Praktyczny przewodnik tworzenia ontologii w języku OWL za pomocą Protégé 5.1

Wykonany na podstawie

Protégé OWL Tutorial. A step-by-step guide to modelling in OWL using the popular Protégé OWL tools. Matthew Horridge i inni dla projektu CO-ODE.

(http://130.88.198.11/tutorials/protegeowltutorial/)

przez

Tomasza Dżumagę, Błażeja Kącikowskiego, Łukasza Mocka (2012/2013)

i Agnieszkę Ławrynowicz, 28.10.2013, 2.11.2014, 4.12.2016

1. Tworzenie ontologii w języku OWL.

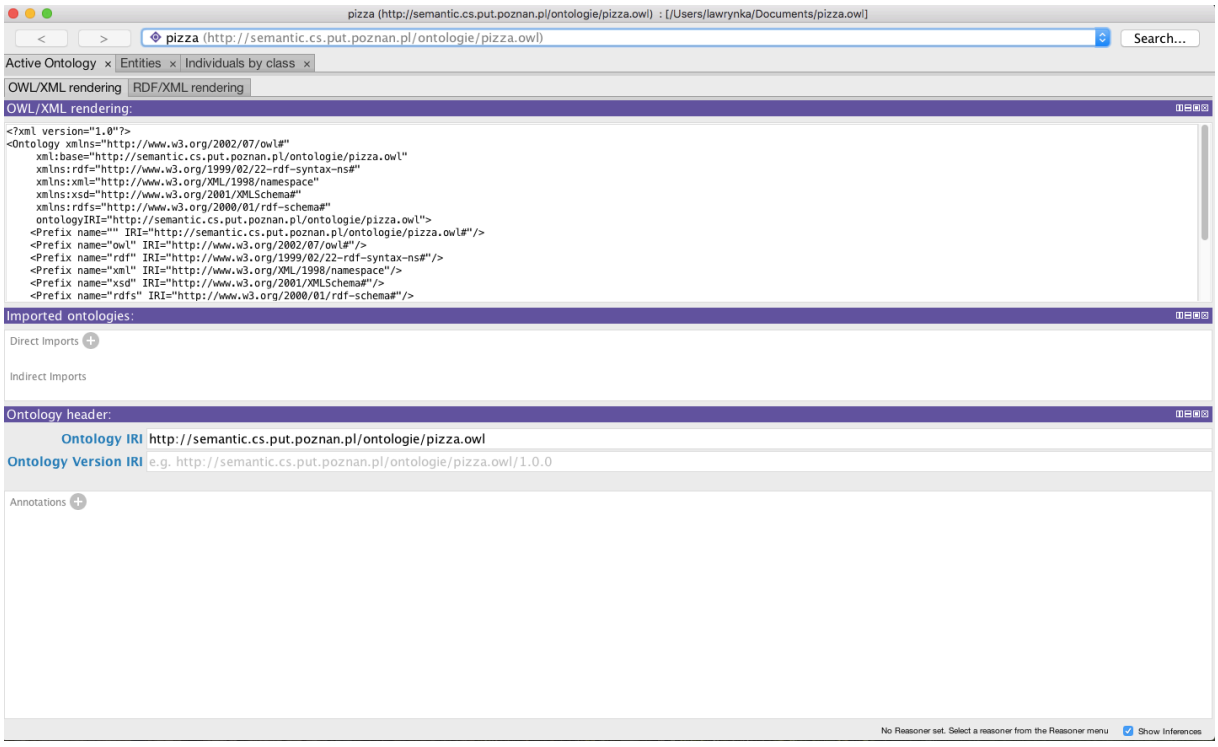
W niniejszym tutorialu będziemy tworzyć ontologię pizzy.

Ćwiczenie 1. Tworzenie nowej ontologii.

---

1. Uruchom Protégé.
2. Każdej ontologii zostaje przydzielony identyfikator za pomocą *Internationalized Resource Identifier (IRI).* Zastąp domyślnie wygenerowany IRI w polu *Ontology IRI* poprzez *http://semantic.cs.put.poznan.pl/ontologie/pizza.owl*. Zapisz ontologię w formacie *Turtle.*

Jak widać na Rysunku 1, zakładka *Active Ontology* dostarcza informacji na temat określonej ontologii. Za jej pomocą można na przykład zmienić IRI ontologii, dodać lub edytować adnotacje dotyczące ontologii, takie jak np. komentarze, informacje o twórcy, wersji itp., można także za pomocą tej zakładki ustalić przestrzenie nazw i importy innych ontologii.

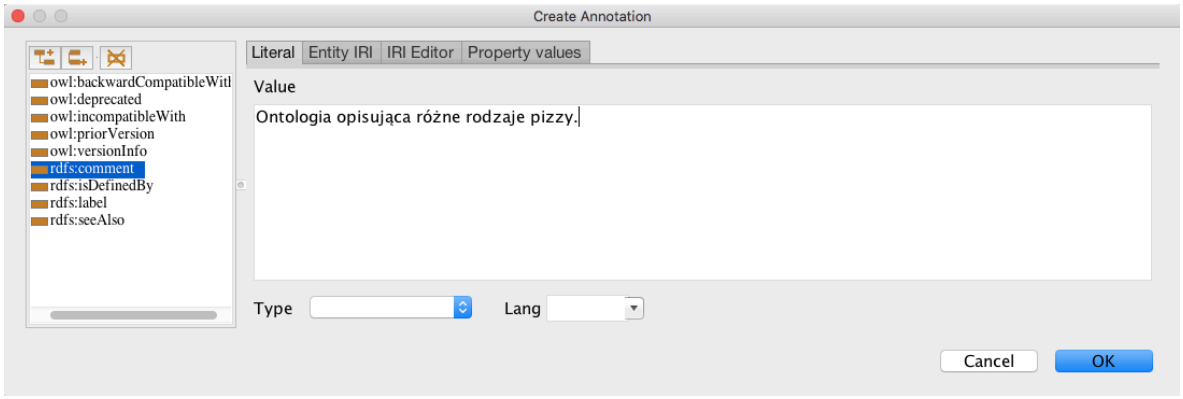


Rysunek 1.

Ćwiczenie 2. Dodawanie komentarza do ontologii.

---

1. Upewnij się, że zakładka *Active Ontology* jest aktywna.
2. W widoku *Annotations* kliknij ikonę *Add* ( ) obok napisu *Annotations*. W efekcie pojawi się okno edycji. Z menu po lewej stronie wybierz *comment* i w polu z prawej strony wpisz dowolny tekst.
3. Kliknij *OK*. W tym momencie widok adnotacji powinien wyglądać podobnie jak na Rysunku 2.



Rysunek 2.

2. Nazwane klasy.

Ontologia zawiera klasy, są one głównymi elementami ontologii w języku OWL. W Protégé 5.1, edycja klas jest obsługiwana poprzez zakładkę *Class hierarchy*, przedstawioną na Rysunku 3. Początkowe drzewo hierarchii klas powinno przypominać to, które zostało przedstawione na Rysunku 4. Pusta ontologia zawiera tylko jedną klasę nazwaną *Thing*. Klasy OWL są interpretowane jako zbiory *indywiduów* (instancji). Klasa *Thing* reprezentuje zbiór zawierający wszystkie indywidua. Z tego powodu wszystkie klasy są podklasami klasy *Thing*.

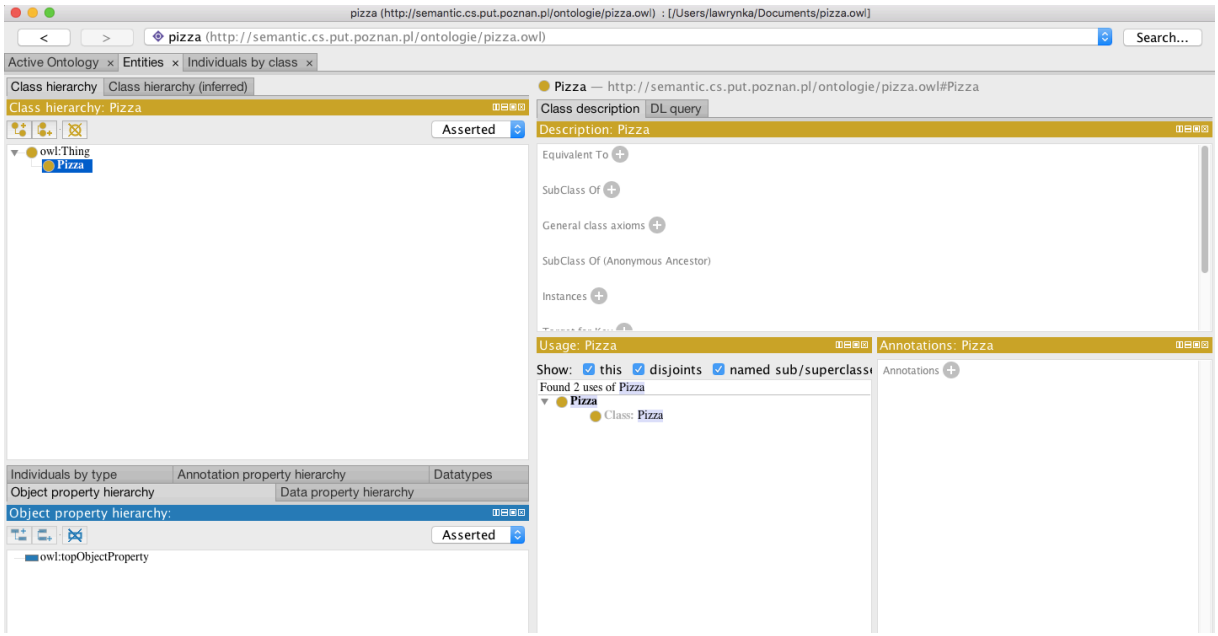
Ćwiczenie 3. Tworzenie klas Pizza, Ciasto i Dodatek.

---

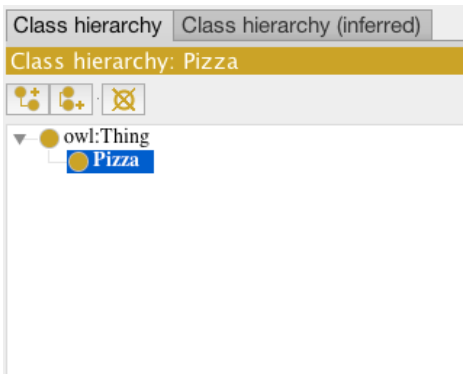
1. Upewnij się, że zakładka *Class hierarchy* jest aktywna.
2. Kliknij na klasę *Thing*, a następnie na ikonę *Dodaj podklasę*, zaznaczoną na Rysunku 4. Ten przycisk tworzy nową klasę jako podklasę zaznaczonej klasy (w tym przypadku chcemy stworzyć podklasę klasy *Thing)*.

3. Pojawi się okienko, w którym należy podać nazwę klasy. Wprowadź tekst *Pizza (tak jak przedstawiono na rysunku 5)* i naciśnij *OK.*
4. Powtórz powyższe kroki, aby dodać klasy Ciasto i Dodatek, upewniając się, że klasa *Thing* jest zaznaczona, zanim klikniesz na ikonę *Dodaj podklasę*, dzięki czemu te klasy zostaną stworzone jako podklasy *Thing*.

Hierarchia klas powinna przypominać hierarchię, widoczną na rysunku 6.



Rysunek 3.



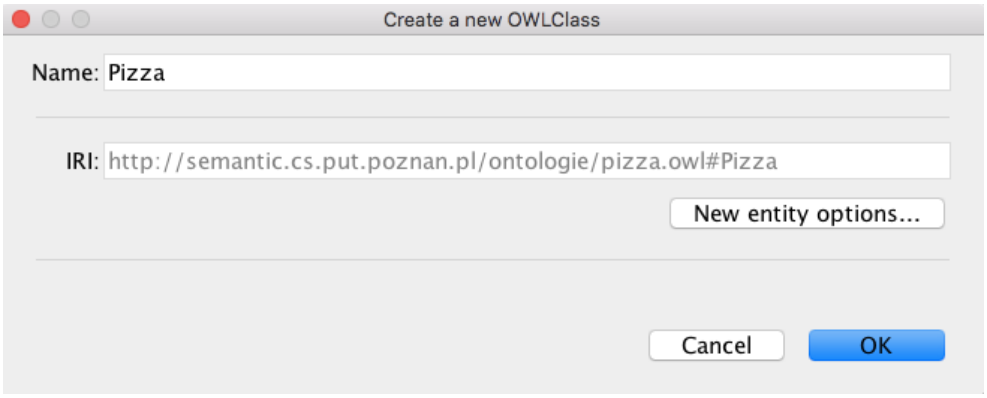
Legenda:

Dodaj podklasę

Dodaj klasę sąsiadującą

Usuń klasę

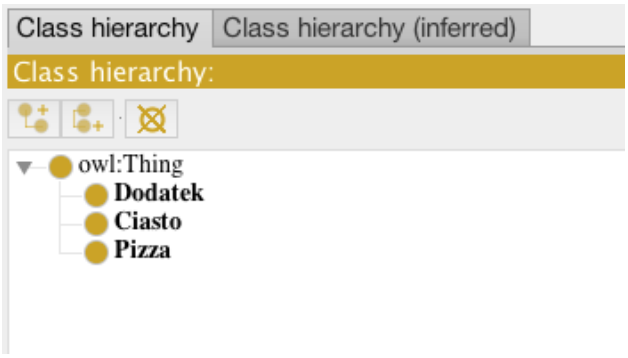
Rysunek 4.



Rysunek 5.

Po stworzeniu klasy Pizza, zamiast ponownie wybierać klasę *Thing* i używać przycisku *Dodaj podklasę* do stworzenia klas *Ciasto* i *Dodatek* jako podklasy *Thing*, można użyć przycisku *Dodaj klasę sąsiadującą* (pokazanego na rysunku 4). W tym celu należy zaznaczyć klasę *Pizza* i kliknąć wspomniany przycisk.

Hierarchia klas może być także nazywana taksonomią.



Rysunek 6.

3. Klasy rozłączne.

Mając dodane do naszej ontologii klasy *Pizza, Ciasto, Dodatek*, możemy teraz zamodelować, że są to klasy rozłączne, zatem każde indywiduum może być instancją tylko jednej z tych klas. Aby określić, które klasy są rozłączne należy kliknąć przycisk *Disjoint With* znajdujący się na dole widoku *Description* na zakładce *Class hierarchy.*

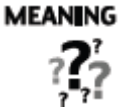
Ćwiczenie 4. Modelowanie rozłączności klas Pizza, Ciasto i Dodatek.

---

1. Zaznacz klasę *Pizza* na drzewie hierarchii klas.

2. Kliknij przycisk *Disjoint With.* Spowoduje to pojawienie się nowego okna, w którym należy zaznaczyć wszystkie klasy i kliknąć *OK*. Spowoduje to, że klasy *Ciasto* i *Dodatek*, będą rozłączne z klasą *Pizza.*

Zauważ, że w widoku *Description*, pod przyciskiem *Disjoint With*, pojawiły się nazwy klas *Ciasto* i *Dodatek*. Po kliknięciu na klasę *Ciasto*, w tym samym miejscu wypisane będą klasy *Pizza* i *Dodatek*, które są rozłączne z klasą *Ciasto*.



Klasy OWL z założenia mogą mieć swoje części wspólne. Nie możemy też założyć, że dana instancja nie jest członkiem konkretnej klasy tylko dlatego, że nie została do niej jednoznacznie przypisana. To uzyskujemy poprzez jawne wyrażenie, że grupa klas jest wzajemnie rozłączna. W ten sposób można zagwarantować, że żadna instancja należąca do klasy w danej grupie, nie może także należeć do innej klasy w tej grupie.
W naszym przykładzie klasy *Pizza, Ciasto, Dodatek* są wzajemnie rozłączne. Nie ma możliwości, aby dane indywiduum było członkiem każdej z tych klas – nie miało by sensu, by ciasto do pizzy było jednocześnie jej obkładem.

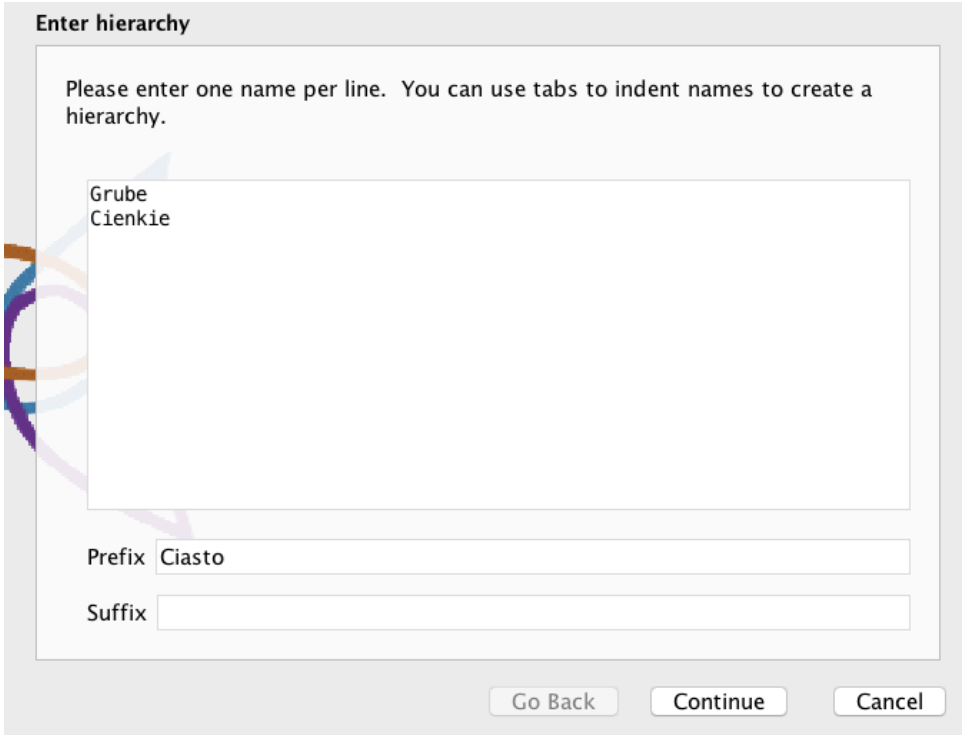
4. Używanie narzędzia *Create class hierarchy* do tworzenia klas.

W tym rozdziale pokażemy, jak użyć narzędzia *Create class hierarchy* do dodania kilku podklas do klasy *Ciasto.*

Ćwiczenie 5. Użycie narzędzia *Create class hierarchy* do stworzenia klas *CiastoGrube* i *CiastoCienkie* jako podklas klasy *Ciasto.*

---

1. Zaznacz klasę *Ciasto* z drzewa hierarchii klas.
2. Na pasku menu programu Protégé, wybierz *Tools*, a następnie *Create class hierarchy.*
3. Pojawi się okno widoczne na Rysunku 7. Musimy poinformować narzędzie o tym jakie podklasy klasy *Ciasto* chcemy stworzyć. W dużym pustym polu wpisz *Cienkie* i wciśnij *Enter*. Następne wpisz *Grube*, a w polu *Prefix* wpisz *Ciasto*. Całość powinna wyglądać tak, jak na Rysunku 7.
4. Naciśnij *Continue.* Protégé sprawdza, czy nazwy klas są unikalne oraz, czy nie zawierają spacji. Jeśli byłyby jakieś błędy, to w tym momencie powinien pojawić się stosowny komunikat.
5. Upewnij się, że opcja *Make sibling classess disjoint* jest zaznaczona i naciśnij *Finish.* W ten sposób utworzone klasy będą klasami rozłącznymi i nie będzie trzeba robić tego ręcznie.



Rysunek 7.

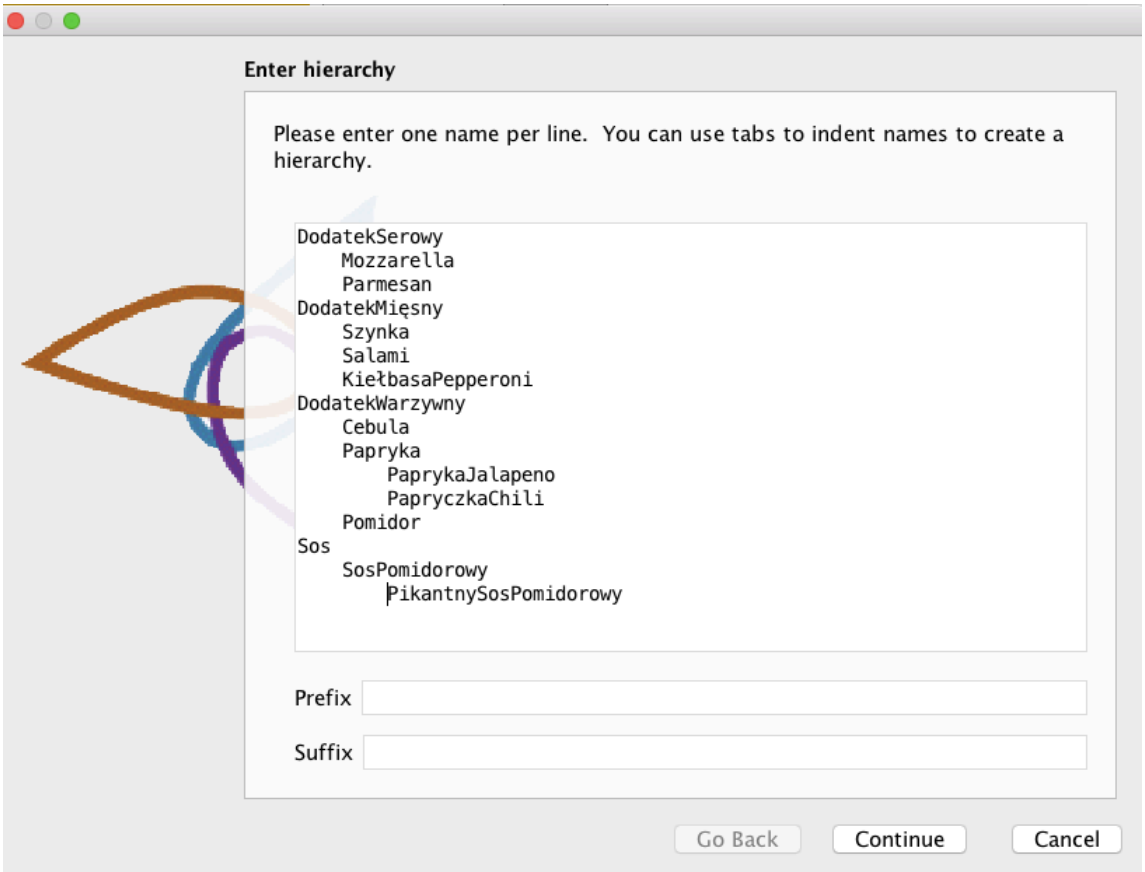
Po kliknięciu przycisku *Finish*, Protégé utworzy nam rozłączne klasy. Ontologia powinna teraz zawierać *CiastoCienkie* i *CiastoGrube* jako podklasy klasy *Ciasto*.

Mając kilka podstawowych klas, możemy stworzyć klasy reprezentujące obkłady pizzy, pogrupowane w różne kategorie: obkłady mięsne, obkłady warzywne, obkłady serowe itd.

Ćwiczenie 6. Tworzenie podklas klasy *Dodatek*.

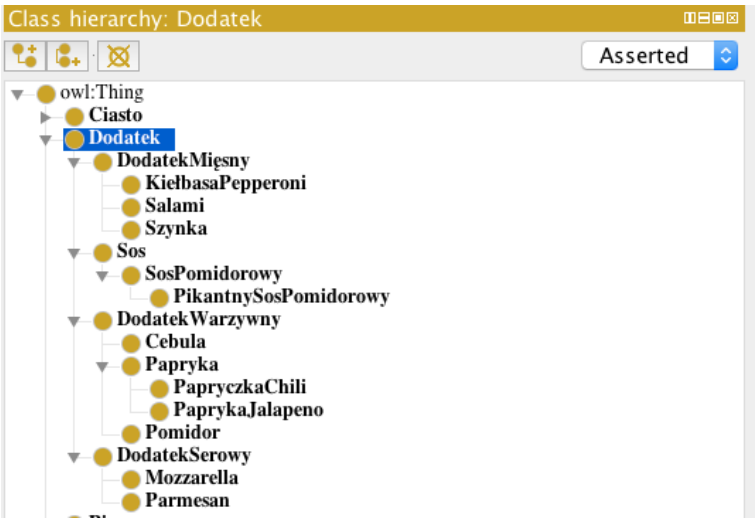
---

1. Zaznacz klasę *Dodatek* z drzewa hierarchii klas.
2. Uruchom *Create class hierarchy* tak jak w poprzednim punkcie.
3. Upewnij się, że *Dodatek* jest zaznaczone i kliknij *Continue*.
4. Istnieje możliwość wprowadzenia odpowiedniej hierarchii nowych klas już na tym etapie. Odbywa się to za pomocą tabulatora. Wprowadź nazwy klas zgodnie z Rysunkiem 8.
5. Po wprowadzeniu wszystkich nazw klas, naciśnij *Continue*. Upewnij się, że opcja *Make sibling classess disjoint* jest zaznaczona, aby wszystkie tworzone klasy były rozłączne.
6. Kliknij *Finish*, aby utworzyć klasy.



Rysunek 8.

Hierarchia klas dotycząca dodatków powinna teraz wyglądać tak jak na rysunku 9.



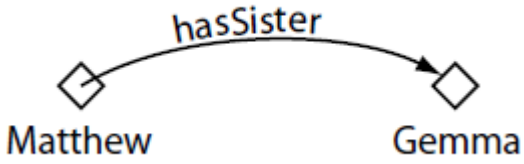
Rysunek 9.

Oczywiście wszystkie elementy należące do klasy *PapryczkaChili*, należą także do klas *Papryka* oraz *DodatekWarzywny*.

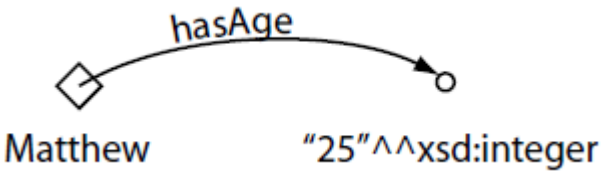
W tym momencie mamy już za sobą tworzenie kilku nazwanych klas, z których niektóre są podklasami innych klas. Konstrukcja hierarchii klas może wydawać się intuicyjna, jednakże, czy wiemy co naprawdę w języku OWL oznacza, że coś jest podklasą innej klasy? Np. co to znaczy, że *DodatekMięsny* jest podklasą klasy *Dodatek*. W OWLu relacja *podklasy* oznacza *implikację*. Inaczej mówiąc, jeżeli *DodatekMięsny* jest podklasą klasy *Dodatek*, to wszystkie instancje klasy *DodatekMięsny* są instancjami klasy *Dodatek.*

5. Własności OWL.

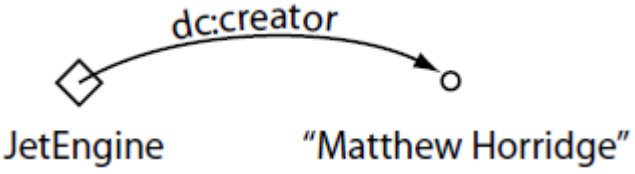
Własności OWL reprezentują relacje. Istnieją dwa główne typy własności, *obiektowe* i *literałowe*. Własności obiektowe reprezentują relacje pomiędzy dwoma obiektami. W tym rozdziale skupimy się na własnościach obiektowych, własności literałowe są przedstawione w kolejnym rozdziale. Własności obiektowe łączą obiekty między sobą. OWL pozwala także na wprowadzenie trzeciego typu własności: własności *adnotacyjnych*, które mogą być używane do dodania informacji w postaci notatek do klas, obiektów i ich własności. Na Rysunku 10 przedstawione zostały przykłady każdego typu własności.



Własność obiektowa łącząca obiekt Matthew z obiektem Gemma.



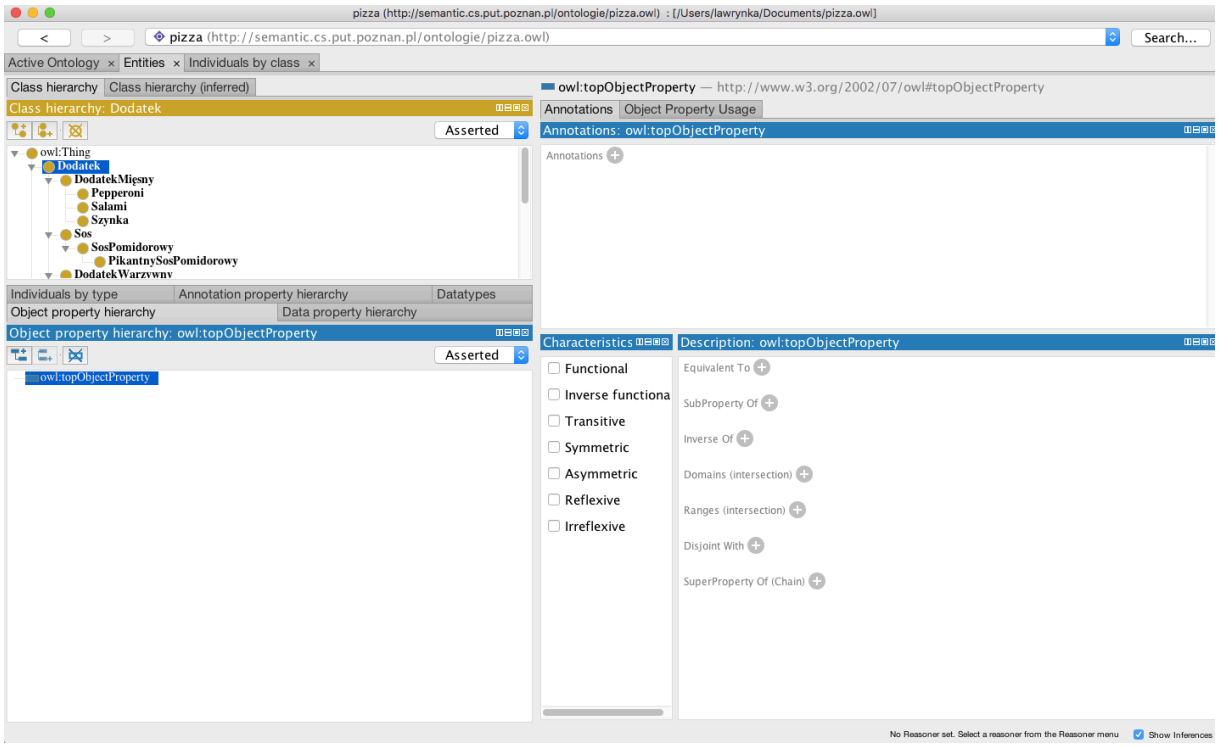
Własność literałowa łącząca obiekt Matthew
z wartością „25”, będącą typu xsd:integer



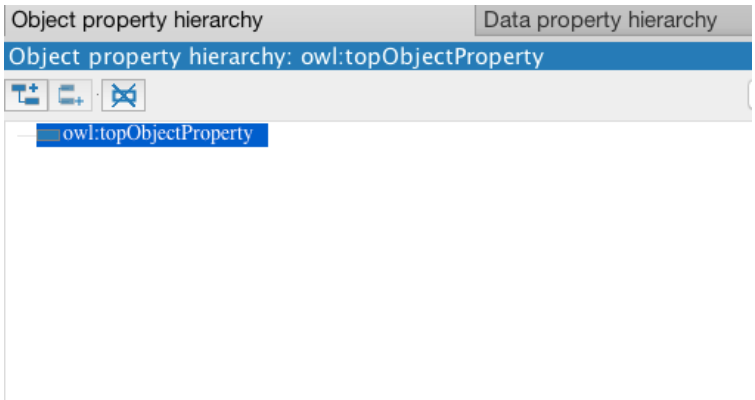
Własność adnotacyjna łącząca klasę JetEngine
z wartością „Matthew Horridge”

Rysunek 10.

Własności mogą być tworzone za pomocą zakładki *Object property hierarchy* widocznej na Rysunku 11. Rysunek 12 przedstawia przyciski znajdujące się w lewym górnym narożniku zakładki *Object property hierarchy*, które są używane do tworzenia własności OWL. Za pomocą tych przycisków można tworzyć wszystkie rodzaje własności. Większość własności tworzonych w tym tutorialu będzie własnościami obiektowymi.



Rysunek 11.



Legenda:

 Dodaj podwłasność

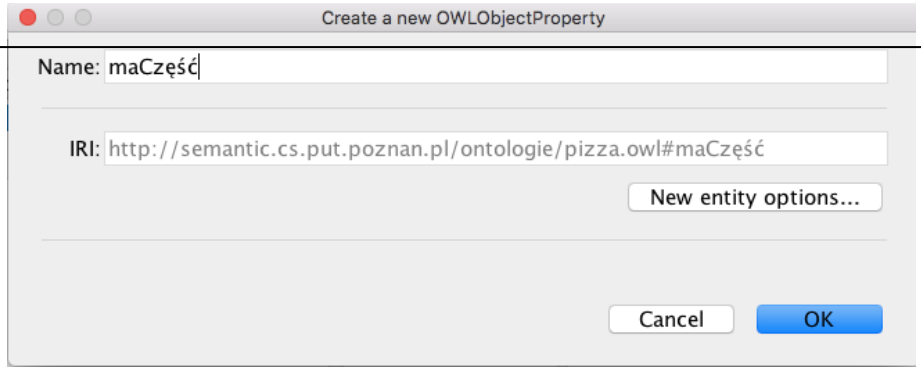
 Dodaj własność sąsiadującą

 Usuń własność

Rysunek 12.

Ćwiczenie 7. Tworzenie własności obiektowej *maCzęść.*

1. Wybierz zakładkę *Object property hierarchy.* Kliknij główną własność *topObjectProperty*, a następnie użyj przycisku *Add sub property* do stworzenia nowej własności.
2. Nadaj nazwę własności *maCzęść*, wykorzystując okienko dialogowe, takie jak na Rysunku 13.



Rysunek 13.

Mając już dodaną własność *maCzęść*, możemy dodać kolejne dwie własności: *maCiasto* i *maDodatek.* W OWLu, własności mogą mieć podwłasności, więc możliwe jest tworzenie ich hierarchii. Podwłasności są bardziej szczegółowe od swoich nadrzędnych własności (w taki sam sposób, jak podklasy uszczegóławiają nadrzędne klasy). Na przykład własność *maMatkę* może specjalizować własność *maRodzica*. W ten sposób dochodzimy do wniosku, że w naszej ontologii własności *maCiasto* i *maDodatek* powinny być stworzone jako podwłasności *maCzęść*. Jeżeli np. *maCiasto* będzie łączyć dwa różne obiekty, to oznacza, że te dwa obiekty są także połączone własnością *maCzęść*.

Ćwiczenie 8. Tworzenie *maCiasto* i *maDodatek* jako podwłasności *maCzęść.*

1. Aby stworzyć własność *maCiasto* jako podwłasność *maCzęść*, zaznacz *maCzęść* na drzewie hierarchii własności.
2. Kliknij przycisk *Add sub property.*
3. Nazwij nową własność jako *maCiasto*.
4. Powtórz powyższe kroki nazywając kolejną nową własność jako *maDodatek.*

Zauważ, że jest także możliwe stworzenie podwłasności typu literałowego. Jednak nie jest możliwe mieszanie własności obiektowych i własności literałowych, czyli np. próba stworzenia własności obiektowej jako podwłasności literałowej nie zostanie pomyślnie zakończona.

6. Dziedziny i przeciwdziedziny własności.

Własności mogą mieć swoje dziedziny i przeciwdziedziny. Własności łączą obiekty z dziedziny z obiektami z przeciwdziedziny. Przykładowo w naszej ontologii własność *maDodatek* będzie prawdopodobnie łączyć obiekty należące do klasy *Pizza* z obiektami należącymi do klasy *Dodatek*. W tym przykładzie dziedzina własności *maDodatek* to *Pizza* a zakres to *Dodatek*.

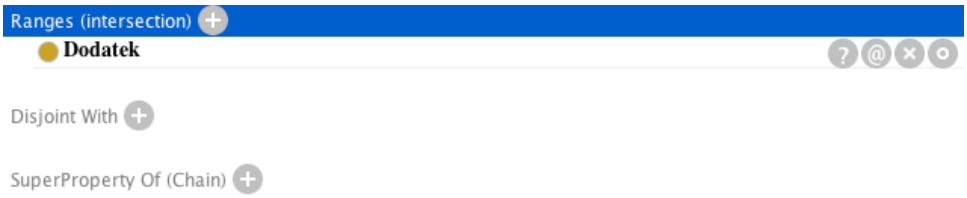
Należy sobie uświadomić, że w języku OWL dziedziny i przeciwdziedziny nie powinny być postrzegane jako ograniczenia. Jeśli własność *maDodatek* ma dziedzinę *Pierogi* i zastosujemy tę własność do klasy *Lody* (instancje będące członkami klasy *Lody*), to nie powinno powodować żadnego błędu. Zamiast tego, zostanie to wykorzystane do wywnioskowania, że klasa *Lody* powinna być podklasą klasy *Pizza*.

Chcemy teraz ustalić przeciwdziedzinę własności *maDodatek* jako *Dodatek*. Aby to zrobić należy wykorzystać widok przedstawiony na Rysunku 14.

Ćwiczenie 9. Sprecyzowanie zakresu własności *maDodatek*.

---

1. Upewnij się, że własność *maDodatek* jest zaznaczona w hierarchii własności.
2. Kliknij *Add* (⊕) obok napisu *Ranges* w polu *Description* (Rysunek 14), następnie wybierz zakładkę *class hierarchy*. Pojawi się okienko umożliwiające wybór klasy z naszej hierarchii klas.
3. Wybierz *Dodatek* i naciśnij *OK*. Klasa *Dodatek* powinna być teraz wyświetlana na liście przeciwdziedzin (*Ranges*) własności.



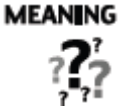
Rysunek 14.

Możliwe jest ustalenie większej liczby klas w przeciwdziedzinie własności. Jeśli więcej klas jest wyspecyfikowanych w Protégé 5.1, przeciwdziedzina własności jest interpretowana jako część wspólna (iloczyn) klas.

Ćwiczenie 10. Ustawienie klasy *Pizza* jako dziedziny własności *maDodatek*.

---

1. Upewnij się, że własność *maDodatek* jest wybrana w hierarchii własności.
2. Kliknij *Add* (⊕) obok napisu *Domains* w polu *Description*. Pojawi się okienko pozwalające na wybór odpowiedniej klasy.
3. Wybierz *Pizza* i kliknij *OK.* Klasa *Pizza* powinna być teraz wyświetlana na liście dziedziny.



To oznacza, że indywidua znajdujące się „z lewej strony" własności *maDodatek* będą wywiedzione jako indywidua klasy *Pizza*. Dowolny obiekt będący „po prawej stronie" będzie wywiedziony jako indywiduum należące do klasy *Dodatek*. Na przykład, jeśli mamy obiekty *a* i *b* oraz asercję *a maDodatek b*, to wniosek będzie taki, że *a* jest członkiem klasy *Pizza* i *b* jest członkiem klasy *Dodatek*.

Ćwiczenie 11. Ustawienie dziedziny i przeciwdziedziny dla własności *maCiasto*.

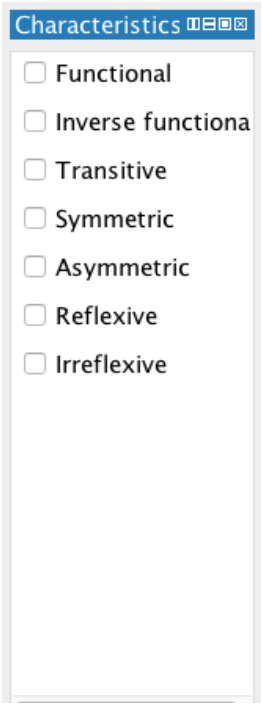
---

1. Wybierz własność *maCiasto*.
2. Ustaw dziedzinę tej własności na *Pizza.*
3. Ustaw przeciwdziedzinę własności na *Ciasto.*

W tym tutorialu ustawialiśmy dziedziny i przeciwdziedziny różnych własności, jednakże w rzeczywistości nie zawsze doradza się wykonywanie tych czynności. Fakt, że dziedziny i przeciwdziedziny nie zachowują się jak ograniczenia i fakt, że mogą powodować niespodziewane wyniki klasyfikacji mogą prowadzić do problemów i nieoczekiwanych skutków ubocznych, a one mogą być niekiedy trudne do znalezienia i poprawienia w dużej ontologii.

7. Wybrana charakterystyka obiektowych własności OWL.

OWL pozwala na wzbogacenie znaczenia własności poprzez użycie ich charakterystyki (Rysunek 15).

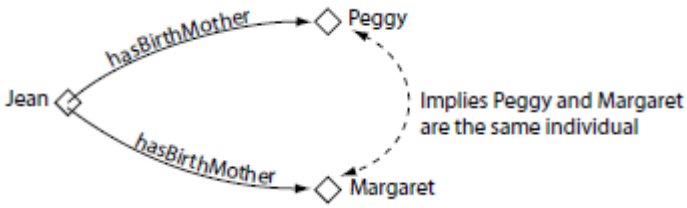


Rysunek 15.

Poniższe podrozdziały przedstawiają wybrane charakterystyki własności.

7.1 Własność funkcyjna

Jeśli własność jest własnością funkcyjną dla danego obiektu, to może istnieć co najwyżej jeden obiekt, z którym może być powiązany poprzez tę własność. Rysunek 16 przedstawia przykład własności funkcyjnej *hasBirthMother* – można mieć tylko jedną biologiczną matkę. Jeśli występuje asercja *Jean hasBirthMother Peggy*, i dodatkowo *Jean hasBirthMother Margaret*, to, ponieważ *hasBirthMother* jest własnością funkcyjną, możemy wywnioskować, że *Peggy* i *Margaret* musi być tym samym obiektem. W przeciwnym wypadku powyższe stwierdzenia byłyby ze sobą niezgodne.

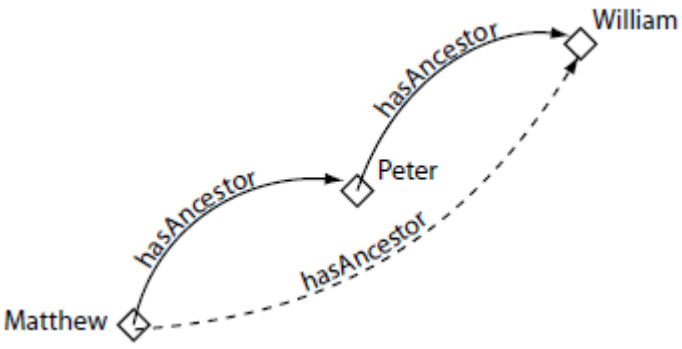


Rysunek 16.

7.2 Własności przechodnie.

Jeżeli własność jest przechodnia i łączy ona obiekt *a* z obiektem *b*, a także obiekt *b* z obiektem *c*, to można powiedzieć, że własność ta łączy obiekt *a* z obiektem *c*. Przykład pokazany jest na Rysunku 17. Jeżeli obiekt *Matthew* ma

przodka *Peter*, a on z kolei ma przodka *William*, to jest on także przodkiem *Matthew*.



Rysunek 17.

Ćwiczenie 12. Oznaczenie *maCzęść* jako własność przechodnią.

---

1. Wybierz własność *maCzęść* z hierarchii własności.
2. Na widoku *Characteristic* zaznacz *Transitive*.

Jeżeli własność jest przechodnia, to nie może być funkcyjna.

Chcemy także, aby każda nasza pizza miała tylko jeden rodzaj ciasta. Zrobimy to poprzez zmianę własności *maCiasto* na funkcyjną, dzięki czemu będzie mogła mieć tylko jedną wartość dla danego obiektu.

Ćwiczenie 13. Oznaczenie *maCiasto* jako własność funkcyjną.

---

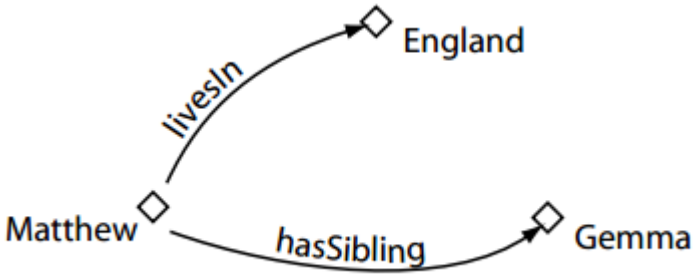
1. Wybierz własność *maCiasto* z hierarchii własności.
2. Na widoku *Characteristic* zaznacz *Functional*.

8. Opisywanie i definiowanie klas.

Mając stworzone własności możemy teraz użyć ich do definiowania i opisu klas naszej ontologii.

8.1 Ograniczenia właściwości

Jak pewnie wiesz w OWLu własności opisują relacje binarne. Własności literałowe (datatype) opisują relacje pomiędzy obiektami a danymi. Własności obiektowe opisują relację pomiędzy dwoma obiektami. Rysunek 18 przedstawia dwie przykładowe własności; obiekt *Matthew* jest połączony z obiektem *Gemma* przez własność *hasSibling*. Możemy myśleć o takich obiektach jako o klasie obiektów które posiadają relację *hasSibling.* Główną ideą jest to, że klasy obiektów są opisywane/definiowane przez relacje, w których są one stronami. W OWL możemy definiować takie klasy używając ograniczeń.



Rysunek 18.

Ograniczenia opisują klasy obiektów na podstawie relacji, w jakich są obiekty będące częścią danej klasy. Innymi słowy *ograniczenie* jest rodzajem klasy, tak samo jak i klasy nazwane.

Przykłady ograniczeń

Spójrzmy na kilka przykładów które pomogą nam wyjaśnić jakiego rodzaju klasy obiektów moglibyśmy opisywać na podstawie ich własności.

- Klasę obiektów, które są w przynajmniej jednej relacji *hasSibling*,
- Klasę obiektów, które są w przynajmniej jednej relacji *hasSibling* z członkiem klasy *Man* - na przykład obiekty które mają chociaż jednego brata,
- Klasy obiektów, które są w tylko jednej relacji *hasSibling* z obiektem, który należy do klasy *Women* - są to obiekty które posiadają tylko siostry,
- Klasy obiektów, które są w więcej niż trzech relacjach typu *hasSibling*,
- Klasy obiektów, które są w przynajmniej jednej relacji typu *maDodatek* z obiektem, który należy do klasy *DodatekMięsny*,
- klasy obiektów, które są w relacji *maDodatek* z obiektami, które należą do klasy *DodatekWarzywny*.

W OWL możemy opisać wszystkie powyższe klasy obiektów z wykorzystaniem *ograniczeń.* Ograniczenia w OWL możemy podzielić na 3 główne kategorie:

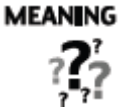
- ograniczenia kwantyfikatorowe,
- ograniczenia ilościowe,
- ograniczenia "posiada wartość".

Na początku będziemy używać ograniczeń kwantyfikatorowych, które można podzielić na egzystencjalne i uniwersalne. W tutorialu zilustrujemy oba typy ograniczeń.

Ograniczenia Egzystencjalne i Uniwersalne:

- Ograniczenia egzystencjalne opisują klasy indywiduów, które są stroną w *conajmniej jednej* relacji określoną własnością z obiektem, który jest instancją określonej klasy. Na przykład: klasa indywiduów, które są w *conajmniej jednej* (some) relacji *maDodatek* z instancją klasy *DodatekWarzywny*. W Protégé 5.1 słowo kluczowe "**some"**(jakieś) jest używane do wskazania ograniczenia egzystencjalnego.
- Ograniczenie uniwersalne opisuje klasy indywiduów, które dla określonej własności mają relacje tylko z obiektami należącymi do określonej klasy. Na przykład: klasa obiektów, które są w relacji *maDodatek* tylko z obiektami klasy *DodatekWarzywny*.

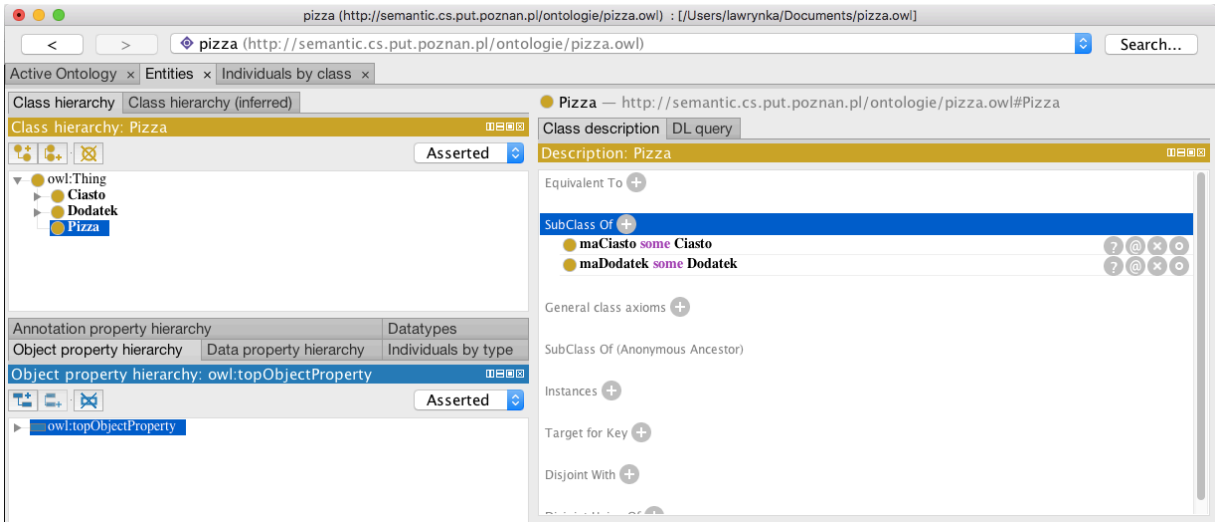
Przyjrzyjmy się bliżej przykładowi ograniczeń egzystencjalnych. Ograniczenie *maDodatek some* (jakieś) *DodatekWarzywny* jest ograniczeniem egzystencjalnym (wskazuje na to zastosowanie słowa kluczowego "some"/"jakieś"), które jest wprowadzone za pomocą własności *maDodatek*, która ma "wypełnienie" (wartość przeciwdziedziny) *DodatekWarzywny*. To ograniczenie opisuje *klasę*, indywiduów które są w *przynajmniej jednej* relacji *maDodatek* z obiektem, który jest członkiem klasy *DodatekWarzywny*.



Ograniczenie opisuje *klasę anonimową* (nienazwaną klasę). Klasa anonimowa zawiera wszystkie obiekty, które spełniają dane ograniczenie - na przykład: wszystkie obiekty, które są w danej relacji należą do danej klasy.

Ograniczenia klas są wyświetlane i edytowane za pomocą widoku "*Class Description*", który jest przedstawiony na Rysunku 19. Widok "*Class Description*" zakładki **"***Class hierarchy***"** w Protégé przechowuje wszystkie informacje użyte do opisu klasy.

Ograniczenia są używane w opisie klas OWL do tworzenia anonimowych nadklas opisywanych klas. Przykładowo, moglibyśmy opisać klasę Pizza jak na Rysunku 19.



Rysunek 19.

8.2 Ograniczenia egzystencjalne

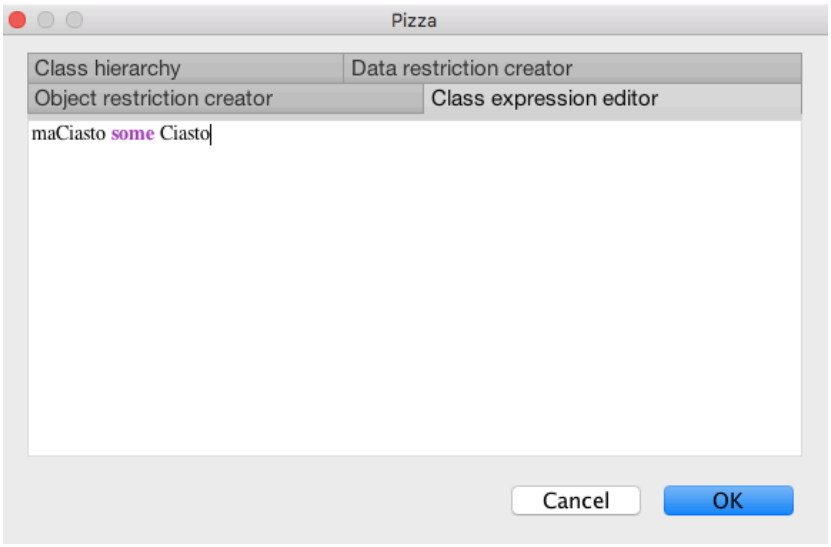
Ograniczenia egzystencjalne są zdecydowanie najczęstszym typem ograniczeń wykorzystywanym w ontologiach OWL. Ograniczenie egzystencjalne opisuje klasę indywiduów które są w *co najmniej jednej* ("some"/"jakiś") relacji (wykorzystując określoną własność) z indywiduum, który jest członkiem określonej klasy. Na przykład, *maCiasto some Ciasto* opisuje wszystkie obiekty, które są w *co najmniej jednej* relacji *maCiasto* z obiektem, który jest członkiem klasy *Ciasto* - mówiąc prościej - wszystkie obiekty, które mają *co najmniej jedno* ciasto.

Ćwiczenie 14. Dodanie ograniczenia do klasy *Pizza*, stanowiącego o tym, że *Pizza* musi mieć *Ciasto*

---

1. Wybierz *Pizza* na zakładce "Class hierarchy".
2. Wybierz przycisk "*Add*" ( ) znajdujący się obok nagłówka "*Sub Class of*" w widoku "*Class Description* ", przedstawionym na rysunku 20, w celu stworzenia potrzebnego warunku.
3. Wybierz zakładkę "*Class expression editor*" - spowoduje to wyświetlenie pola tektsowego gdzie wpiszemy nasze ograniczenie, tak jak przedstawiono to na rysunku 21.

Rysunek 20. Tworzenie ograniczenia



Rysunek 21. Tworzenie ograniczenia

Pole tekstowe w zakładce "*Class expression editor*" pozwala Ci na tworzenie ograniczeń używając nazw klas, własności i indywiduów.
Aby stworzyć ograniczenie musimy wykonać trzy kroki:

- wprowadzić nazwę własności, która ma być ograniczona (lub wybrać ją z listy "*property list*"),
- wprowadzić typ ograniczenia (np. "some" gdy chcemy stworzyć ograniczenie egzystencjalne),
- wprowadzić "filler"(klasę / indywiduum, która / które ma być w związku z klasą, dla której definiujemy to ograniczenie).

Ćwiczenie 15. Dodanie ograniczenia do klasy *Pizza*, stanowiącego o tym, że *Pizza* musi mieć *Ciasto* (kontynuacja)

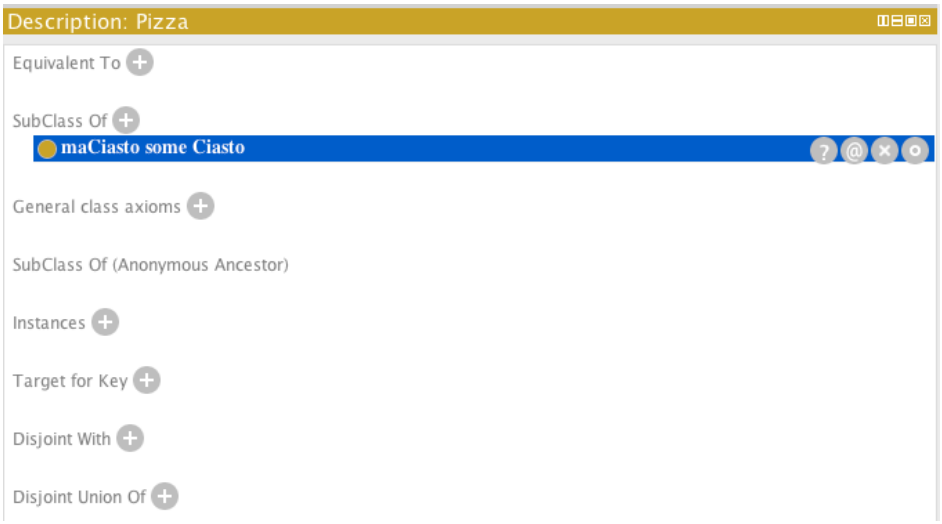
---

1. Możesz korzystając z "drag and drop" przeciągnąć *maCiasto* z listy "property list" na pole tekstowe edytora lub po prostu je wpisać,
2. Teraz dodaj typ ograniczenia, w tym przypadku jest to ograniczenie egzystencjalne - należy więc wpisać słowo kluczowe "some",
3. Uzupełnij "*filler*" jako *Ciasto* - aby to zrobić wprowadź *Ciasto* lub skorzystaj z opcji "drag and drop".
4. Wciśnij przycisk *OK* aby utworzyć ograniczenie i zamknąć edytor. Jeżeli wszystkie informacje zostały poprawnie wprowadzone edytor zamknie się a wprowadzone przez nas ograniczenie zostanie przedstawione pod nagłówkiem "*SubClass of*".

Jeśli wpisane zostały błędne dane będą one podkreślone czerwonym kolorem i okienko z błędem, które się pojawi będzie zawierało podpowiedź co jest źródłem błędu.

Bardzo użyteczną funkcją "expression builder'a" jest możliwość automatycznego uzupełniania nazw klas, własności oraz obiektów. Automatyczne uzupełnianie jest aktywowane poprzez wciśnięcie "Ctrl-Space".

Widok "class description" powinien wyglądać teraz podobnie do tego przedstawionego na rysunku 22.



Rysunek 22. Widok "class description": Opis klasy *Pizza*.

Opisaliśmy klasę *Pizza* jako podklasę *Thing* oraz podklasę obiektów, które mają ciasto, które jest rodzajem klasy *Ciasto***.** Zauważmy, że jest to warunek konieczny – jeżeli coś jest *Pizzą* to jest konieczne, żeby było członkiem klasy Thing (w OWL wszystko jest członkiem klasy *Thing*) i jest konieczne, aby posiadało jakiś rodzaj *Ciasta***.**
Bardziej formalnie, jeżeli coś jest obiektem klasy *Pizza* to jest konieczne, aby był on w relacji z członkiem klasy *Ciasto* poprzez własność *maCiasto*.

Kiedy do opisu klas używamy ograniczeń, to właściwie określamy anonimowe nadklasy klas opisywanych. Na przykład, możemy powiedzieć, że *PizzaMięsna* jest podklasą, między innymi, klasy *Pizza* i również podklasą rzeczy, które mają co najmniej jeden dodatek będący *DodatkiemMięsnym***.**

Tworzenie różnych rodzajów Pizzy.

Nadszedł czas, aby dodać różne rodzaje pizzy do naszej ontologii. Zaczniemy od dodania *Margherita*. Aby utrzymać porządek naszej ontologii, pogrupujemy nasze różne pizze w klasie *NazwanaPizza*.

Ćwiczenie 16. Tworzenie podklasy klasy *Pizza* - *NazwanaPizza* i podklasy klasy *NazwanaPizza* - Margherita

---

1. Wybierz zakładkę „Class hierarchy" i z hierarchii klas wybierz klasę *Pizza*
2. Wybierz przycisk "*Add*" ( ), aby stworzyć nową podklasę klasy *Pizza*, i nazwij ją *NazwanaPizza***.**
3. Stwórz nową podklasę klasy *NazwanaPizza* i nazwij ją *Margherita*
4. Dodaj komentarz do klasy *Margherita* używając widoku **„***Annotations***",** który jest zlokalizowany obok widoku hierarchii klas. Możesz wpisać: *Pizza*, która ma dodatki: sos pomidorowy i mozzarella. Pamiętajmy, że zawsze dobrze jest dokumentować klasy, własności itp., zwłaszcza, jeżeli będziemy później wykorzystywać je do budowy innych, bardziej, rozbudowanych ontologii.

Mając stworzoną klasę *Margherita* musimy teraz zdefiniować jej dodatki. W tym celu dodamy dwa ograniczenia mówiące, że *Margherita* ma dodatek *Mozzarella* i *SosPomidorowy***.**

Ćwiczenie 17. Utworzenie ograniczenia egzystencjalnego na *Margherita*, stanowiącego, że *Margherita* ma co najmniej jeden dodatek *SosPomidorowy*.

---

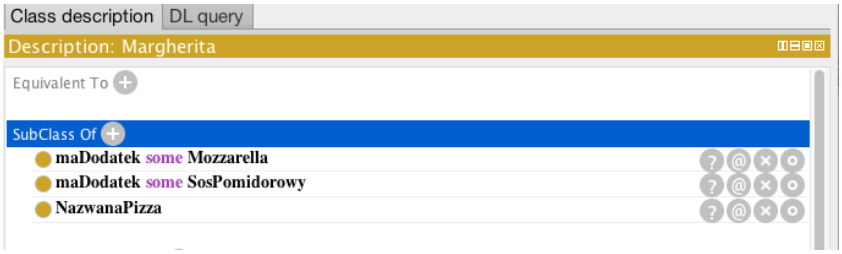
1. Upewnij się, że zaznaczona jest klasa *Margherita* w hierarchii klas.
2. Wybierz przycisk "*Add*" ( ) sekcji *Subclass Of* w widoku „*Class Description*".
3. Wybierz zakładkę „*Object restriction creator*".
4. Wybierz *maDodatek* w polu „*Restricted property*"
5. Wybierz *some* jako „*Restriction type*"
6. Wybierz klasę *SosPomidorowy* jako "*Restriction filler*"
7. Kliknij *OK* aby stworzyć ograniczenie – jeżeli pojawią się jakiekolwiek błędy, ograniczenie nie zostanie stworzone.

Teraz dodamy jeszcze imformację o dodatku *Mozzarella***.**

Ćwiczenie 18. Stworzenie ograniczenia egzystencjalnego na *Margherita*, stanowiącego, że *Margherita* ma co najmniej jeden dodatek *Mozzarella*.

1. Upewnij się, że zaznaczona jest klasa *Margherita* w hierarchii klas.
2. Wybierz przycisk "*Add*" ( ) sekcji *Subclass Of* w widoku „Class Description”.
3. Wybierz zakładkę „*Object restriction creator*”.
4. Wybierz *maDodatek* w polu „*Restricted property*”
5. Wybierz *some* jako „*Restriction type*”
6. Wybierz klasę *Mozzarella* jako “*Restriction filler*”
7. Kliknij *OK* aby stworzyć ograniczenie – jeżeli pojawią się jakiekolwiek błędy, ograniczenie nie zostanie stworzone.

Sekcja „*Description*” powinna wyglądać jak na rysunku 23.



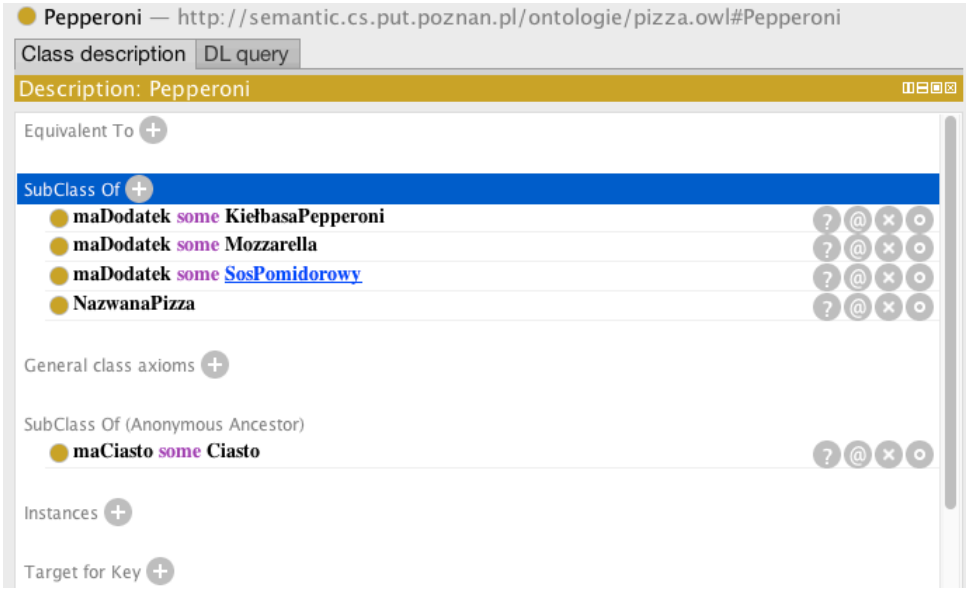
Rysunek 23. Opis Margherita

Teraz stworzymy klasę reprezentującą pizzę Pepperoni, która posiada dodatki: sos pomidorowy, ser mozzarella i kiełbasę pepperoni. Ponieważ klasa *Pepperoni* jest podobna do klasy *Margherita* (Pepperoni jest prawie taka sama, różnicą jest to, że Pepperoni ma o jeden więcej dodatek - kiełbasę pepperoni) stworzymy kopię klasy *Margherita* i dodamy dodatkowe ograniczenie, które opisze, że ma jeszcze jeden dodatek w postaci kiełbasy pepperoni.

Ćwiczenie 19. Stworzenie klasy *Pepperoni* jako kopii i modyfikacji klasy *Margherita*

1. Wybierz zakładkę „*Class hierarchy*” i z hierarchii klas wybierz klasę *Margherita*
2. Wybierz „*Duplicate selected class*” z menu *Edit*. W oknie, które się pojawi, należy wpisać nazwę nowej klasy. Wpisujemy *Pepperoni***.**
3. Upewnij się, że klasa *Pepperoni* jest wybrana. Teraz wybierz przycisk "*Add*" ( ) sekcji *Subclass Of* w widoku „Class Description”
4. Przejdź na zakładkę „*Object restriction creator*”.
5. Wybierz *maDodatek* w polu „*Restricted property*”
6. Wybierz *some* jako „*Restriction type*”
7. Wybierz klasę *KiełbasaPepperoni* jako “*Restriction filaer*”
8. Kliknij *OK* aby stworzyć ograniczenie.

Sekcja „*Description*” powinna wyglądać jak na rysunku 24.



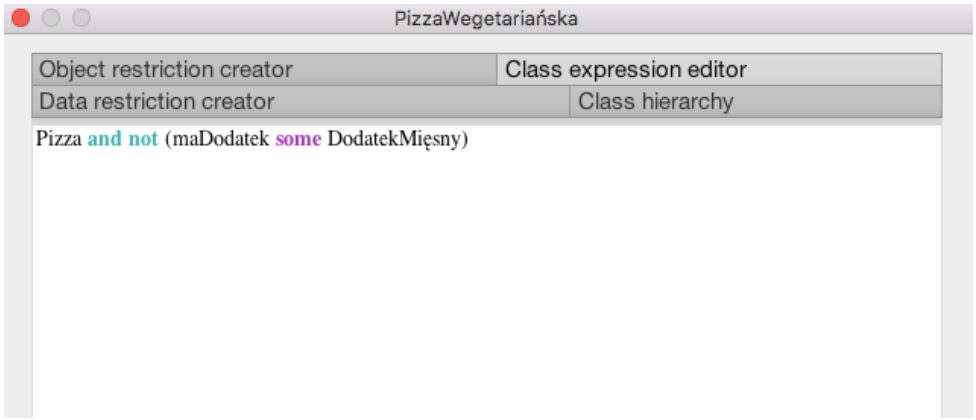
Rysunek 24. Opis Pepperoni

Sekcja ‘*SubClass Of*’ w ‘*Description*’ pozwala na podanie warunku *koniecznego* jakie muszą spełniać instancje danej klasy. Jest to *częściowa definicja* klasy. Możliwe jest także zamodelowanie pełnej *definicji* klasy za pomocą sekcji ‘*Equivalent To*’, która umożliwia podanie warunku *koniecznego i wystarczającego* do tego aby spełniający go obiekt znalazł się w danej klasie.

Ćwiczenie 20. Utworzenie klasy *PizzaWegetariańska*

---

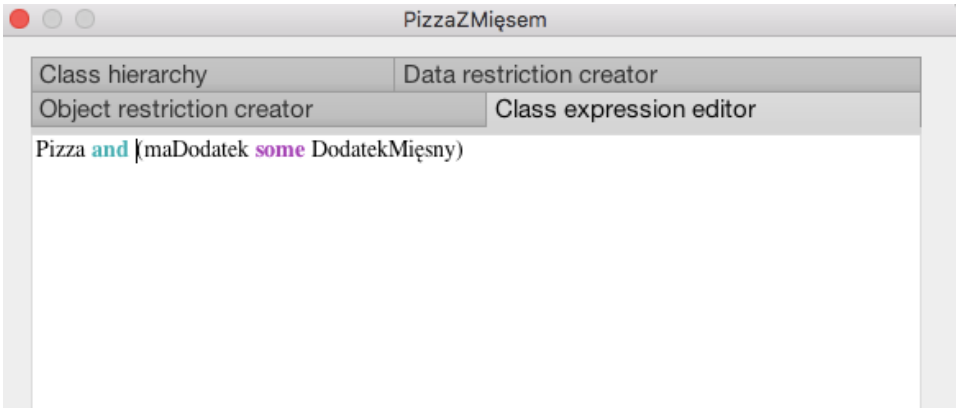
1. Stwórz nową podklasę klasy *Pizza* i nazwij ją *PizzaWegetariańska*
2. Upewnij się, że klasa *PizzaWegetariańska* jest wybrana. Teraz wybierz przycisk "*Add*" (⊕) sekcji *Equivalent To* w widoku „*Class Description*”
3. Wybierz zakładkę "*Class expression editor*" - spowoduje to wyświetlenie pola tektsowego gdzie wpiszemy nasze ograniczenie, tak jak przedstawiono to na rysunku 25.



Rysunek 25. Tworzenie ograniczenia dla klasy *PizzaWegetariańska*

Ćwiczenie 21. Utworzenie klasy *PizzaZMięsem*

1. Stwórz nową podklasę klasy *Pizza* i nazwij ją *PizzaZMięsem*
2. Upewnij się, że klasa *PizzaZMięsem* jest wybrana. Teraz wybierz przycisk "*Add*" (⊕) sekcji *Equivalent To* w widoku *„Class Description”*
3. Wybierz zakładkę "*Class expression editor*" - spowoduje to wyświetlenie pola tektsowego gdzie wpiszemy nasze ograniczenie, tak jak przedstawiono to na rysunku 26.

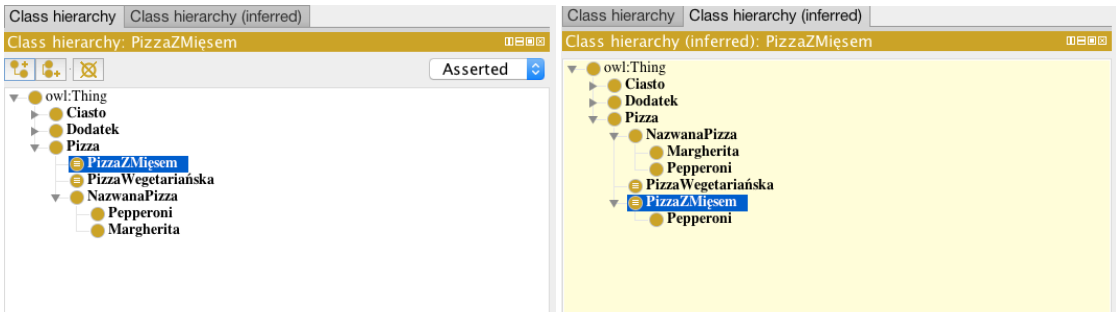


Rysunek 26. Tworzenie ograniczenia dla klasy *PizzaZMięsem*

10. Wnioskowanie

Ćwiczenie 22. Automatyczna klasyfikacja/tworzenie hierarchii klas na podstawie ich własności.

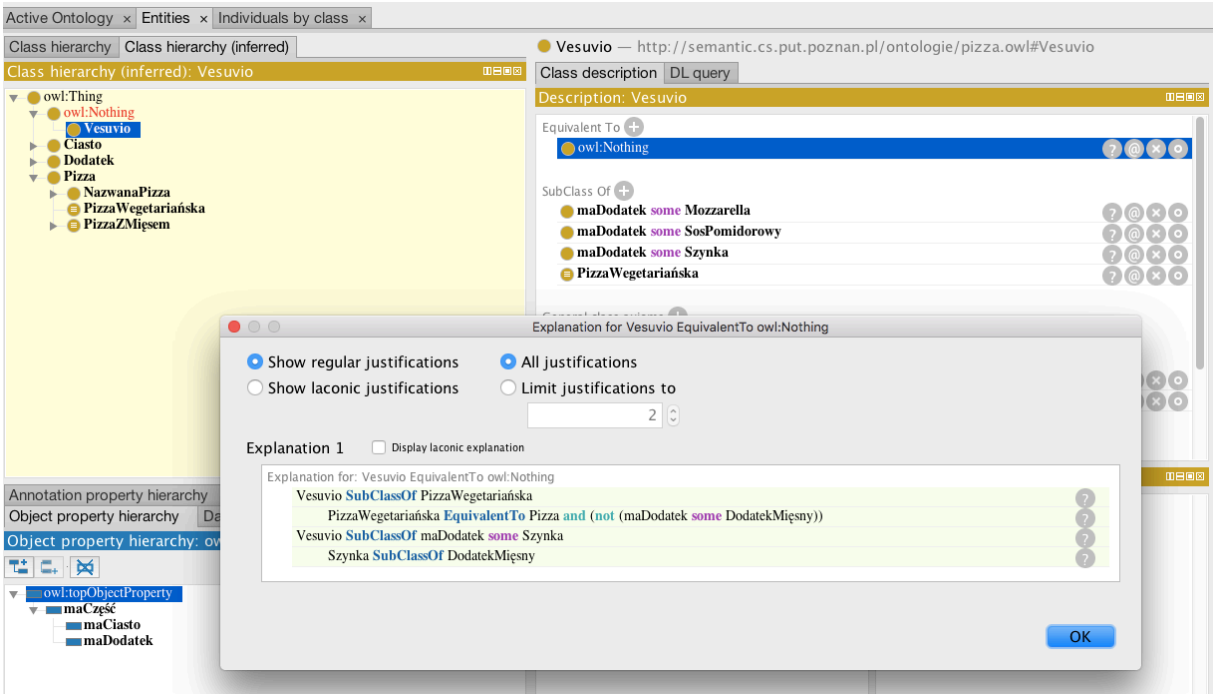
1. Z menu ‘*Reasoner*’ wybierz ‘*Start reasoner*’ (zostaw zaznaczony domyślny silnik wnioskujący lub wybierz dowolny jeżeli żaden nie jest zaznaczony). Silnik wnioskujący dokona automatycznej (re-)klasyfikacji hierarchii klas.
2. Porównaj hierarchię klas jaką wprowadziłeś (‘*Class hierarchy*’) z tą jaką wywnioskował silnik (‘*Class hierarchy*’ (*inferred*)). Zauważ, że w tym drugim przypadku, klasa *Pepperoni* została automatycznie zaklasyfikowana jako podklasa klasy *PizzaZMięsem* (rys. 26).



Rysunek 26. Porównanie wprowadzonej i automatycznie wywnioskowanej hierarchii klas.

Ćwiczenie 23. Automatyczna klasyfikacja – wykrywanie niespójności.

1. Wprowadź nową podklasę klasy *PizzaWegetariańska* o nazwie *Vesuvio*.
2. Wprowadź dla tej klasy ograniczenia własności o postaci:
   a. *maDodatek some Mozzarella*
   b. *maDodatek some SosPomidorowy*
   c. *maDodatek some Szynka*
3. Ponownie uruchom silnik wnioskujący (z menu ‘*Reasoner*’ wybierz ‘*Start reasoner*’ *bądź* też ‘*Synchronize reasoner*’).
4. Silnik wnioskujący powinien wykryć niespójność z uwagi na to, że pizza wegetariańska nie może mieć dodatku mięsnego jakim jest szynka. W wyniku tej niespójności, klasa *Vesuvio* staje się niespełnialna (nie ma takich instancji, które mogłyby spełnić ograniczenia nałożone na tą klasę, są one sprzeczne). Klasa *Vesuvio* zostaje oznaczona na czerwono i w ‘*Class hierarchy*’ (*inferred*) zostaje umieszczona jako podklasa klasy *Nothing*, oznaczającej niespełnialne klasy.
5. W widoku ‘*Description*’ klasy *Vesuvio*, klasa *Nothing* pojawia się jako ‘Equivalent Class’.
6. Po kliknięciu ikonki ze znakiem zapytania, można odczytać wyjaśnienie takiej dedukcji (Rysunek 27).



Rysunek 27. Ilustracja niespełnialnej klasy *Vesuvio* i objaśnienia takiej dedukcji o niespełnialności.