этому онтологии используют «гипотезу открытого мира», то есть если чего-то нет в базе знаний, то это не обязательно не существует, а просто не описано.

В информационной безопасности тоже есть задачи, где онтологии приносят пользу. Уменьшение возможностей для атак злоумышленников за счет установки патчей и обновления ПО никогда не дает 100%-защиту, поскольку остаются уязвимости, связанные с небезопасным пользовательским поведением, небезопасной конфигурацией инфраструктуры, ошибками в настройках внедренных средств защиты, нестойкими паролями и недостаточным контролем за привилегированным доступом. Противодействовать zero-day атакам очень сложно, так как в защищающихся системах не существует правил для обнаружения таких атак, распознавать атаку и реагировать надо "на лету". Одним из способов распознать атаку при отсутствии готового шаблона распознавания является использование накопленных знаний и «логический вывод» (ризонинг) из этих знаний с учетом имеющейся информации о происходящем событии. Способом хранения таких знаний могут быть онтологии, которые хранят информацию о взаимосвязях различных сущностей между собой.

Введение

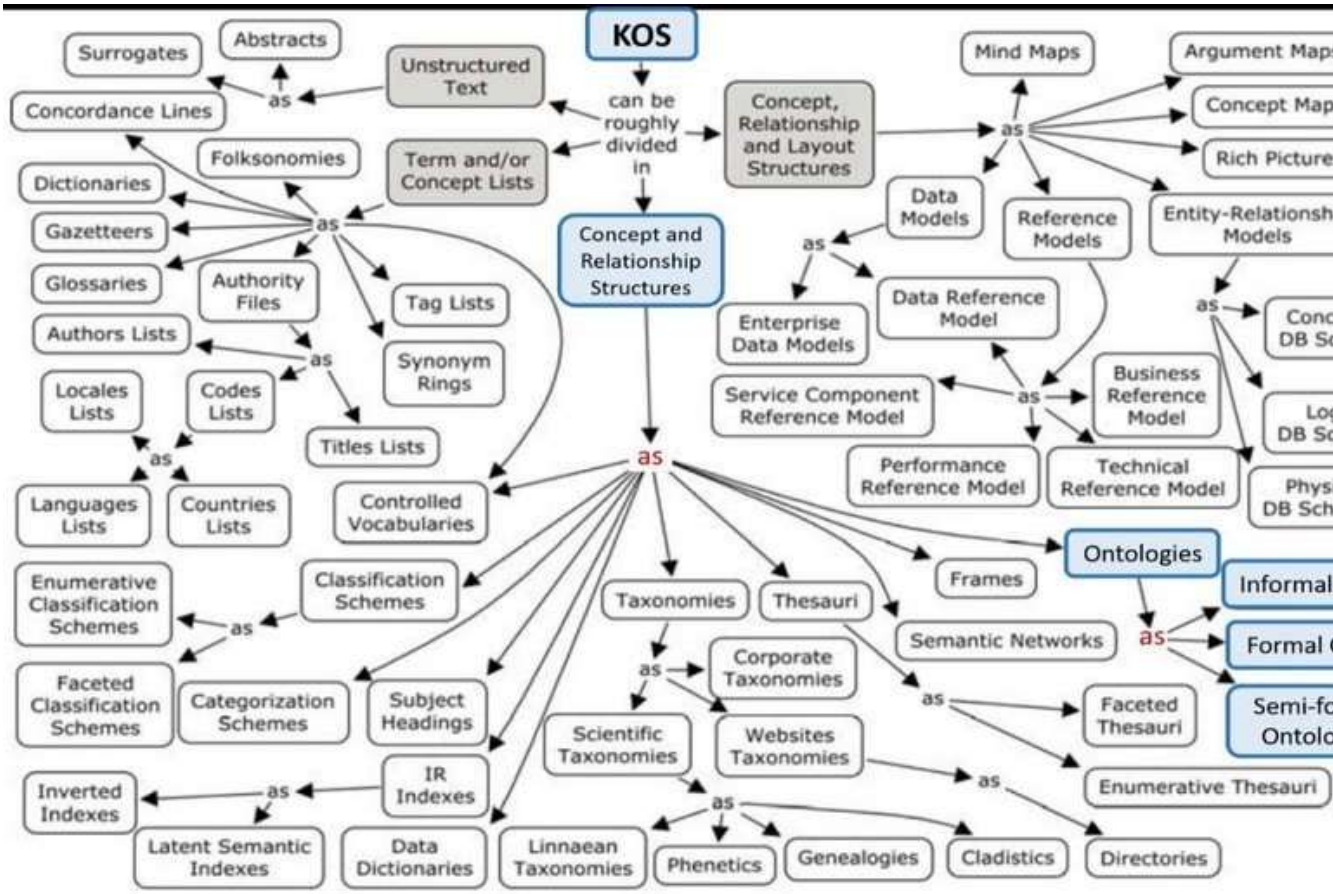
В мире каждый день появляется много нового, все чаще возникают новые предметные области, о возможности появления которых мы даже не задумывались еще несколько лет назад. При этом старые предметные области уходят, не выдержав конкуренции. Каждая предметная область характеризуется прежде всего специальными знаниями, описывающими объекты этой области и их свойства. Практическое использование таких знаний является уделом экспертов. Собственно, в обладании такими знаниям и состоит профессиональная компетентность эксперта. Однако оставаться всезнающим экспертом в наши дни становится все сложнее, поскольку

ускоряется смена технологий и содержания экспертных знаний. В том числе это происходит потому, что практически в любой предметной области, в любой отрасли генерируется огромное количество данных.



Подмечено, что сейчас количество данных увеличивается по экспоненте, то есть скорость изменения количества данных линейно зависит от текущего объема данных. Чем больше становится данных, тем быстрее скорость увеличения их количества. Это очень серьезный тренд – и психологически, и технологически. Такой большой объем новой информации становится все сложнее передавать, обрабатывать, сохранять, несмотря на существенное увеличение производительности оборудования. Но

самая большая проблема заключается не в количестве данных, а в их неструктурированности. Данные появляются из различных источников, в разных форматах, в разное время. Поэтому перед использованием в практических задачах данные упорядочивают, преобразуют, приводят в форму, эффективную для хранения и использования. Традиционным способом хранения и использования данных являются реляционные базы данных, в которых данные хранятся в виде связанных таблиц. Однако не всегда табличные данные эффективны в использовании, поэтому стали появляться альтернативные формы, например, системы организации знаний (Knowledge Organization Systems, KOS), как правило базирующиеся на графах знаний.



Пример системы организации знаний

Источник.

Для хранения знаний используются разные структуры:

- управляемые словари: обеспечивают способ организации знаний для последующего поиска, используются в схемах предметной индексации, предметных рубриках, тезаурусах, таксономиях и других системах организации знаний,
- тезаурусы: объединяют термины в группы по определенному признаку, например, с учетом похожести (синонимы), таксономии: категорированные слова, упорядоченные по иерархическому признаку,
- онтологии: формальное описание знаний из какого-то домена (предметной области) с учетом имеющихся сложных правил и связей между элементами, позволяющим сделать автоматическое извлечение знаний,
- датасеты: наборы машиночитаемых данных.

Онтологии служат для систем организации знаний и применяются в тех областях, где требуется обнаружить новые факты, выявить скрытые взаимосвязи между элементами (например, рекомендательные и экспертные системы). Это альтернатива классическим базам данных, которые используют «гипотезу закрытого мира», когда все, чего нет в базе данных – не существует. В противоположность этому онтологии используют «гипотезу открытого мира», то есть если чего-то нет в базе знаний, то это не обязательно не существует, а просто не описано.

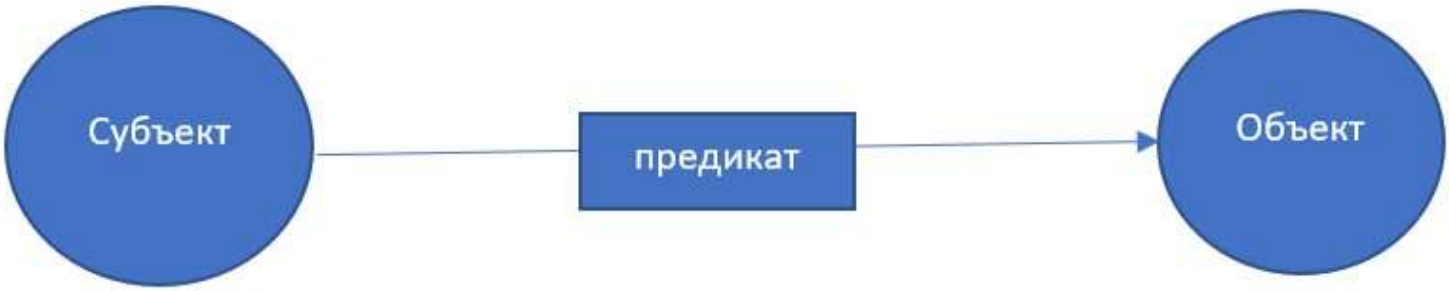
Системы организации знаний на основе онтологий уже очень распространены и используются во многих отраслях. Самый яркий пример это knowledge graph для поиска информации в Интернет, благодаря этой технологии качество поиска стало очень высоким. Другие примеры использования онтологий на практике:

- банки используют графы знаний для анализа транзакций (fraud detection), в консалтинге используются графы на основе юридических документов,
- в здравоохранении используются накопленные сведения на основе данных о здоровье пациентов, Health Electronic Record,
- в промышленности графы знаний используются для анализа цепочек поставщиков (supplychain management), в целом для Индустрии 4.0 характерно взаимодействие киберфизических систем между собой, что приводит к автоматизации и необходимости управлять знаниями,
- во многих отраслях базы знаний используются для организации работы чат-ботов, в том числе и для обработки сложных запросов на естественном языке (например, сервис asknow),
- онтологии могут применяться для широкого круга задач обработки естественного языка: аннотирование текстов с помощью онтологий, извлечение знаний, NER, Named Entity Linking, Relation Linking, автоматический вывод новых знаний, ризонинг (reasoning). Направление SemTech переживает в настоящее время бурное развитие, в том числе в России, но информации на русском языке по данной тематике по-прежнему очень мало.

В информационной безопасности тоже есть задачи, где онтологии приносят пользу. Уменьшение возможностей для атак злоумышленников за счет установки патчей и обновления ПО никогда не дает 100%-защиту, поскольку остаются уязвимости, связанные с небезопасным пользовательским поведением, небезопасной конфигурацией инфраструктуры, ошибками в настройках внедренных средств защиты, нестойкими паролями и недостаточным контролем за привилегированным доступом. Противодействовать zero-day атакам очень сложно, так как в защищающихся системах не существует правил для обнаружения таких атак, распознавать атаку и реагировать надо "на лету". Одним из способов распознать атаку при отсутствии готового шаблона распознавания является использование накопленных знаний и «логический вывод» (ризонинг) из этих знаний с учетом имеющейся информации о происходящем событии. Способом хранения таких знаний могут быть онтологии, которые хранят информацию о взаимосвязях различных сущностей между собой.

Что такое онтологии?

В математическом основании онтологии лежит так называемая дескриптивная логика (раздел математики), предполагающая, что любая информация, высказанная на естественном языке, может быть представлена в виде цепочки триплетов:

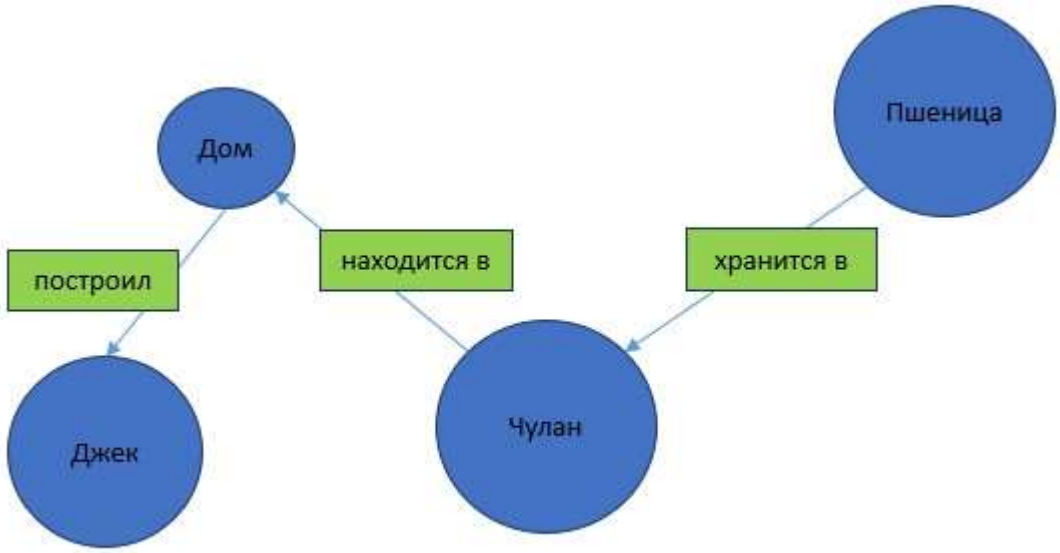


Отрывок из стихотворения «Дом, который построил Джек…» (пер. С.Я. Маршака).

*Вот **дом**,*
*Который **построил Джек**.*
*А это **пшеница**,*
*Которая в тёмном **чулане хранится***
*В **доме**,*
*Который **построил Джек**.*

...

Описанные в стихах отношения между различными сущностями мы можем представить в виде онтологии.



Онтология представляется в виде графа, вершины которого это сущности, а ребра – отношения между сущностями. Считается, что любое утверждение на естественном языке можно представить в виде простых предложений, из которых можно извлечь сущности и отношения между ними. Есть два основных инструмента: RDF (Resource Description Framework) или OWL (Ontology Web Language). OWL позволяет дополнительно описывать логические правила над данными. Онтологии (в отличие от обычных баз данных) позволяют находить скрытые данные. Обычные хорошо подходят для поиска конкретной информации, а базы знаний нужны там, где надо выявлять новые знания, например, в системах поддержки принятия решений (экспертных системах).

Примеры RDF хранилищ – Virtuoso, 4store, stardog.

Сила онтологии проявляется в том случае, если подробно и качественно описаны взаимосвязи между ее элементами, с использованием математического аппарата дескриптивной логики. Например, для отношений

можно задать их свойства (функциональное, транзитивное, рефлексивное). И тогда можно автоматически из онтологии извлекать факты, этот процесс называется ризонинг (reasoning), есть типовые алгоритмы ризонинга, основанные на графах. Возможные применения: уточнение характеристик объекта и выделение из набора похожих объектов уникального, поиск похожих объектов, «понимание текста» и отнесение текста к определенному классу, помощь в NLP задачах (NER, Relation Extraction), анализ корневых причин, выявление паттернов в данных. Наиболее популярный редактор онтологий, поддерживающий ризонинг, это Protégé. Существуют и другие инструменты, например: IBM Watson, Wolfram Alpha.

Создание процесса онтологий обычно выполняется вручную, силами экспертов. Однако есть и примеры автоматизированного создания онтологий на основе имеющихся баз знаний. Открытые графы знаний (по состоянию на конец 2021 года, по данным The Linked Open Data Cloud):

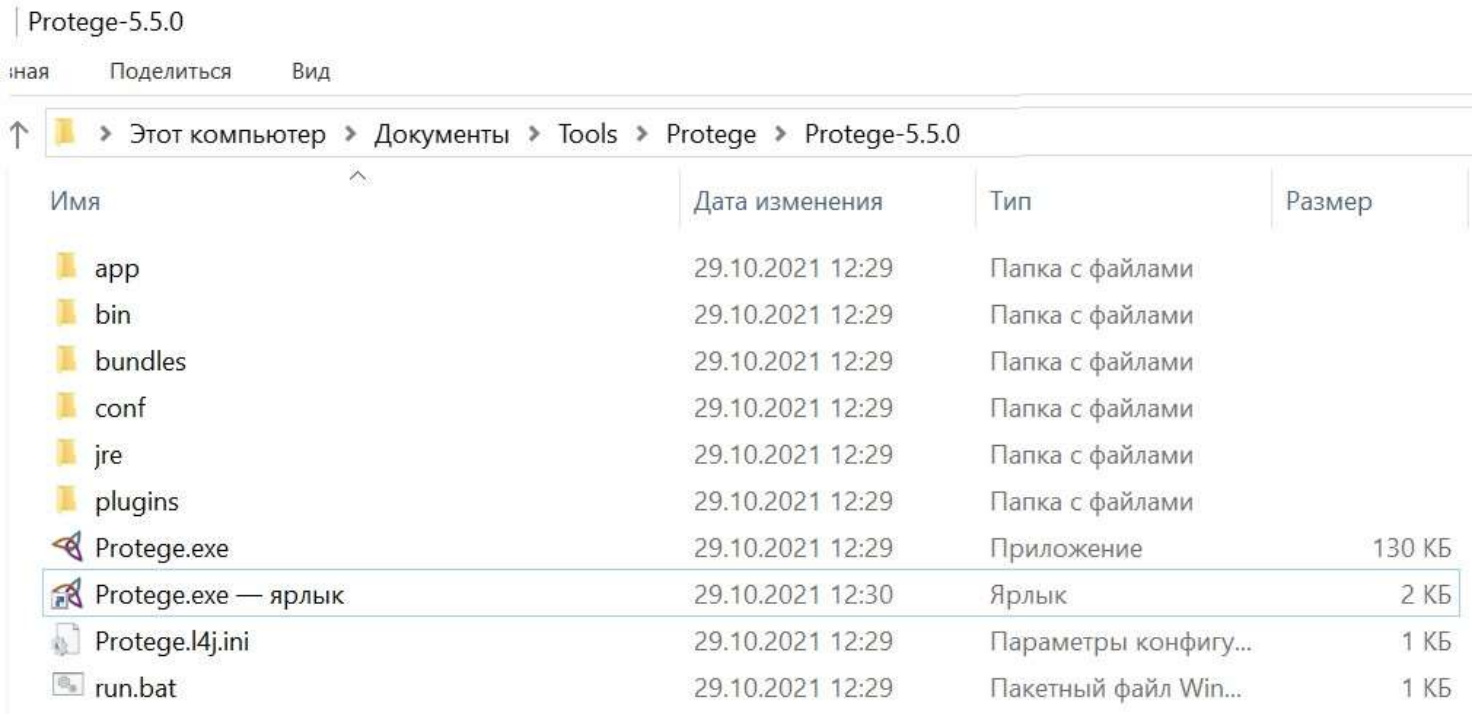
- DBpedia;
- Yago + wordnet.princeton.edu;
- WikiData;
- Открытые базы знаний, в том числе предметные и отраслевые базы знаний, например, медицина: BioPortal, Bio2RDF
- (https://www.sciencedirect.com/science/article/pii/S1532046408000415?via%3Dihub), PubMed.

Описанные выше инструменты содержат различные методы работы с онтологиями. Однако наиболее популярным инструментом является Protégé, который мы далее рассмотрим подробнее.

Работа с Protégé

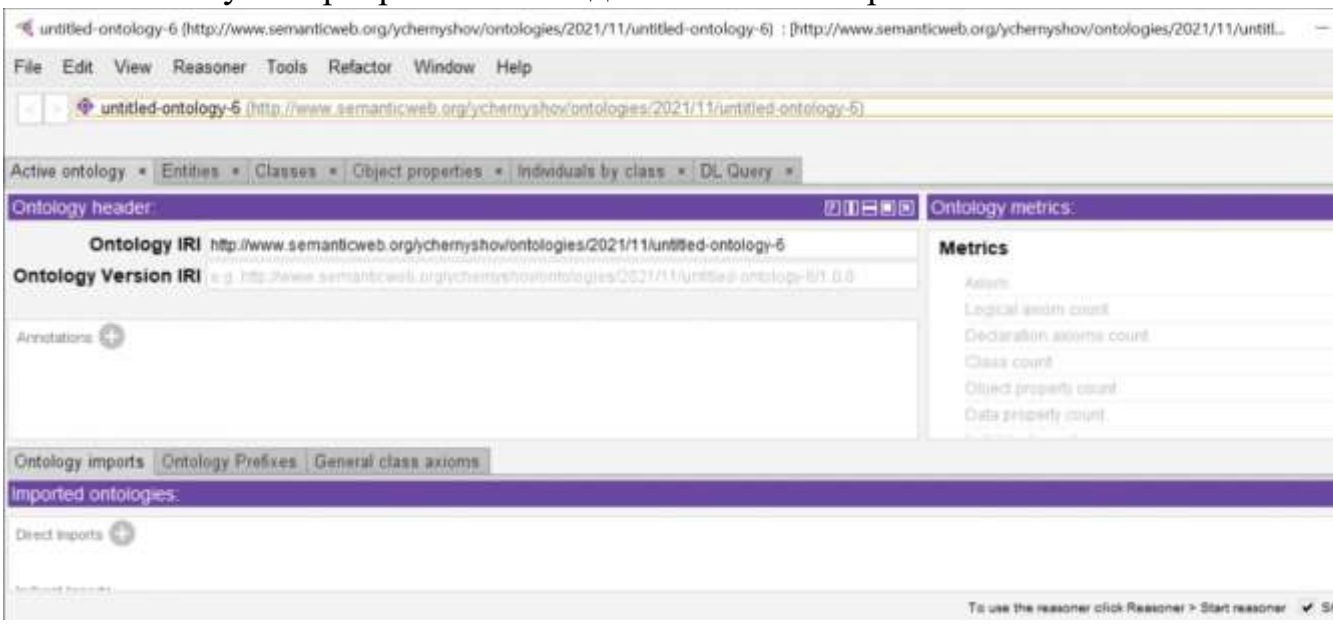
Установка

Программа Protégé для создания, редактирования и использования онтологий разработана в Стэнфордском университете, распространяется бесплатно, ее можно скачать себе на компьютер по ссылке, либо воспользоваться web-версией. На официальной странице можно скачать архив для использования на локальном компьютере. Важно обратить внимание на операционную систему и разрядность процессора (64 или 32). При необходимости использования 32-разрядной версии надо выбрать более низкую версию Protégé, например 4.3. Можно разархивировать в любую папку и запустить файл Protégé.exe. Теперь можно начинать создавать онтологии. Конечно, мы в самом начале большого и сложного пути, но оно того стоит.



Создание проекта

После запуска программы мы видим вот такое стартовое окно.



Тут все написано на Java и на английском, но для понимания происходящего достаточно английского.

Для каждого проекта есть уникальный идентификатор IRI (Internationalized Resource Identifier).

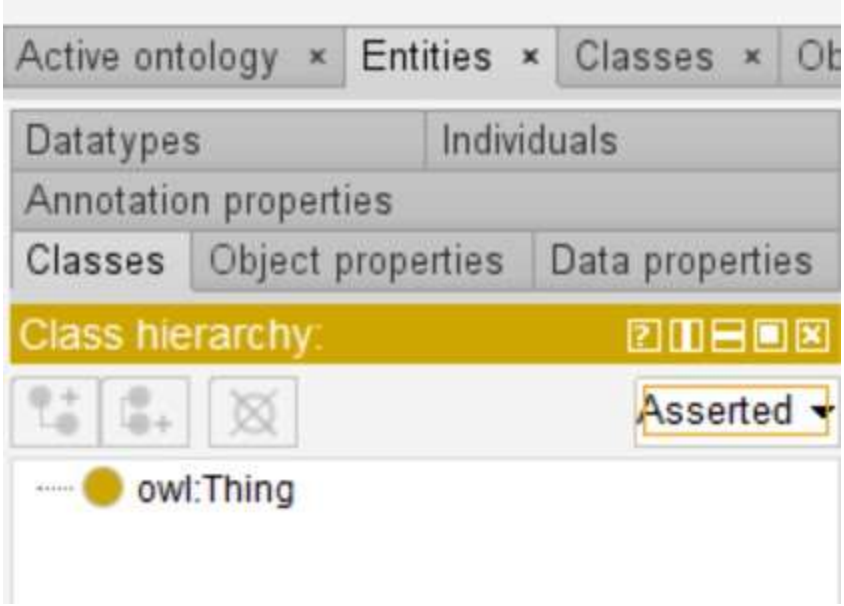
Protégé позволяет записывать триплеты, то есть тройки вида «субъект-предикат-объект».

Раздел «Сущности» (Entities) позволяет описывать субъекты и объекты.

Важные вкладки в этом разделе:

- экземпляры классов (Individuals) это объекты классов, как в объектно-ориентированном программировании. Например, класс «Сервер», объект «prod-serv-002».
- свойства (object properties или data properties), похоже на свойства классов в ООП, однако в онтологии свойства имеют независимую природу, свойство не является неотделимым от класса (как в ООП).

Для предикатов мы можем задавать разные свойства, например:



- функциональный;
- обратный;
- транзитивный.

По описанной онтологии можно применять **ризонер**, который дает предложения по найденным фактам (можно принять или отказаться).

Онтологию можно сохранить во внешний файл, в формате owl, который можно открыть в текстовом редакторе и увидеть, что это xml.

Эти и другие понятия легче разобрать на конкретном примере.

Практические задачи для решения в Protégé

Доступ сотрудников в помещения

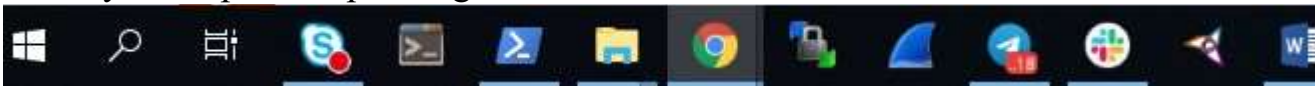
Рассмотрим ситуацию, в которой нам надо определить, в каком помещении находится каждый из сотрудников. При этом нам известно, что

- Иванов не имеет доступа в серверную и кабинет 101;
- Петров не имеет доступа в кабинет 101;
- Сидоров не имеет доступа в хранилище документов.

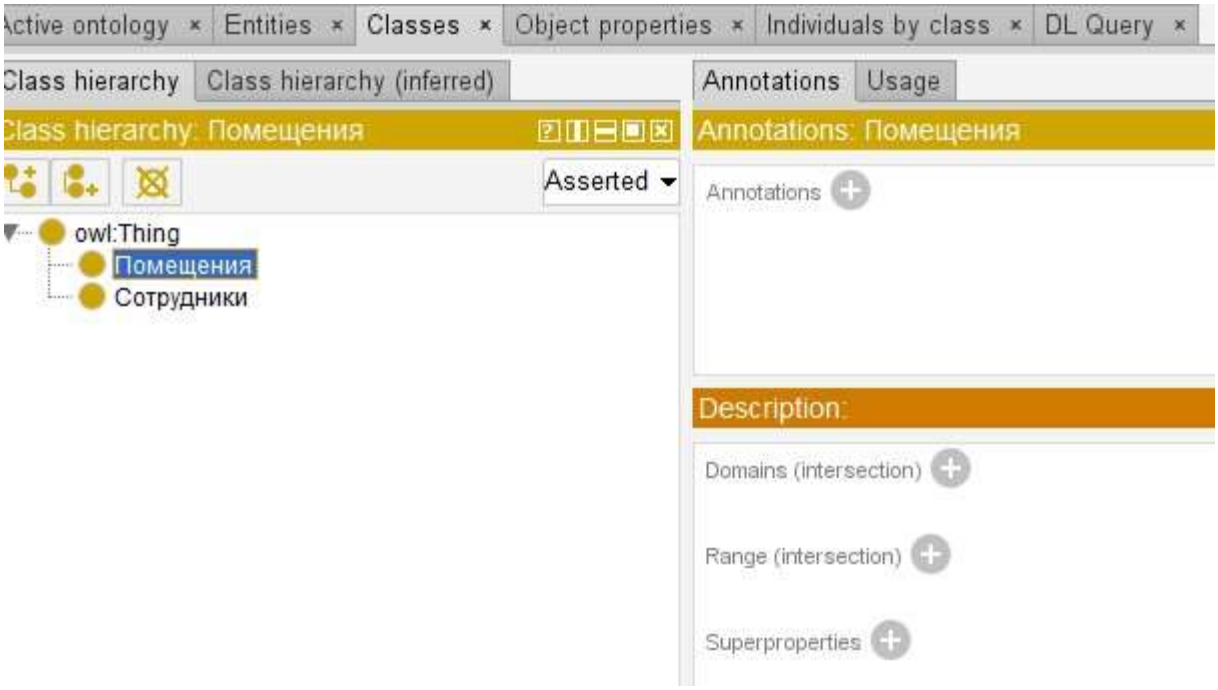
То есть у нас три сотрудника (Иванов, Петров, Сидоров) и три помещения (серверная, кабинет 101 и хранилище документов). Для однозначного решения задачи нам надо более строго определить свойства элементов онтологии, например, зафиксировать тот факт, что у нас нет других сотрудников и помещений и что каждый сотрудник имеет доступ ровно к одному помещению, в противном случае не удастся логически однозначно решить задачу. При введении таких строгих ограничений решение задачи становится тривиальным, из первого условия сразу следует, что Иванов имеет доступ в хранилище документов (поскольку серверная и кабинет 101 исключаются), Петров (которому остались серверная и кабинет 101) имеет доступ в серверную, а

Сидорову остался кабинет 101. Тем проще нам будет проверить результат работы ризонера в Protégé, которому мы и доверим решение этой задачи.

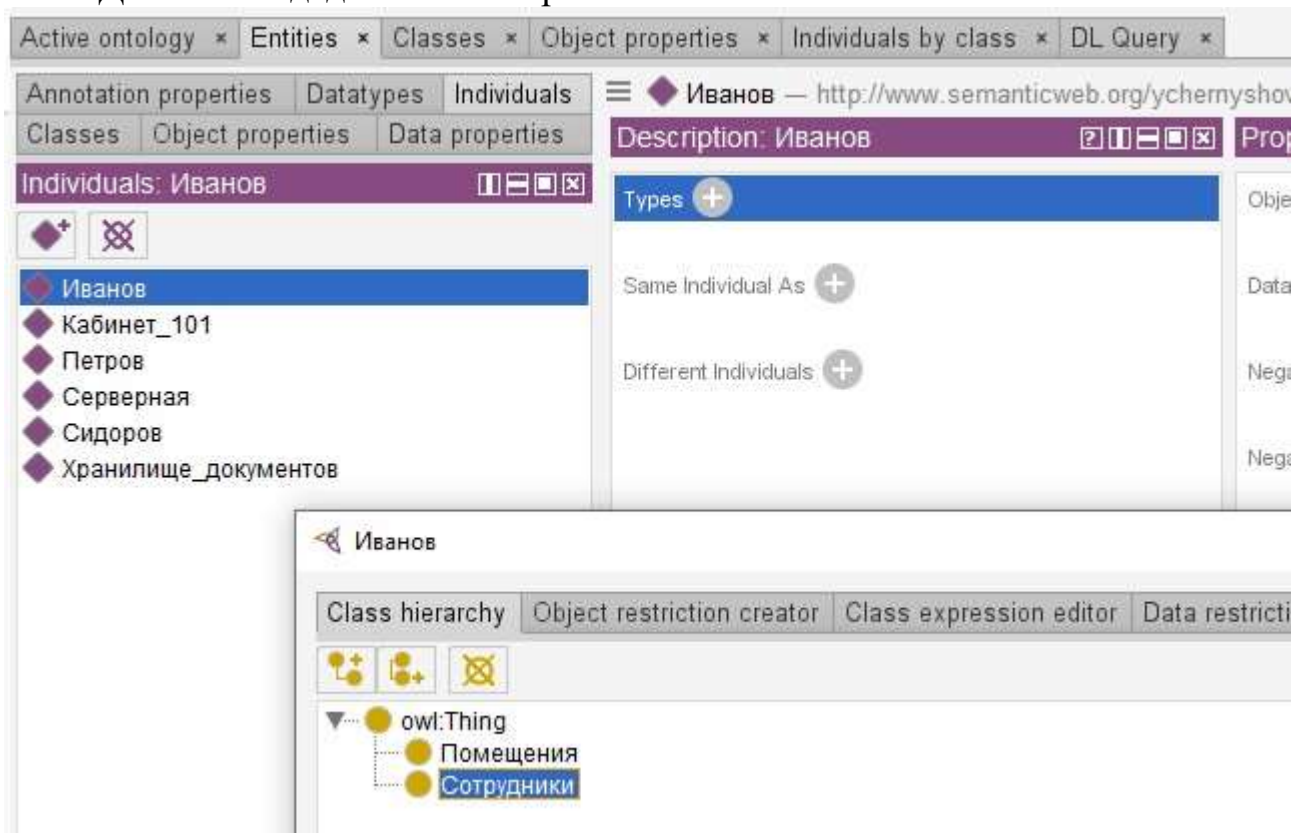
Запускаем редактор Protégé

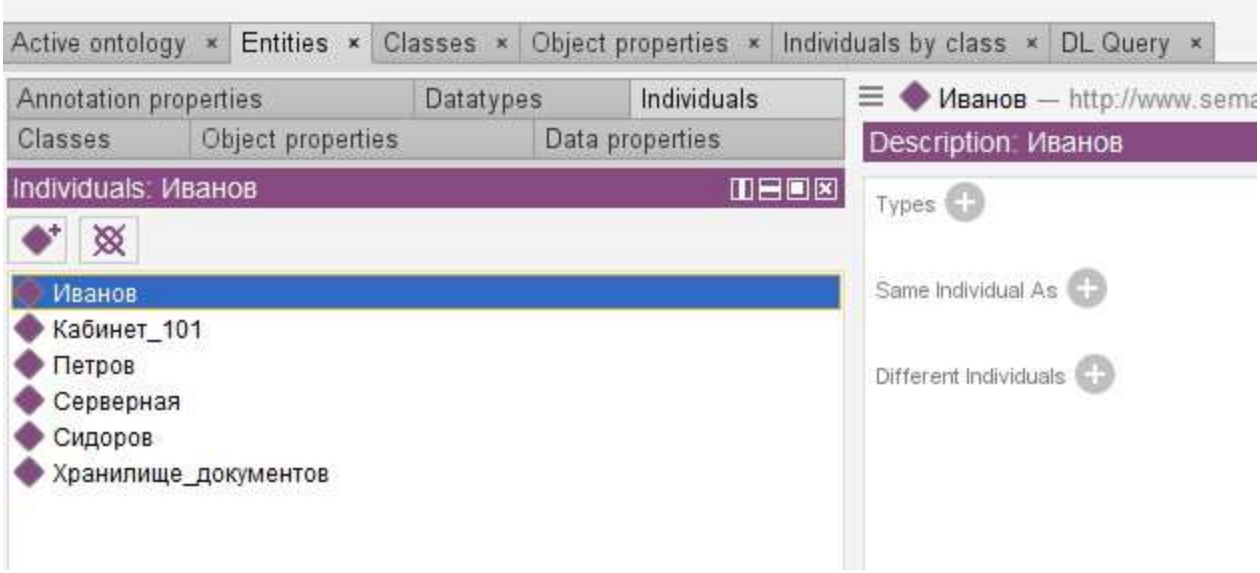


И в появившемся рабочем окне начнем создавать онтологию для нашей задачи. Сначала создадим два класса: «Сотрудники» и «Помещения».

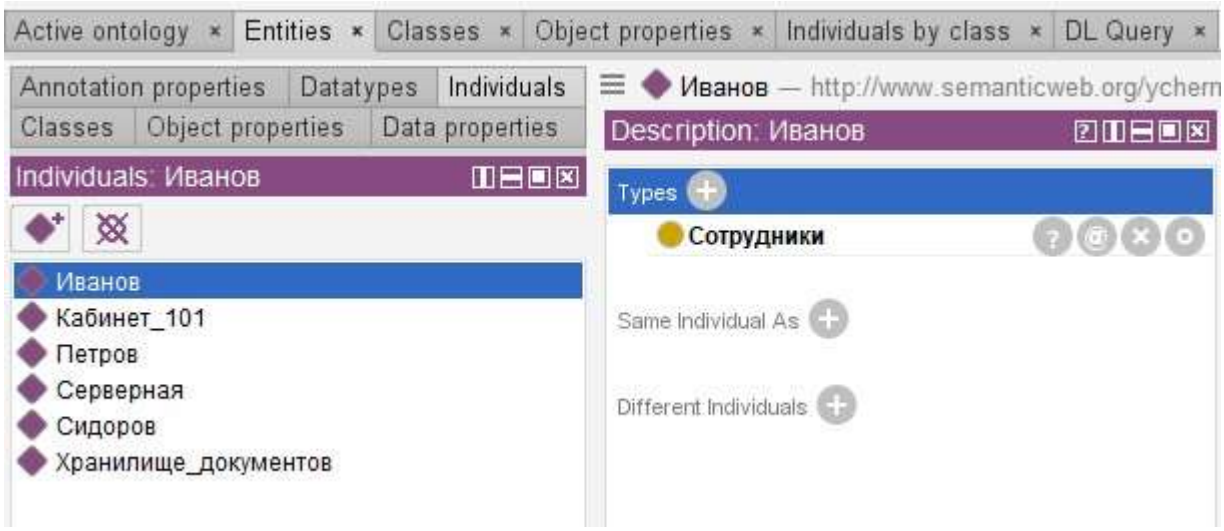


Давайте создадим экземпляры этих классов.

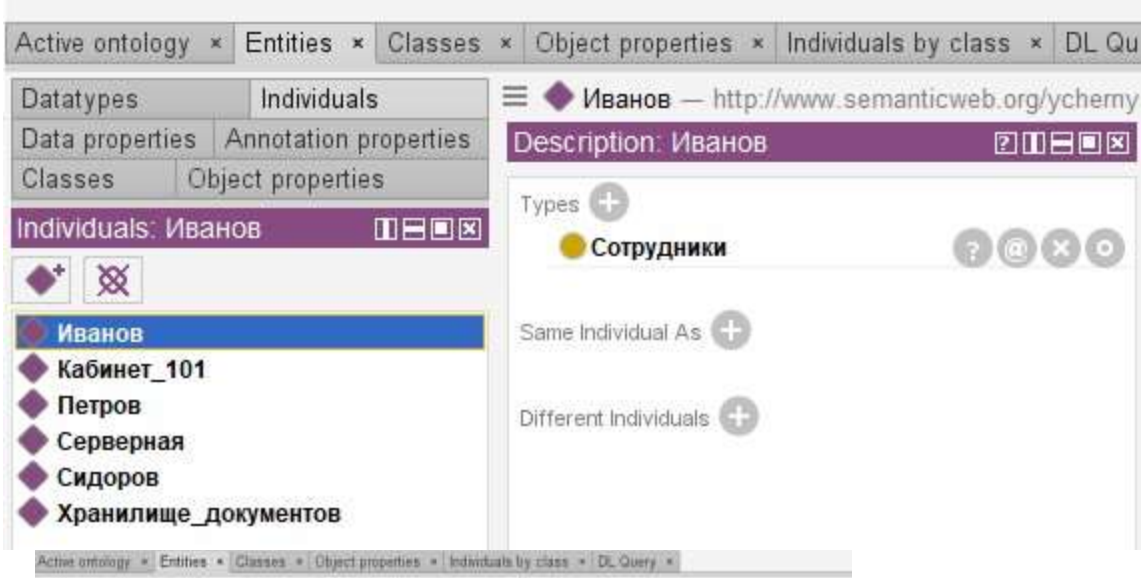


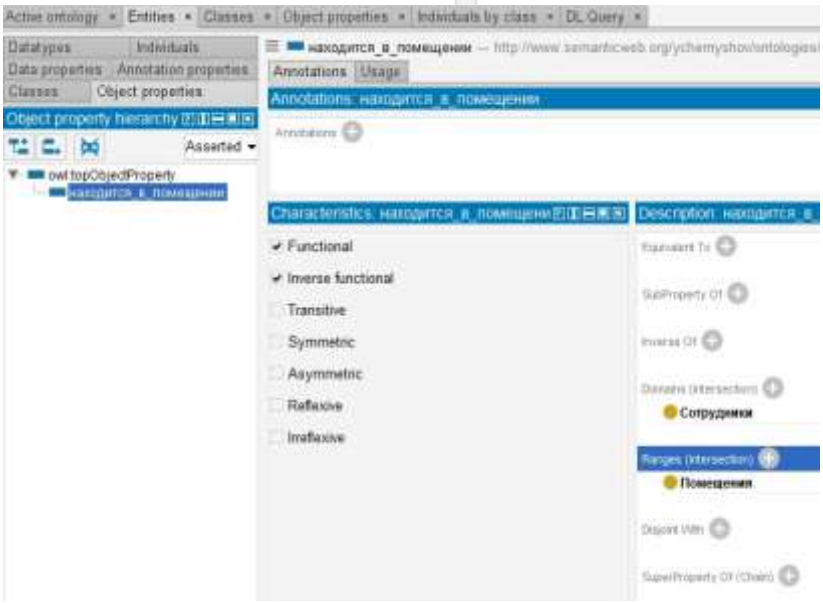


Аналогичную процедуру проделаем для остальных сотрудников (Петров, Сидоров) и помещений.

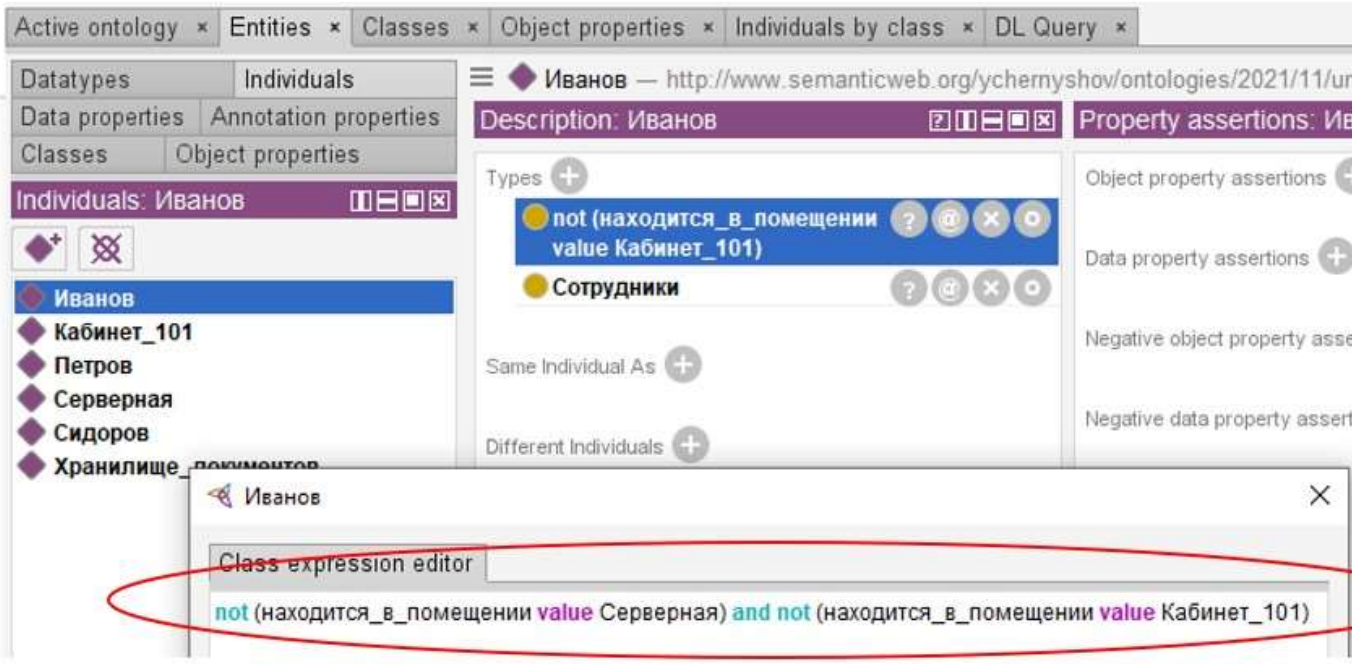


Теперь создадим свойство объектов «находится_в_помещении», которое будет связывать Сотрудников и Помещения. При создании свойства объекта (object property) необходимо указать сущности, которые связывает этот предикат. В нашем случае связываются Сотрудники (это пункт Domain) и Помещения (Ranges). Сотрудники могут находится в помещениях, а помещения могут содержать сотрудников, свойство является функциональным и действует из домена Сотрудники в область Помещения. С учетом физического смысла один сотрудник может находиться ровно в одном помещении, этот факт тоже учитывается в определении свойства «находится_в_помещении»: этому свойству устанавливаются параметры Functional и Inverse Functional, получается бинарное отношение «один-к-одному».



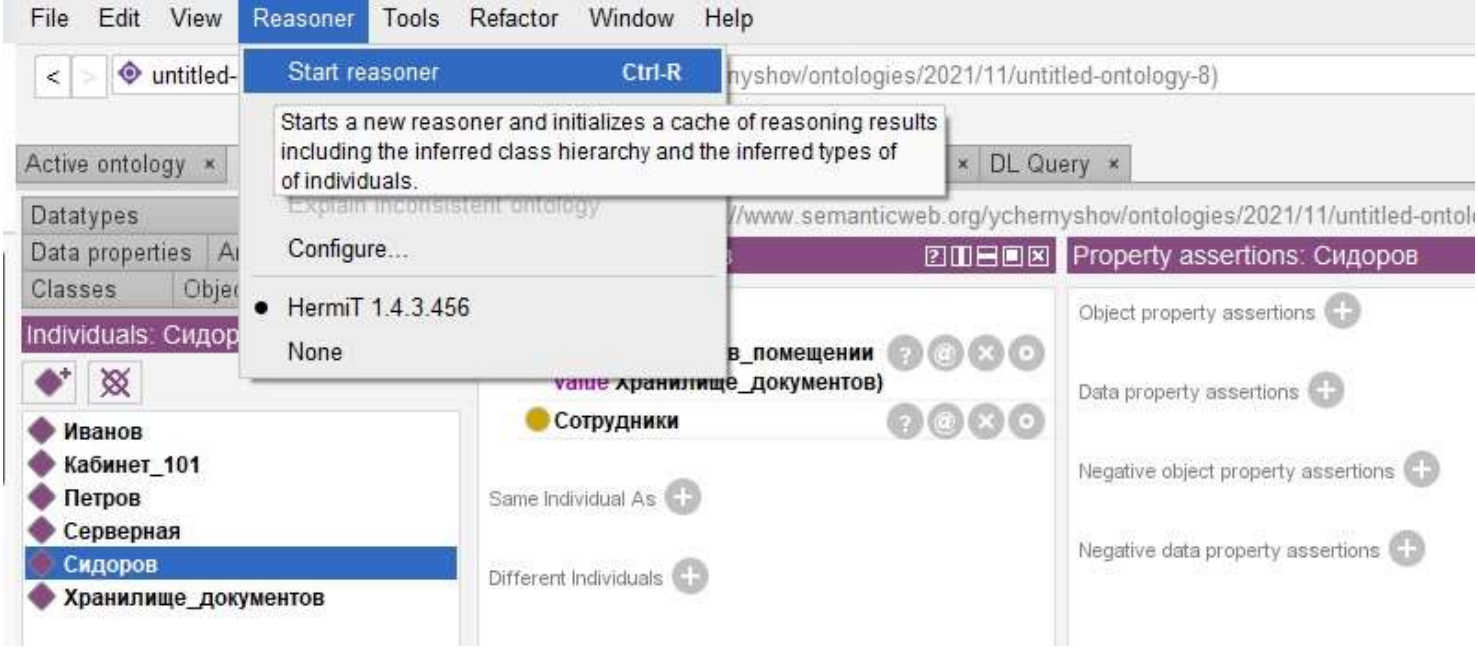


Теперь добавим в нашу онтологию информацию о том, что нам известно о местонахождении сотрудников.

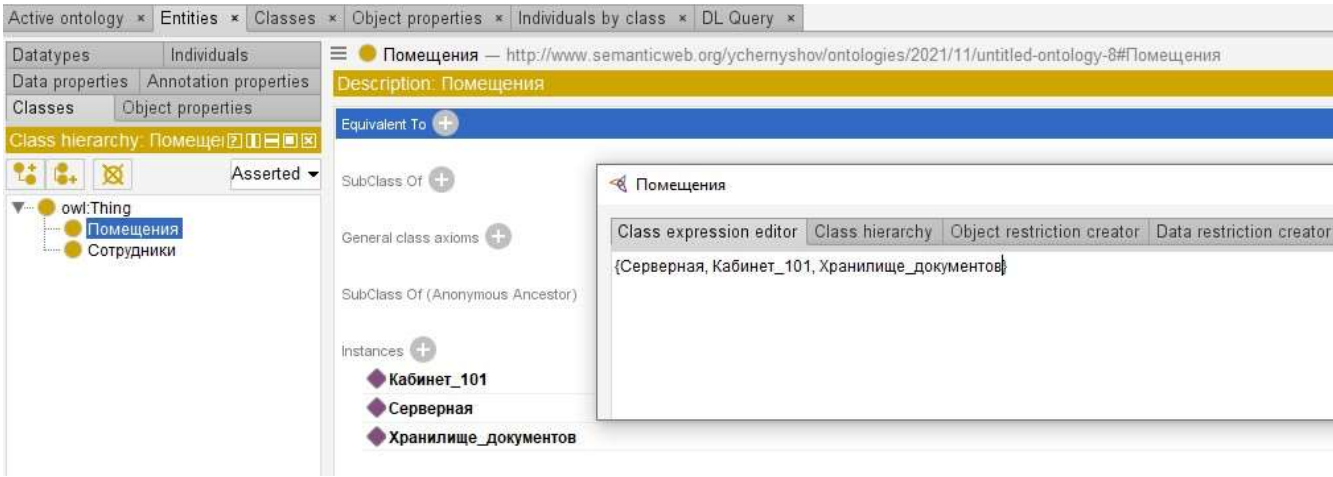


Аналогичную информацию добавляем о сотрудниках Петров и Сидоров, используя логические предикаты not и and.

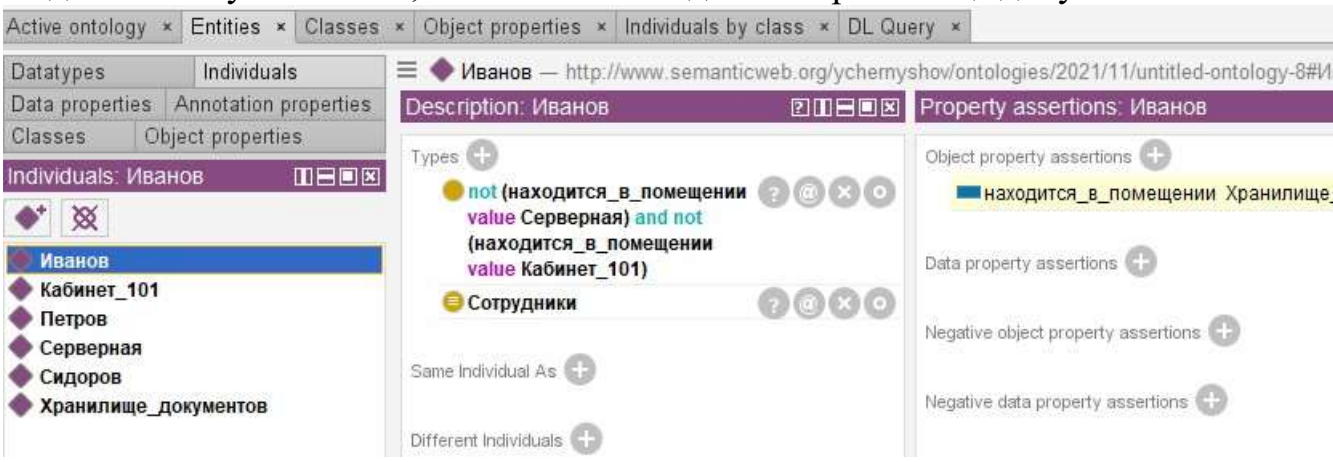
Мы можем попробовать запустить ризонер,



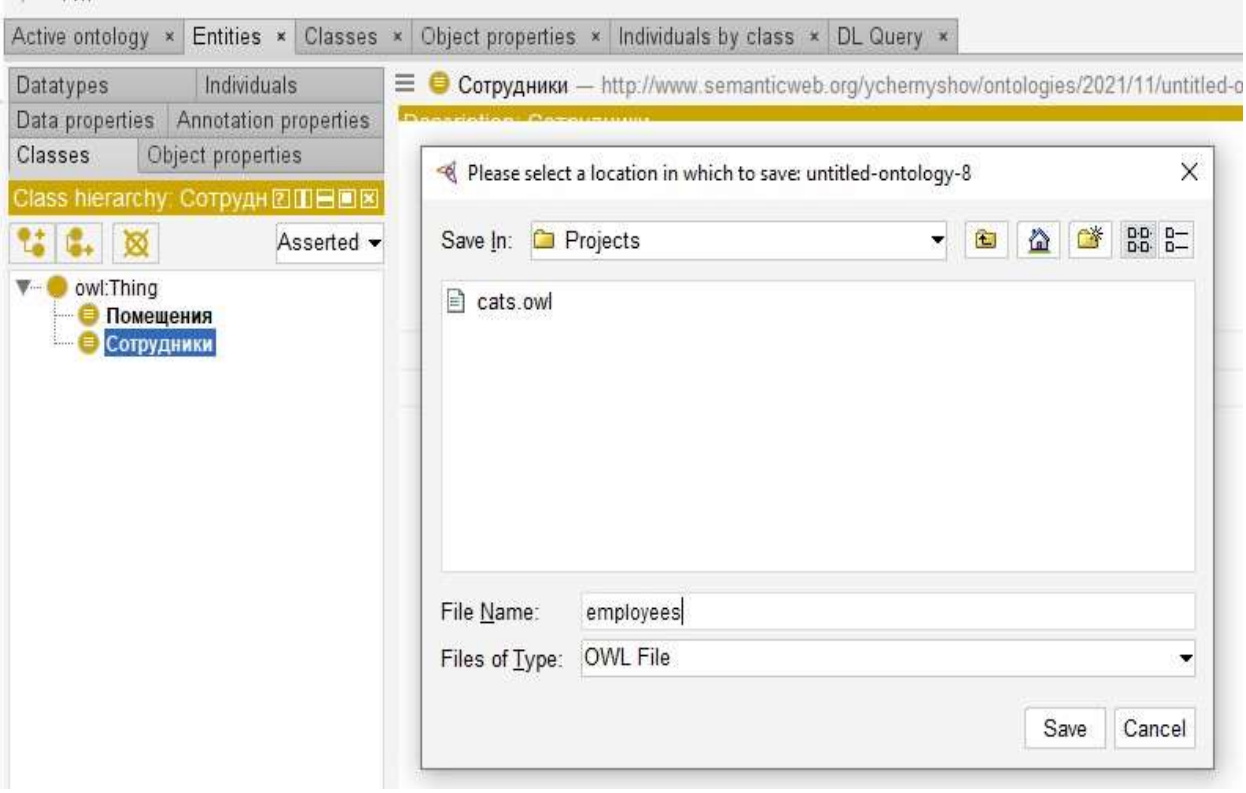
Для окончательного установления взаимно-однозначных отношений между сущностями нашей онтологии необходимо в явном виде описать тот факт, что каждый сотрудник обязательно должен быть хотя бы в одной комнате.



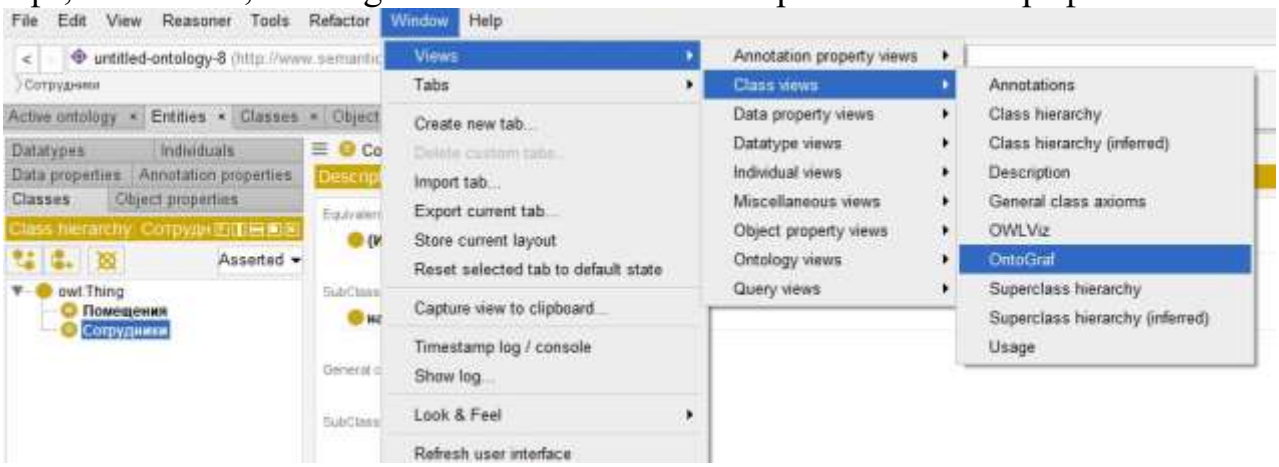
Запустив (или синхронизировав) ризонер, мы получим информацию о точном местонахождении сотрудников в конкретном помещении. Например, мы однозначно установили, что Иванов находится в хранилище документов.



Мы можем сохранить выбранную онтологию, например, в формате owl.



В результате получим owl-файл, этот файл можно открыть обычным текстовым редактором, изучить его содержание. Но конечно же гораздо удобнее воспринимать информацию в виде хорошо структурированного графа, к счастью, в Protégé есть возможность построения такого графа.



В результате мы получим хорошую визуализацию.

Конечно же, описанная задача очень проста и решается путем несложных логических рассуждений без всяких онтологий. Но давайте взглянем на более сложную задачу, которую немногие смогут решить в уме.

Задача Эйнштейна

Постановка задачи

Задача Эйнштейна – это логическая задача, заданная 15-ми утверждениями, из которых нужно найти ответы на вопросы, о том, кто пьёт воду и держит зебру.

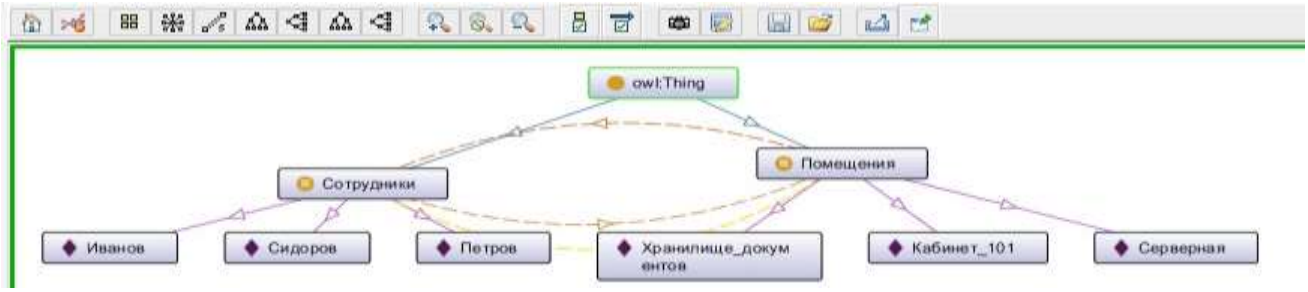
Оригинальный текст задачи выглядит следующим образом:

1. На улице стоят пять домов.
2. Англичанин живёт в красном доме.
3. У испанца есть собака.
4. В зелёном доме пьют кофе.
5. Украинец пьёт чай.

6. Зелёный дом стоит сразу справа от белого дома.
7. Тот, кто курит Old Gold, разводит улиток.
8. В жёлтом доме курят Kool.
9. В центральном доме пьют молоко.
10. Норвежец живёт в первом доме.
11. Сосед того, кто курит Chesterfield, держит лису.
12. В доме по соседству с тем, в котором держат лошадь, курят Kool.
13. Тот, кто курит Lucky Strike, пьёт апельсиновый сок.
14. Японец курит Parliament.
15. Норвежец живёт рядом с синим домом.

Кто пьёт воду? Кто держит зебру?

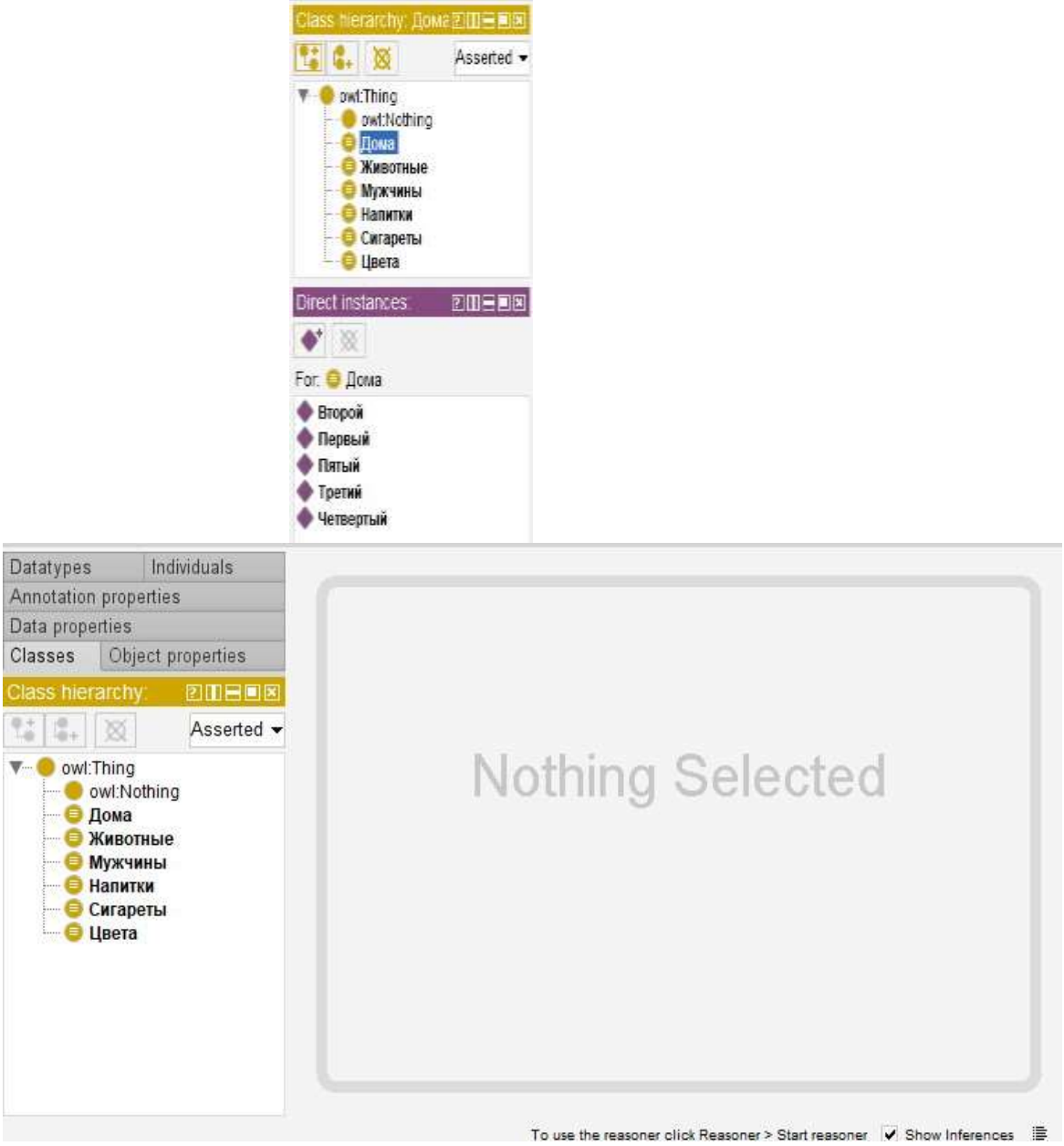
В целях ясности следует добавить, что каждый из пяти домов окрашен в свой цвет, а их жители — разных национальностей, владеют разными животными, пьют разные напитки и курят разные марки



американских сигарет. Ещё одно замечание: в утверждении 6 *справа* означает справа относительно *вас*.

Решение

На первом этапе определяем классы. Из утверждений можно понять, что их пять: «Дома», «Мужчины», «Животные», «Напитки», «Сигареты». Так как у класса «Дома» есть два признака – это номер дома и цвет, то выделим в отдельный класс цвет дома с названием «Цвета».

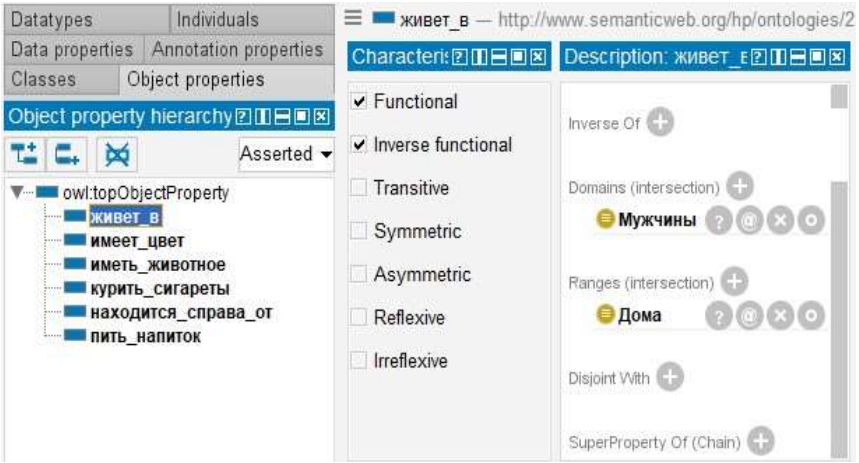


Затем создаем экземпляры классов (объекты)

Аналогичную процедуру проведем с оставшимися пятью классами.

Затем создадим свойства объектов и их характеристики. В разделе «Object Properties» создаем 6 свойств (предикатов): «жить_в», «иметь_цвет», «иметь_животное», «курить_сигареты»,

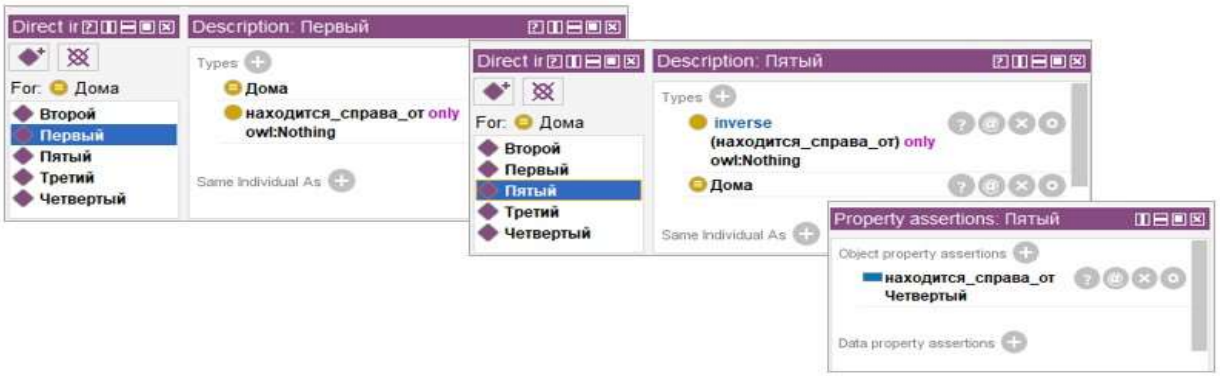
«находиться_справа_от», «пить_напиток». При создании этих предикатов указываем сущности, которые они будут связывать, и их характеристики.



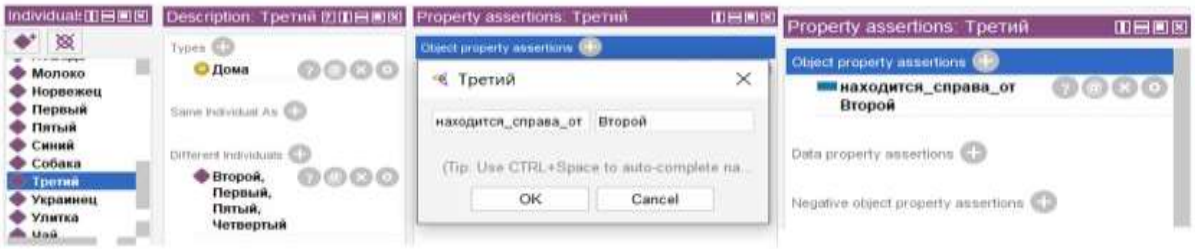
Аналогичную работу проделываем и с остальными предикатами.

Стоит отметить, что у всех шести свойств устанавливаются параметры Functional и Inverse functional, что означает бинарное отношение «один-к-одному».

Теперь добавляем в онтологию все известные факты о наших сущностях. Первым делом добавим информацию о том, что дома стоят по порядку с 1 по 5. Для этого у объекта «Первый» укажем, что он стоит справа от пустого множества, а у объекта «Пятый» укажем, что он стоит справа от четвертого дома и слева от пустого множества.



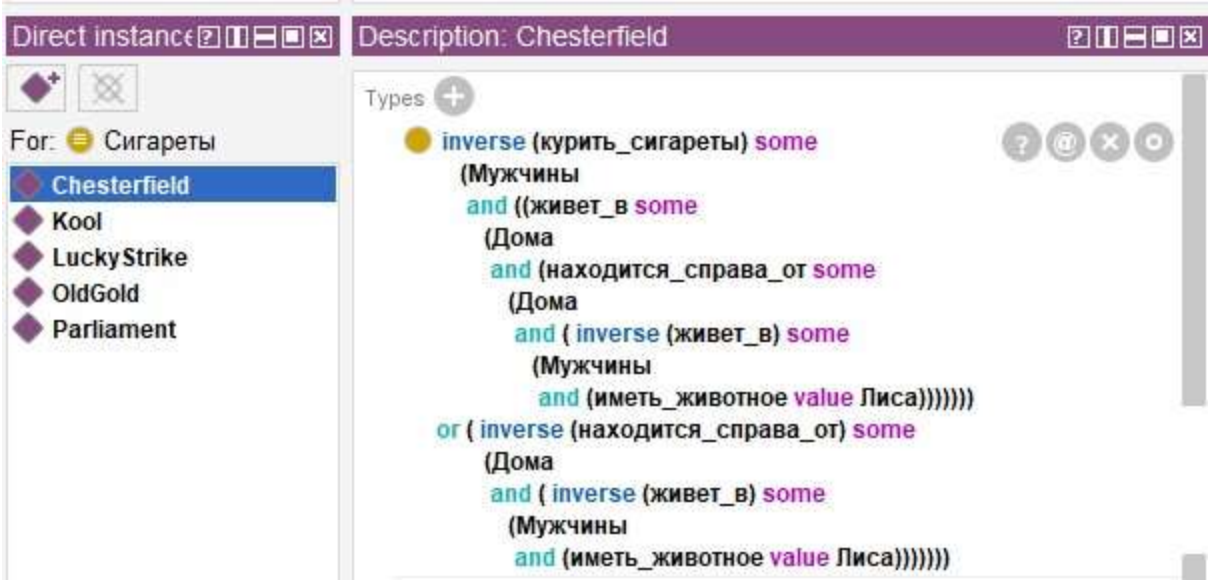
Для объектов «Второй», «Третий», «Четвертый» добавим информацию о том, что они находятся справа от предыдущего в разделе «Object property assertions».



Аналогичным способом добавляем информацию из более простых утверждений, которые уже представлены в виде триплетов. К примеру, выражение «Испанец держит собаку».

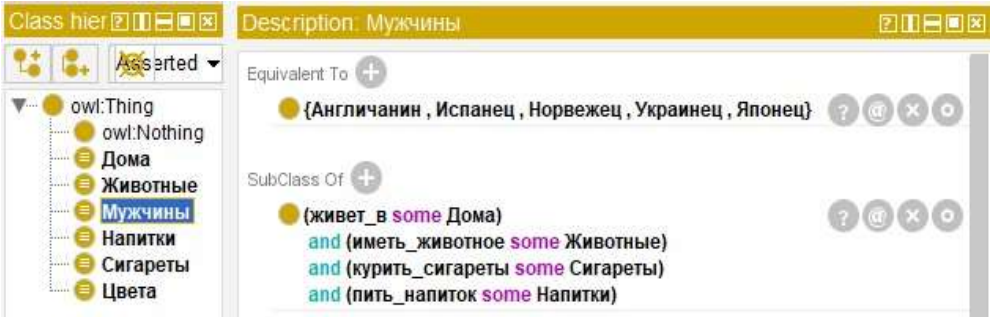
Далее вносим информацию из утверждений, из которых сложно сразу понять, кто в каком доме живет, что пьет, курит и какое животное держит. В данном случае мы будем использовать логические операции and и or.

В приведенном примере мы использовали служебное слово inverse для того, чтобы сделать марку сигарет субъектом высказывания, так как по логике выражения сигареты марки Chesterfield выкуриваются мужчиной, а не наоборот.

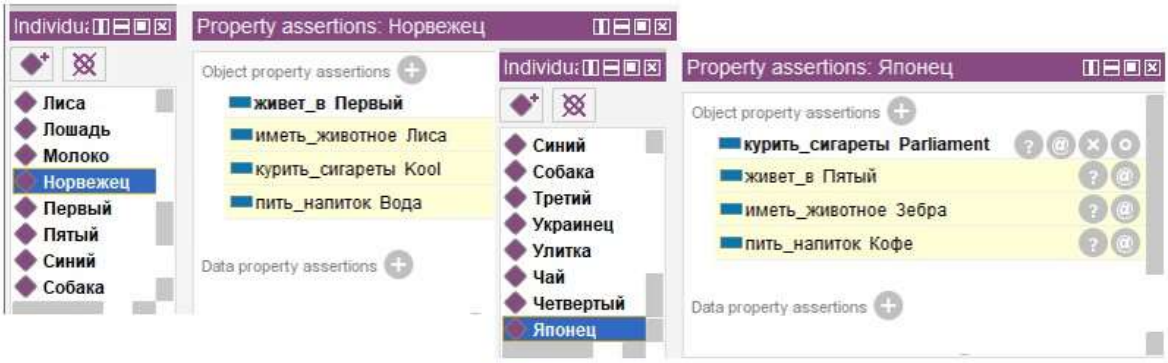


Аналогичным способом вносим информацию из остальных утверждений. Теперь попробуем запустить ризонер и видим, что после запуска ризонера каких-то существенных фактов извлечь не удастся, так как в текущей постановке у нас все классы открыты, а это значит, что сохраняется возможность добавления новых объектов. Чтобы ризонер смог однозначно разрешить задачу необходимо более строго определить классы.

Аналогичным способом строго определим остальные пять классов. Для окончательного определения нашей онтологии необходимо в явном виде описать тот факт, что каждый мужчина обязательно живет в каком-то доме, пьет какой-то напиток, держит какое-то животное и курит сигареты какой-то марки.



Запустив ризонер, мы получили ответы на поставленные вопросы. Теперь мы знаем, что Норвежец пьет воду, а Японец держит зебру.



Как видите, хорошо описанные онтологии могут пригодиться для решения непростых задач, однако для качественного решения задачи необходимо очень кропотливо описывать свойства элементов онтологии, сущностей и предикатов.

Замечание

Важно понимать, что далеко не всегда онтологии оказываются действенным инструментом для решения задачи, необходимо подходить к выбору способа решения с учетом особенностей предметной области.

В информационной безопасности уже созданы базы знаний, например MITRE ATT&CK и SHIELD, CVE, CAPEC. Эти базы знаний применяются при анализе инцидентов и реагировании на них, при проведении расследований и выявлении слабых мест в защите. О них мы поговорим в следующей части.

Создайте собственное описание области знаний, область практического приложения не важна, важна сложность связанности. Т.е. это не база данных, а некая структура, обладающая уникальными зависимостями для однотипных объектов или уникальными связями для схожих объектов. Поэтому сначала продумайте описание (можно проконсультироваться с преподавателем) и после создайте свою онтологию, сохранив результат в виде графического представления в отчете и файла own.