Języki modelowania ontologii: RDFs, OWL

Agnieszka Ławrynowicz

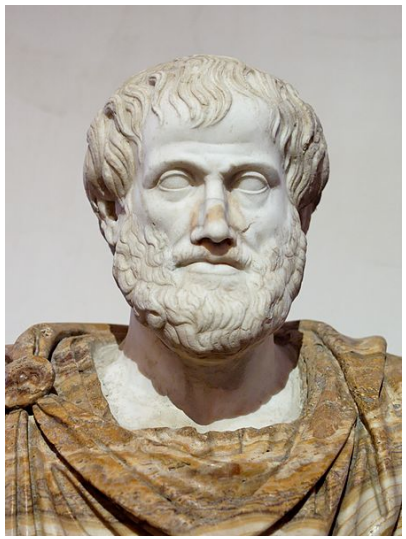
Instytut Informatyki
Politechniki Poznańskiej

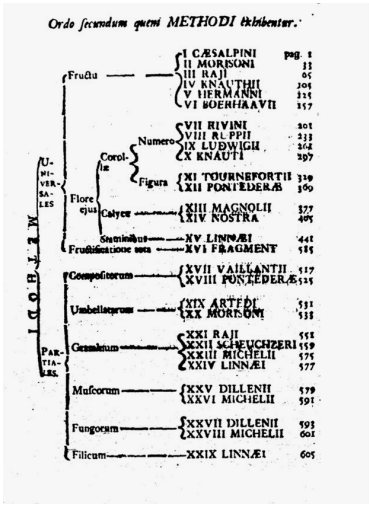
Poznań, 2017
Ver 2.4

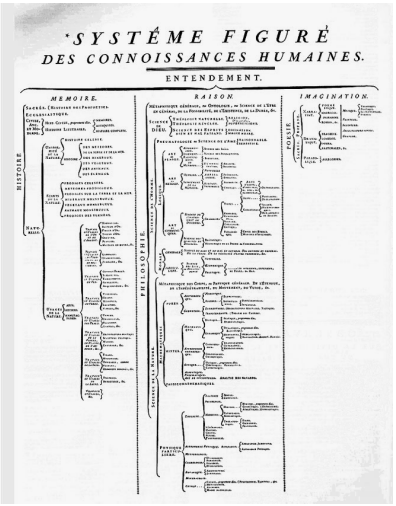
Ontologia z punktu widzenia filozofa

Ontologia (metafizyka)= *nauka o bycie* (Arystoteles, *Metafizyka*,ks. IV)

- ***co to jest byt? co charakteryzuje byt?***
- ***jak dokonywać klasyfikacji bytów?***







Ontologia z punktu widzenia informatyka

"engineering artefact [...]" **(Guarino 98)**

"An ontology is a

formal specification ⇨ maszynowa interpretacja

of a shared ⇨ grupa osób, konsensus

conceptualization ⇨ abstrakcyjny model zjawisk, pojęcia

of a domain of interest" ⇨ wiedza dziedzinowa

(Gruber 93)

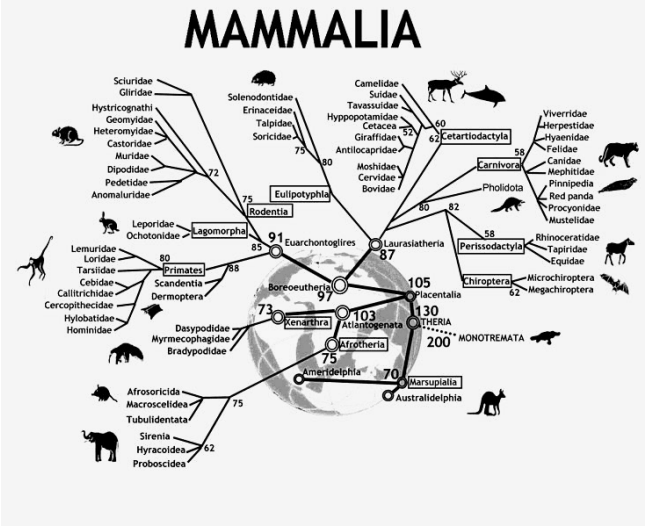
ontologia = formalna specyfikacja pojęć z danej dziedziny

Przykłady ontologii: proste taksonomie

Taksonomia (gr. *taksis* - układ, porządek + *nomos* - prawo)
nauka o zasadach i metodach klasyfikowania

- Klasyfikacja organizmów Linneusza

  (**Karol Linneusz 1707-1778, „ojciec współczesnej taksonomii”**)



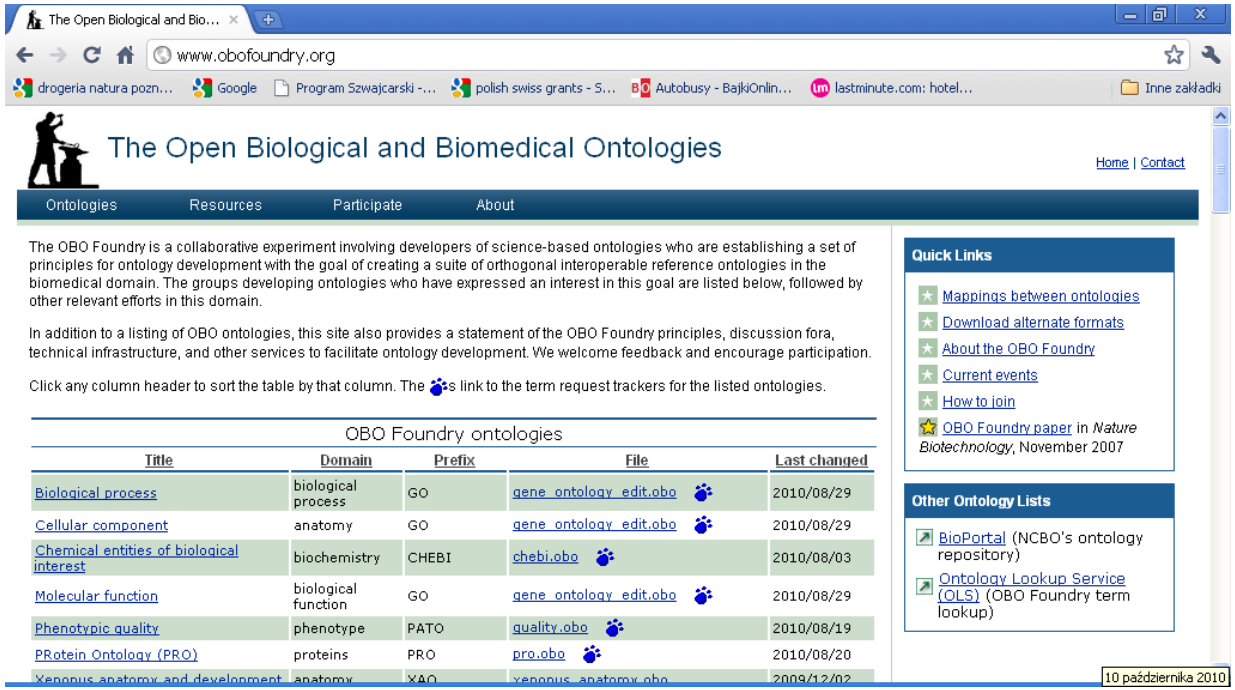
- kategorie wyszukiwarki Yahoo!

  (http://dir.yahoo.com/)
- Open Directory Project 590,000 kategorii

  (http://dmoz.org/)
- katalog produktów Amazon

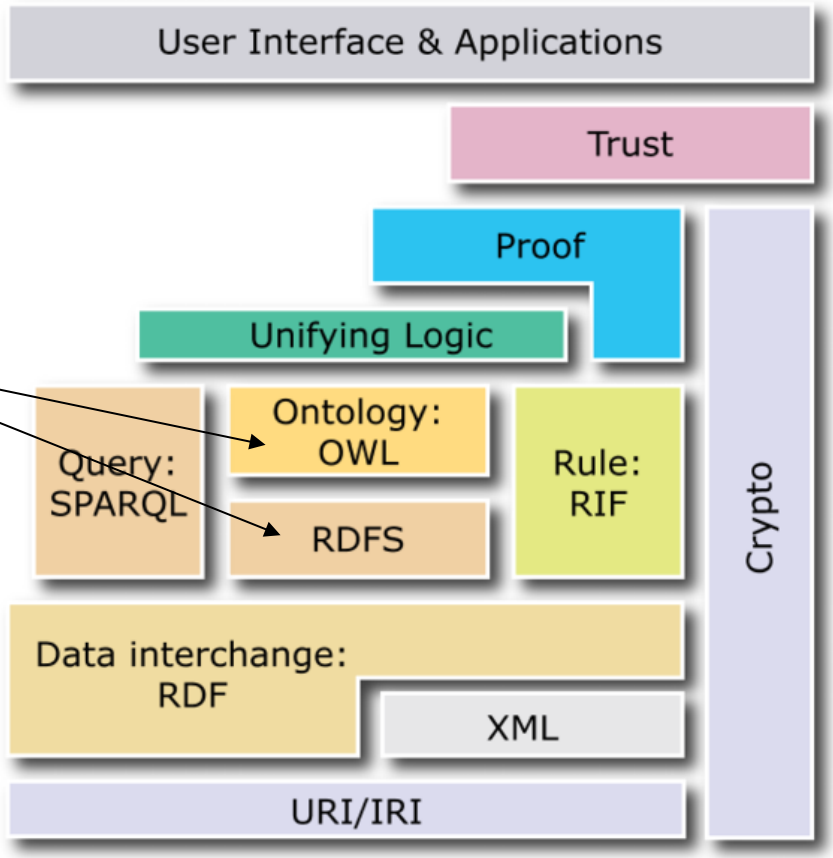
Przykłady ontologii: złożone

- Ontologie wyższe (‘upper ontologies’): DOLCE, BFO, Cyc, Sumo
- Ontologie biomedyczne: Obo Foundry (Open Biological and Biomedical Ontologies),w tym GO (Gene Ontology); Snomed CT, NCI, Galen



Stos języków Sieci Semantycznej

Języki modelowania ontologii



RDFS – RDF Schema

- **RDF definiuje tylko model danych**

RDF Schema pozwala na definiowanie słowników pojęć wraz z relacjami między pojęciami

  – pomaga wyrazić jak dane pojęcie powinno być interpretowane

RFDS – kluczowe klasy

rdfs:Resource – zasoby

rdfs:Class – klasy

rdfs:Literal – typy proste

odziedziczone z RDF:

rdf:type – określa klasę zasobu (której zasób jest instancją)

RDFS – kluczowe własności

- **rdfs:subClassOf** – określa nadklasę danej klasy

  wszystkie instancje klasy są także instancjami jej nadklasy
- **rdfs:subPropertyOf** – wiąże własność z jedną z jej podwłasności

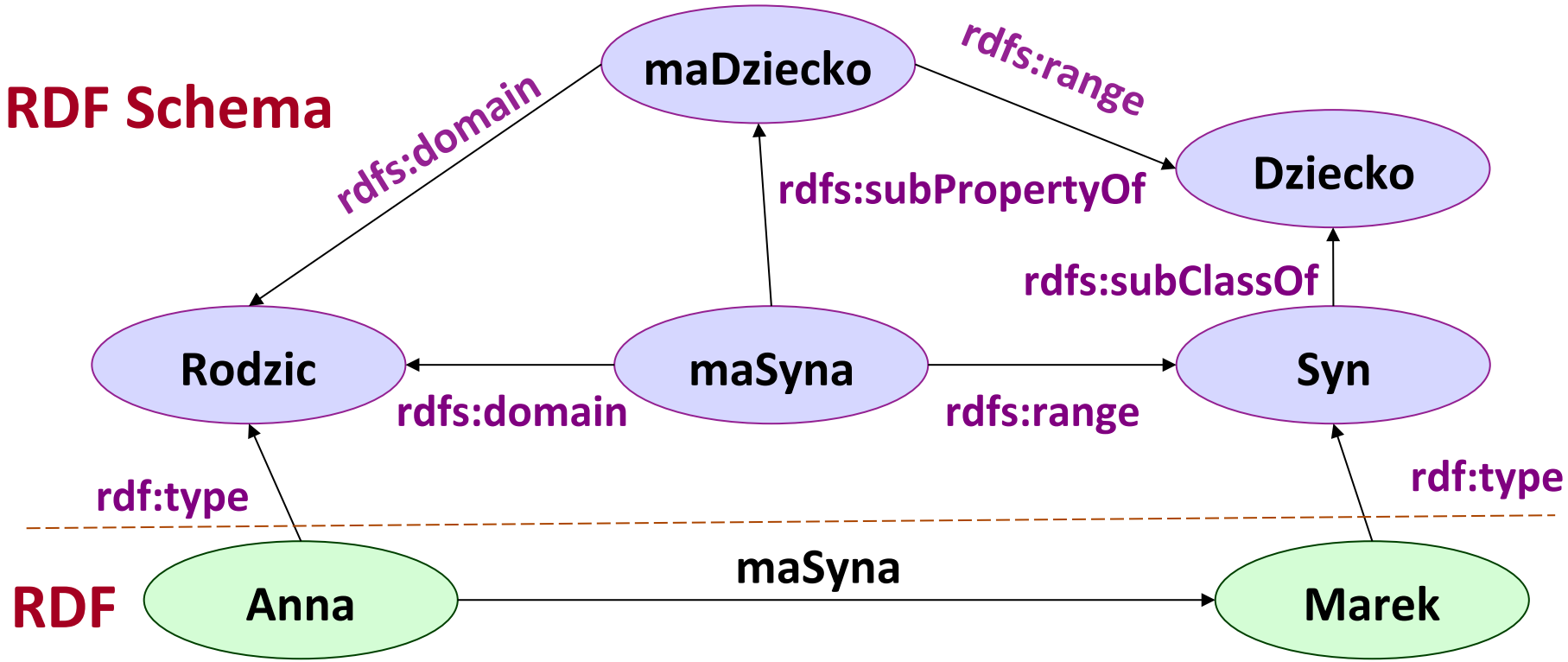
```
:Syn rdfs:subClassOf :Dziecko .

:maSyna rdfs:subPropertyOf :maDziecko .
```

RDFS – kluczowe własności c.d.

- **rdfs:domain** – specyfikuje dziedzinę własności:
  - czyli klasę tych zasobów, które mogą się pojawiać jako podmiot (subject) w trójce z tym predykatem
  - Jeśli nie podano dziedziny, to w zdaniu może wystąpić dowolny zasób
- **rdfs:range** – określa zakres własności:
  - czyli klasę tych zasobów, które mogą się pojawiać jako obiekt (object) w trójce z tym predykatem

```
:maDziecko rdf:type rdf:Property ;
           rdfs:domain :Rodzic ;
           rdfs:range :Dziecko .
```

RDF osadzony w RDFS – przykład



OWL – Web Ontology Language

- **dostarcza bogatej kolekcji operatorów do konstrukcji złożonych pojęć**
- **semantyka języka korzysta z badań w ramach sztucznej inteligencji w zakresie reprezentacji wiedzy – logiki deskrypcyjne**

Skąd pochodzi akronim „OWL”?

Web Ontology Language ≠ WOL

Owl lived at The Chestnuts, an old-world residence of great charm, which was grander than anybody else’s, or seemed so to Bear, because it had both a knocker and a bell-pull. Underneath the knocker there was a notice which said:
PLES RING IF AN RNSER IS REQIRD.
Underneath the bell-pull there was a notice which said:
PLEZ CNOKE IF AN RNSR IS NOT REQID.
These notices had been written by Christopher Robin, who was the only one in the forest who could spell; for Owl, wise though he was in many ways, able to read and write and spell his own name **WOL**, yet somehow went all to pieces over delicate words like MEASLES and BUTTEREDTOAST.

(A.A. Milne, „Kubuś Puchatek”)

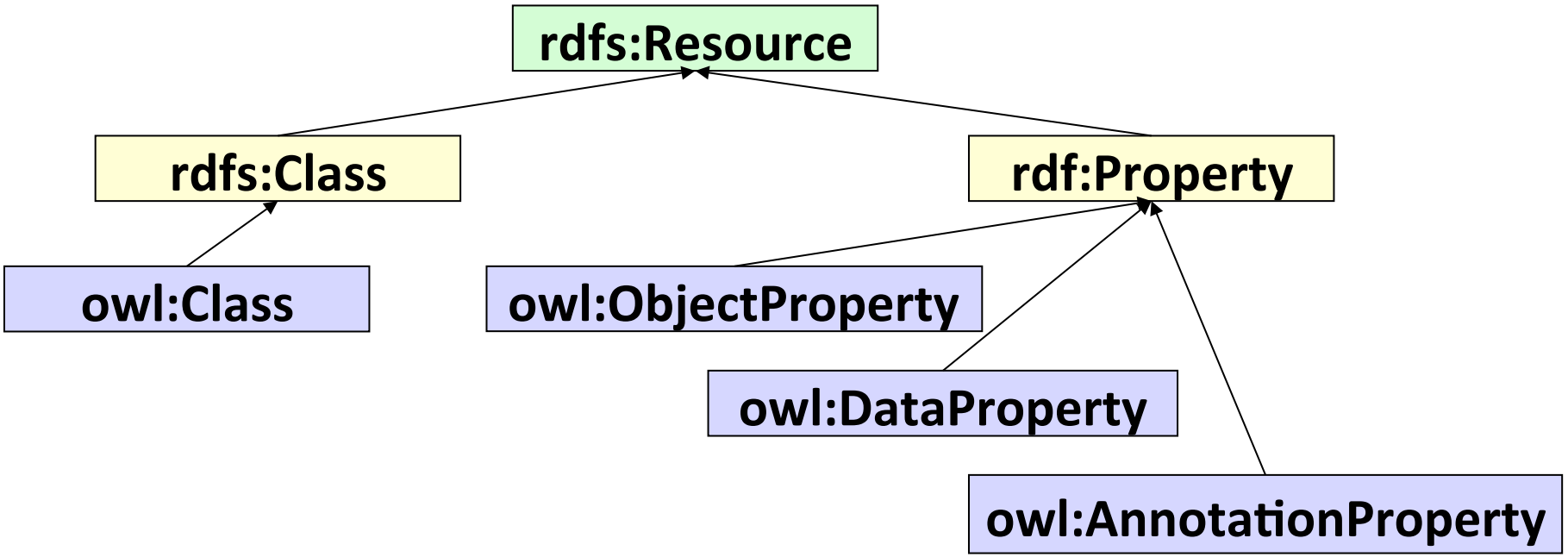
Profile OWL 2

- **OWL EL:**
  - standardowe wnioskowanie w czasie wielomianowym, odpowiedni dla aplikacji korzystających z dużych ontologii (ontologie medyczne)
- **OWL QL:**
  - odpowiedni dla małych ontologii wykorzystujących dużo danych i gdy dostęp do danych jest wymagany w postaci zapytań relacyjnych (np. SQL)
- **OWL RL:**
  - wykorzystuje algorytmy bazujące na technologiach baz danych rozszerzonych o reguły w czasie wielomianowym i manipulujące bezpośrednio na trójkach RDF; („forward chaining rules”)

Składnia

- RDF/XML (główna składnia dla OWL)
- abstrakcyjna składnia (wykorzystywana w dokumencie ze specyfikacją języka)
- składnia Protege-OWL (Manchester OWL Syntax)

Kompatybilność OWL z RDF Schema



owl:Ontology

```
<https://w3id.org/saref>
  rdf:type owl:Ontology ;
  dcterms:created "2015-02-10"^^xsd:date ;
  dcterms:creator "Laura Daniele
   <mailto:laura.daniele@tno.nl>"^^xsd:string ;
  dcterms:issued "2015-04-01"^^xsd:date ;
  dcterms:license "This work is licensed under a Creative Commons
   Attribution License (version 3.0) <http://creativecommons.org/licenses/
   by/3.0/>"^^xsd:string ;
  dcterms:publisher "TNO <https://www.tno.nl/nl/>"^^xsd:string ;
  dcterms:title "SAREF: the Smart Appliances REFerence
   ontology"^^xsd:string ;
  owl:imports geo: ;
  owl:imports <http://www.w3.org/2006/time> ;
  owl:versionInfo "1.0"^^xsd:string ;
.
```

OWL – elementy składowe

Encje – klasy, własności, indywidua i wszelkie inne elementy modelowanej dziedziny

Wyrażenia – złożone pojęcia na temat modelowanej dziedziny

Aksjomaty – twierdzenia, które są prawdziwe w modelowanej dziedzinie

Klasy

- Zbiory instancji, definiowane za pomocą **owl:Class** (podklasa rdfs:Class)

```
:Chłopiec rdf:type owl:Class .
```

- Klasy specjalne:
  - **owl:Thing** (klasa uniwersalna)
  - **owl:Nothing** (klasa pusta, „najniższa”)

Własności

- **własności obiektowe (ang. object properties)**
  - łączą obiekty z innymi obiektami
- **własności literałowe (ang. data properties)**
  - łączą obiekty z literałami (typy danych literałów np. z puli XML Schema)
- **własności adnotacyjne (ang. annotation properties)**
  - łączą obiekty z notatkami na ich temat (*rdfs:label, owl:versionInfo,…)*

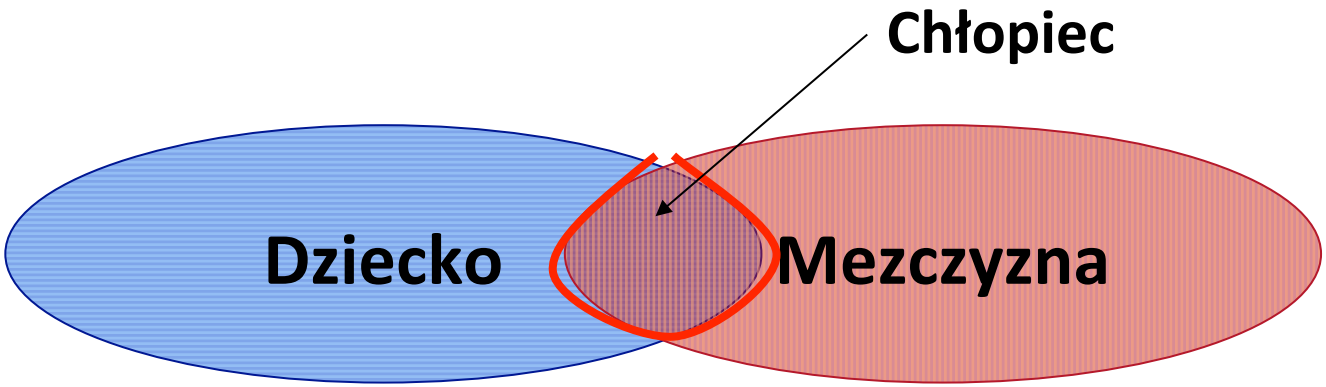
```
:wiek rdf:type owl:DatatypeProperty ;
      rdfs:range xsd:nonNegativeInteger .
```

Klasy – wybrane wyrażenia

- iloczyn
- suma
- negacja
- kwantyfikator egzystencjalny
- kwantyfikator ogólny
- ograniczenie wartości indywiduum

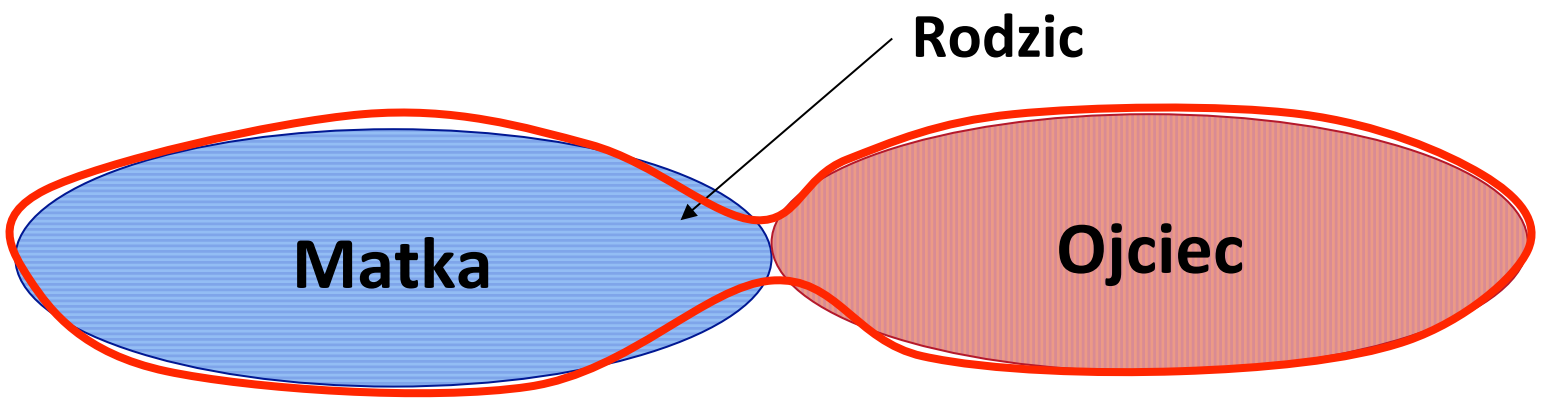
Iloczyn

```
owl:intersectionOf ( :Dziecko
:Mężczyzna
)
```



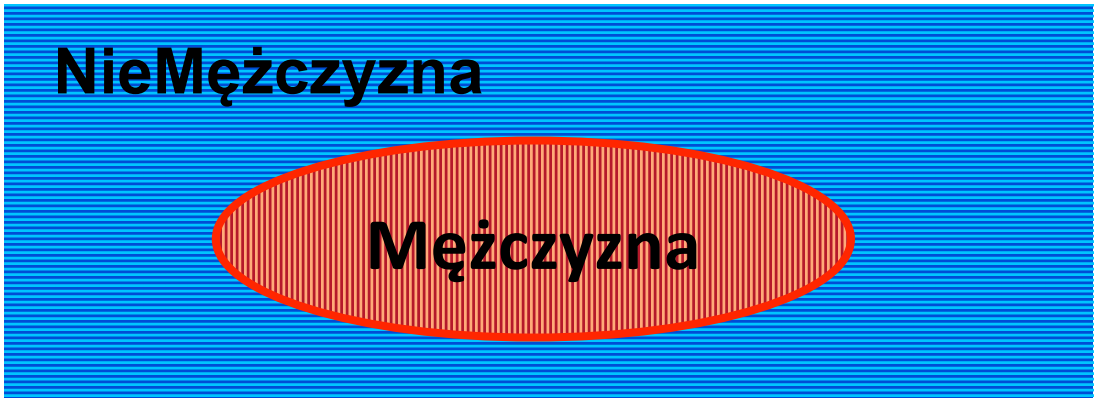
Suma

```
owl:unionOf ( :Matka
:Ojciec
)
```



Negacja (dopełnienie)

```
owl:complementOf :Mężczyzna
```



UWAGA na założenie otwartego świata! (omówione dalej)

Kwantyfikator egzystencjalny

“każdy nauczyciel akademicki musi wykładać przynajmniej jeden przedmiot”

```
:NauczycielAkademicki rdf:type owl:Class ;
        rdfs:subClassOf [ rdf:type owl:Restriction ;
             owl:onProperty :wykłada ;
             owl:someValuesFrom :Przedmiot
        ] .
```

Kwantyfikator ogólny

“asystenci prowadzą tylko laboratoria”

```
:Asystent rdf:type owl:Class ;
  rdfs:subClassOf [ rdf:type owl:Restriction ;
      owl:onProperty :prowadzi ;
      owl:allValuesFrom :Laboratorium
  ] .
```

Kwantyfikator ogólny c.d.

Uwaga! Może to dotyczyć także asystentów, którzy nie prowadzą żadnych zajęć! Wynika to ze znaczenia kwantyfikatora ogólnego w logice pierwszego rzędu.

„wszystkie moje worki z pieniędzmi leżą na tym stole”

Ograniczenie wartości indywiduum

“banany są domyślnie koloru żółtego”

```
:Banan rdf:type owl:Class ;
       rdfs:subClassOf [ rdf:type owl:Restriction ;
                         owl:onProperty :maDomyślnyKolor ;
                         owl:hasValue :żółty
                       ] .
```

Klasy – wybrane aksjomaty

-zawieranie się (subsumcja)

-równoważność

-rozłączność

Opisy (deskrypcje)

- elementarne pojęcia
- SubClassOf

```
:Rodzic rdf:type owl:Class ;
        rdfs:subClassOf [ rdf:type owl:Restriction ;
                          owl:onProperty :maDziecko ;
                          owl:someValuesFrom [ rdf:type owl:Class ;
                                               owl:unionOf ( :Chłopiec
                                                             :Dziewczynka
                                                           )
                                             ]
                        ] .
```

Wszyscy rodzice posiadają między innymi dziecko będące chłopcem lub dziewczynką.

Definicje

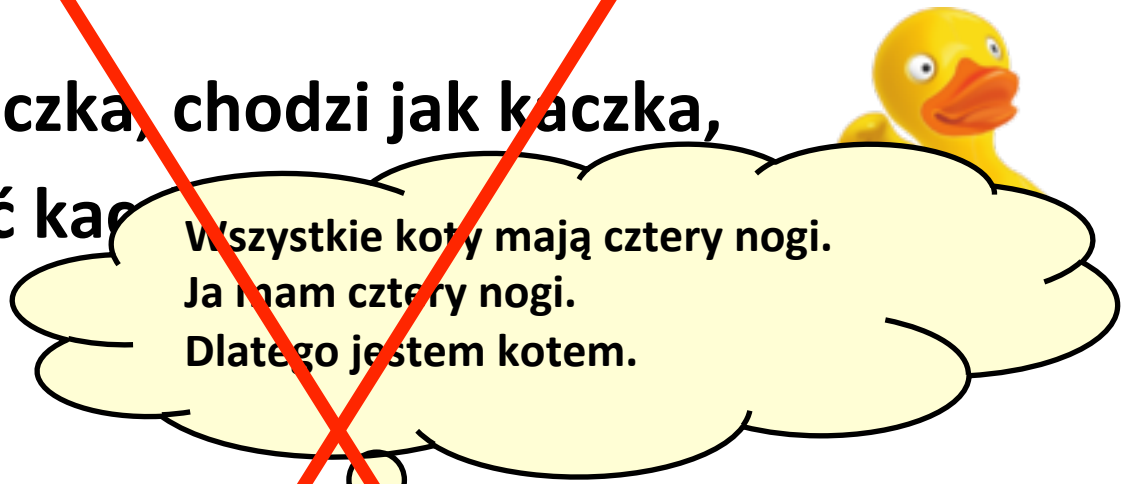
- zdefiniowane pojęcia
- EquivalentClass

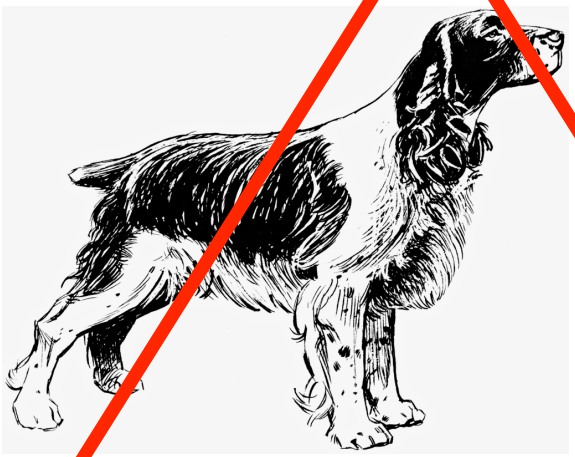
```
:Chłopiec rdf:type owl:Class ;
          owl:equivalentClass [ rdf:type owl:Class ;
          owl:intersectionOf ( :Dziecko
          :Mężczyzna
        ) ] .
```

Każdy kto między innymi jest dzieckiem i jednocześnie mężczyzną jest chłopcem.

Warunki konieczne i wystarczające

„Wygląda jak kaczka, chodzi jak kaczka,
więc musi to być kac





Rozłączność

Dopóki nie są wprowadzone jawnie ograniczenia rozłącznościowe, klasy mogą mieć część wspólną

```
:NauczycielAkademicki owl:disjointWith :Przedmiot
```

Własności obiektowe – wybrane aksjomaty

-własność odwrotna

-własność funkcyjna

-własność przechodnia

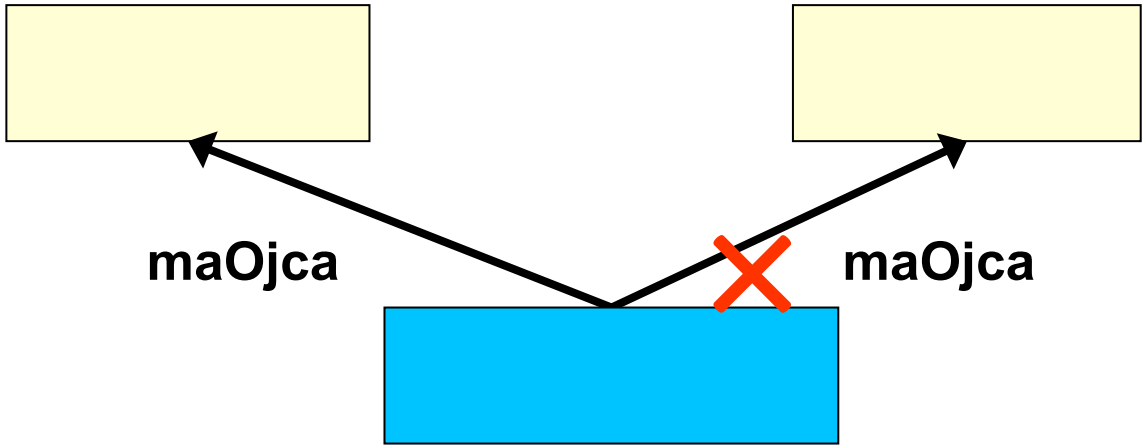
-łańcuchy własności obiektowych (OWL 2)

Własność odwrotna

```
:jestDzieckiem rdf:type owl:ObjectProperty ;
               owl:inverseOf :maDziecko .
```

Własność funkcyjna

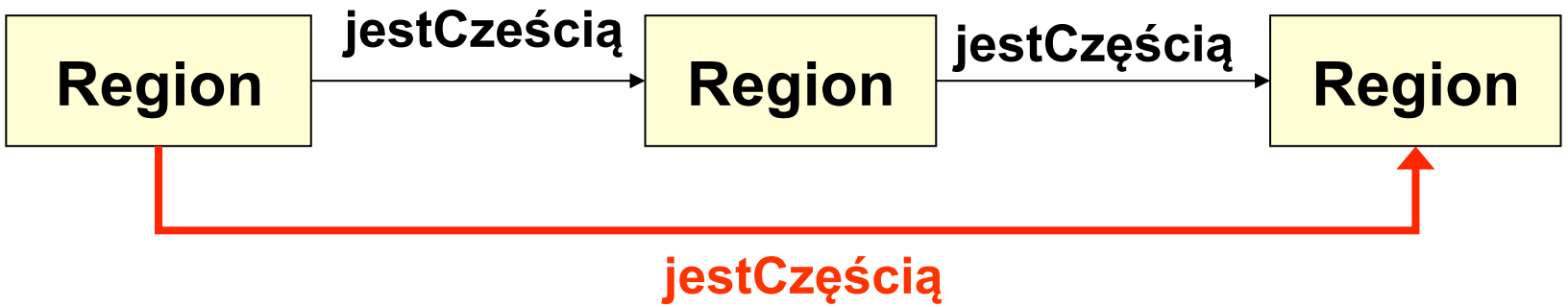
```
:maOjca rdf:type owl:ObjectProperty ,
                 owl:FunctionalProperty .
```



UWAGA na brak założenia o unikalności nazw! (omówione dalej)

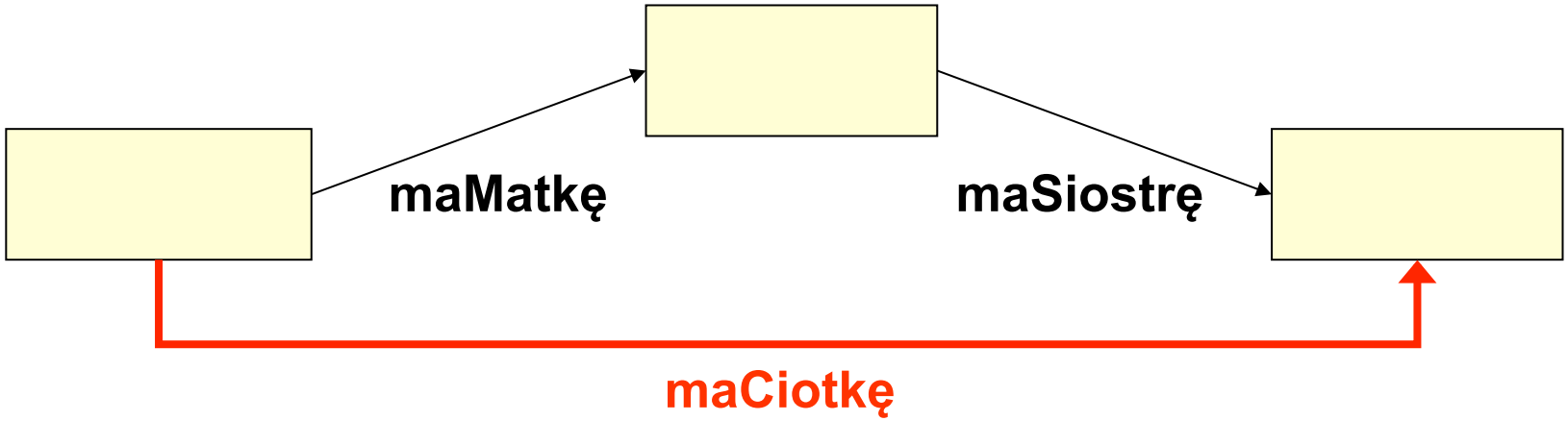
Własność przechodnia

```
:jestCześcią rdf:type owl:ObjectProperty ,
                      owl:TransitiveProperty ;
             rdfs:domain :Region ;
             rdfs:range :Region .
```



Łańcuchy własności obiektowych (OWL 2)

```
:maCiotkę rdf:type owl:ObjectProperty ;
          owl:propertyChainAxiom ( :maMatkę
                                   :maSiostrę
                                 ) .
```



Indywidua

- **Asercje indywiduów do klas:**

```
:Adam rdf:type :Dziecko .
```

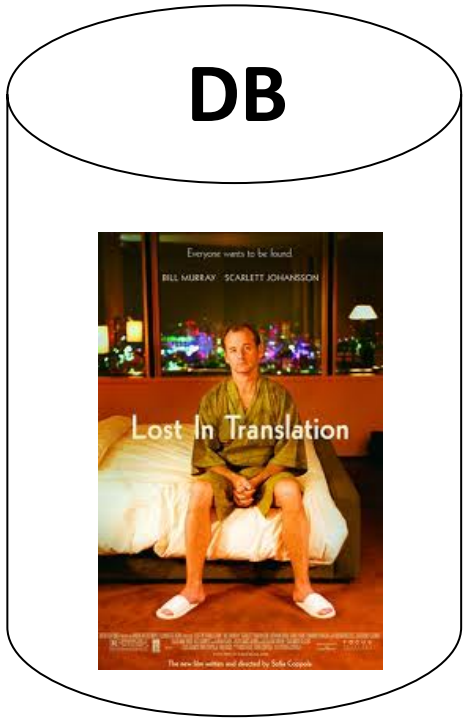
- **Asercje indywiduów do własności:**

```
:Anna :maDziecko :Adam .
```

”Świat zamknięty” kontra ”świat otwarty”

- **Zamknięty świat** (programowanie w logice, bazy danych)
  - *kompletna* wiedza o indywiduach
  - brak informacji jest informacją negatywną (*negation-as-failure*)
- **Otwarty świat** (logika deskrypcyjna, Sieć Semantyczna)
  - *niekompletna* wiedza o indywiduach
  - negacja faktu musi być jawnie podana (*monotonic negation*)

”Świat zamknięty” kontra ”świat otwarty” – przykład 1



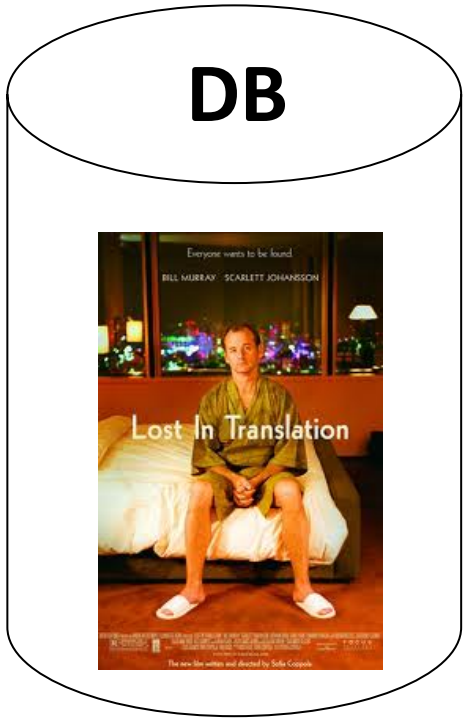
Załóżmy, że w bazie mamy następujące dane:

```
:LostInTranslation rdf:type :OscarMovie .
:SofiaCoppola rdf:type :Director .
:SofiaCoppola :creates :LostInTranslation .
```

Czy „**wszystkie filmy Sofii Coppoli są filmami oskarowymi?**”

Agnieszka Ławrynowicz, Instytut Informatyki Politechniki Poznańskiej,

”Świat zamknięty” kontra ”świat otwarty” – przykład 1



Załóżmy, że w bazie mamy następujące dane:

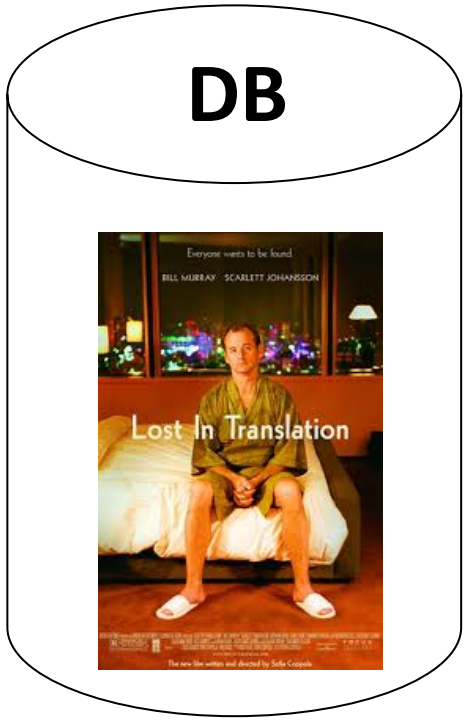
```
:LostInTranslation rdf:type :OscarMovie .
:SofiaCoppola rdf:type :Director .
:SofiaCoppola :creates :LostInTranslation .
```

Czy „**wszystkie filmy Sofii Coppoli są filmami oskarowymi?**”

TAK – zamknięty świat

Agnieszka Ławrynowicz, Instytut Informatyki Politechniki Poznańskiej

”Świat zamknięty” kontra ”świat otwarty” – przykład 1



Załóżmy, że w bazie mamy następujące dane:

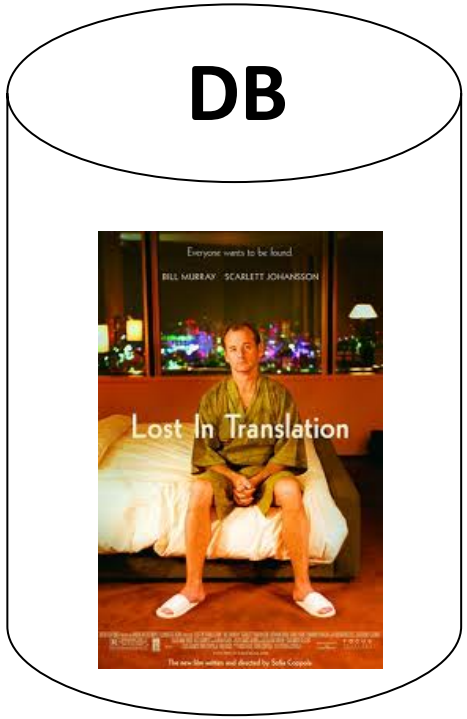
```
:LostInTranslation rdf:type :OscarMovie .
:SofiaCoppola rdf:type :Director .
:SofiaCoppola :creates :LostInTranslation .
```

Czy „**wszystkie filmy Sofii Coppoli są filmami oskarowymi?**”

TAK – zamknięty świat **NIE WIEM – otwarty świat**

Agnieszka Ławrynowicz, Instytut Informatyki Politechniki Poznańskiej

”Świat zamknięty” kontra ”świat otwarty” – przykład 1



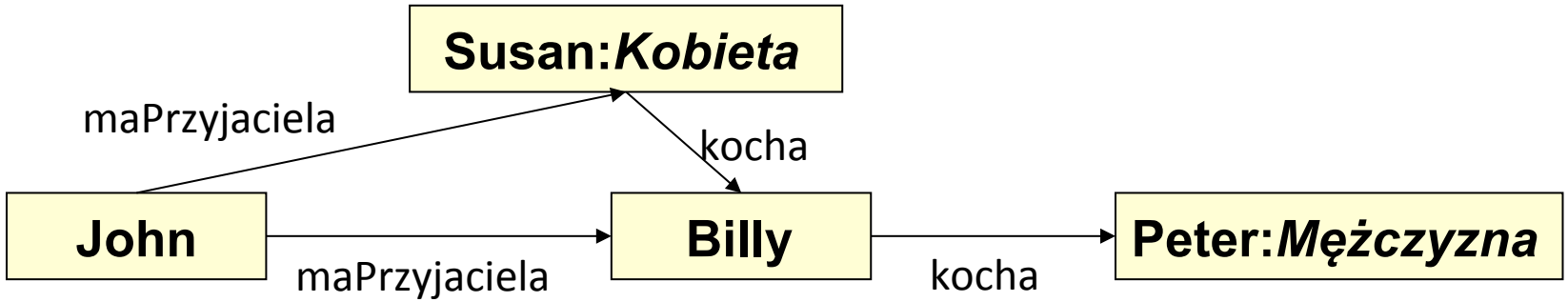
Czy „**wszystkie filmy Sofii Coppoli są filmami oskarowymi?**”

TAK – zamknięty świat

NIE WIEM – otwarty świat

Agnieszka Ławrynowicz, Instytut Informatyki Politechniki Poznańskiej
Posters: Fair use rationale

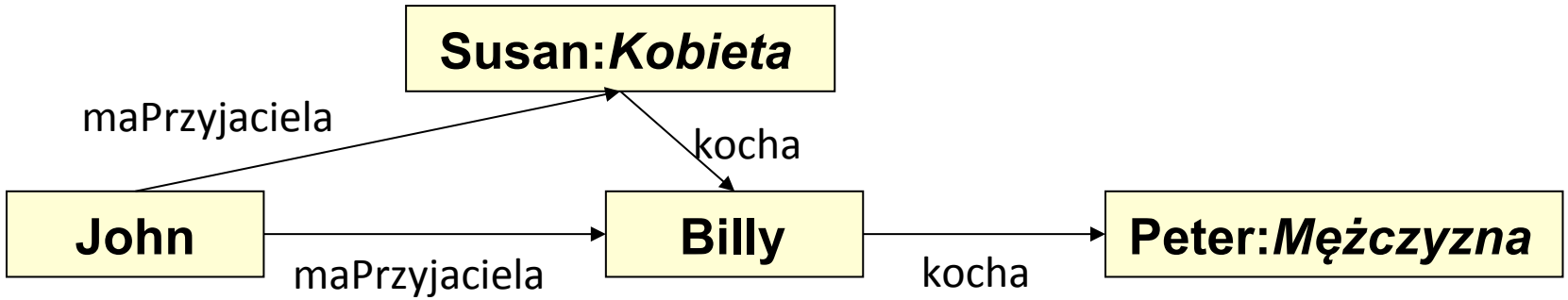
”Świat zamknięty” kontra ”świat otwarty” – przykład 2



Czy John ma przyjaciółkę, która kocha mężczyznę?

Klasyczny przykład ilustrujący wnioskowanie w logikach deskrypcyjnych w ujęciu z: http://galaxy.eti.pg.gda.pl/katedry/kio/dydaktyka/Bazy_Wiedzy/BazyWiedzy_2.pdf, Krzysztof Goczyła, Wojciech Waloszek, Teresa Zawadzka, Michał Zawadzki

”Świat zamknięty” kontra ”świat otwarty” – przykład 2



Kobieta

imię
Susan

Mężczyzna

imię
Peter

Przyjaciele

kto	kogo
John	**Susan**
John	**Billy**

Kochankowie

kto	kogo
Susan	**Billy**
Billy	**Peter**

Klasyczny przykład ilustrujący wnioskowanie w logikach deskrypcyjnych w ujęciu z: http://galaxy.eti.pg.gda.pl/katedry/kio/dydaktyka/Bazy_Wiedzy/BazyWiedzy_2.pdf, Krzysztof Goczyła, Wojciech Waloszek, Teresa Zawadzka, Michał Zawadzki

”Świat zamknięty” kontra ”świat otwarty” – przykład 2

Kobiety

imię
Susan

Mężczyźni

imię
Peter

Przyjaciele

kto	kogo
John	**Susan**
John	**Billy**

Kochankowie

kto	kogo
Susan	**Billy**
Billy	**Peter**

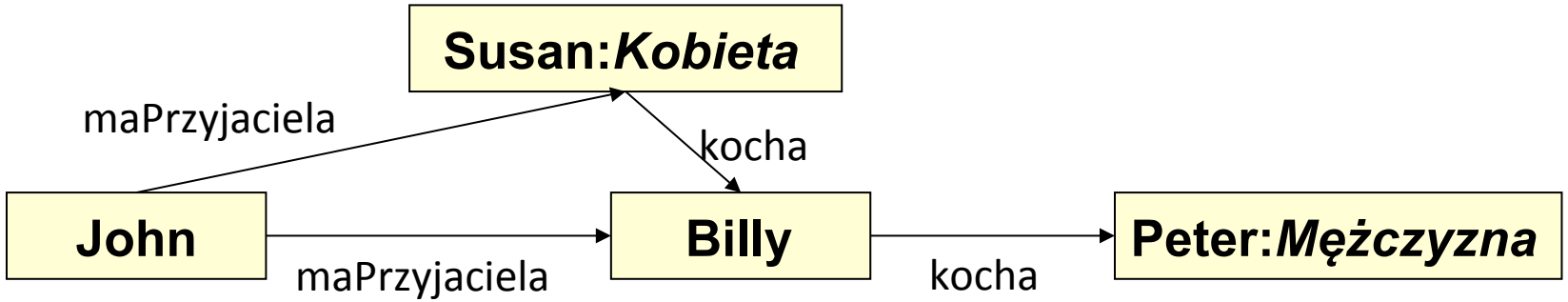
SQL

```
SELECT p.kogo
FROM Kobiety k, Mężczyźni m, Przyjaciele p, Kochankowie c
WHERE p.kogo = k.imię AND k.imię = c.kto AND c.kogo = m.imię
AND p.kto = „John”
```

Odp. NIE (pusty wynik)

Klasyczny przykład ilustrujący wnioskowanie w logikach deskrypcyjnych w ujęciu z: http://galaxy.eti.pg.gda.pl/katedry/kio/dydaktyka/Bazy_Wiedzy/BazyWiedzy_2.pdf, Krzysztof Goczyła, Wojciech Waloszek, Teresa Zawadzka, Michał Zawadzki

”Świat zamknięty” kontra ”świat otwarty” – przykład 2

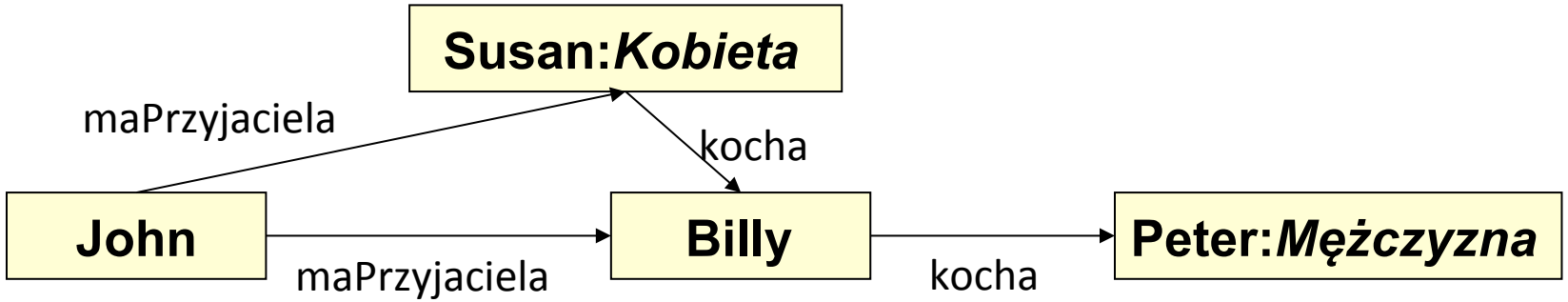


Czy John ma przyjaciółkę, która kocha mężczyznę?

Czyli: czy istnieje taka kobieta, która kocha mężczyznę i jest przyjaciółką Johna?

Klasyczny przykład ilustrujący wnioskowanie w logikach deskrypcyjnych w ujęciu z: http://galaxy.eti.pg.gda.pl/katedry/kio/dydaktyka/Bazy_Wiedzy/BazyWiedzy_2.pdf, Krzysztof Goczyła, Wojciech Waloszek, Teresa Zawadzka, Michał Zawadzki

”Świat zamknięty” kontra ”świat otwarty” – przykład 2



Czy John ma przyjaciółkę, która kocha mężczyznę?

Czyli: czy istnieje taka kobieta, która kocha mężczyznę i jest przyjaciółką Johna?

Wnioskowanie

Billy jest albo mężczyzną, albo kobietą.

Jeśli Billy jest mężczyzną: **TAK!** (ta przyjaciółka to Susan)

Jeśli Billy jest kobietą: **TAK!** (ta przyjaciółka to Billy)

Klasyczny przykład ilustrujący wnioskowanie w logikach deskrypcyjnych w ujęciu z: http://galaxy.eti.pg.gda.pl/katedry/kio/dydaktyka/Bazy_Wiedzy/BazyWiedzy_2.pdf, Krzysztof Goczyła, Wojciech Waloszek, Teresa Zawadzka, Michał Zawadzki

Brak założenia o unikalności nazw

JFK

John F. Kennedy

John Fitzgerald Kennedy

wszystkie (różne) nazwy mogą oznaczać ten sam obiekt!

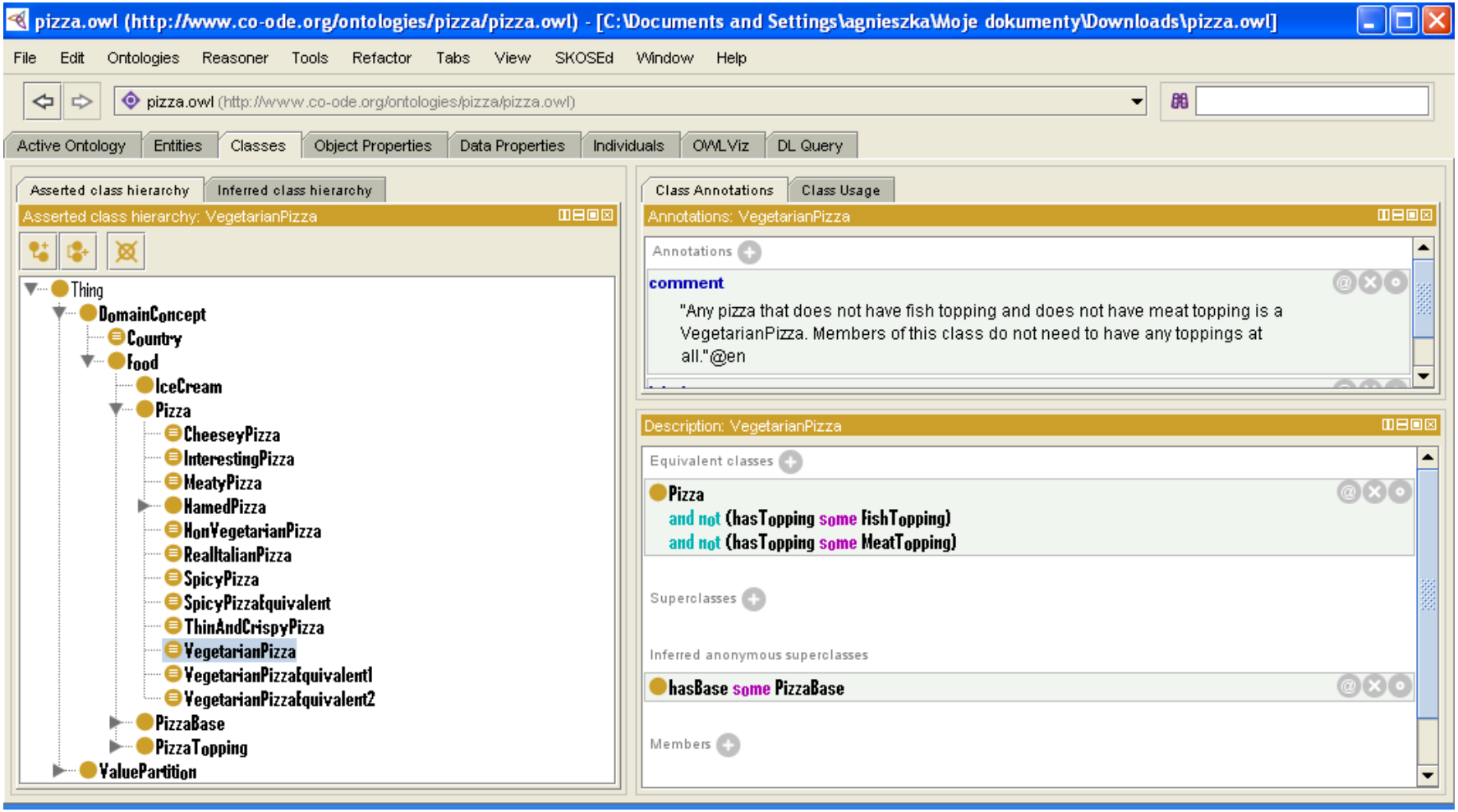
Aksjomaty o jawnej tożsamości lub rozłączności indywiduów:

```
:JFK rdf:type owl:NamedIndividual ;
     owl:sameAs :John_F_Kennedy .

[ rdf:type owl:AllDifferent ;
  owl:distinctMembers ( :JFK
                        :Donald_Trump
                      )
] .
```

Narzędzia

- Edytory
  - **Protégé, TopQuadrant Composer, NeOn Toolkit, Fluent Editor…**
- Silniki wnioskujące
  - **Pellet, FaCT++, HermiT…**
- API, zestawy narzędzi
  - **OWL-API, Jena API,…**

Protégé


pizza.owl (http://www.co-ode.org/ontologies/pizza/pizza.owl) - [C:\Documents and Settings\agnieszka\Moje dokumenty\Downloads\pizza.owl]
File Edit Ontologies Reasoner Tools Refactor Tabs View SKOSEd Window Help
pizza.owl (http://www.co-ode.org/ontologies/pizza/pizza.owl)
Active Ontology Entities Classes Object Properties Data Properties Individuals OWLViz DL Query
Asserted class hierarchy Inferred class hierarchy
Asserted class hierarchy: VegetarianPizza
Thing
DomainConcept
Country
Food
IceCream
Pizza
CheeseyPizza
InterestingPizza
MeatyPizza
NamedPizza
NonVegetarianPizza
RealItalianPizza
SpicyPizza
SpicyPizzaEquivalent
ThinAndCrispyPizza
VegetarianPizza
VegetarianPizzaEquivalent1
VegetarianPizzaEquivalent2
PizzaBase
PizzaTopping
ValuePartition
Class Annotations Class Usage
Annotations: VegetarianPizza
Annotations
comment
"Any pizza that does not have fish topping and does not have meat topping is a VegetarianPizza. Members of this class do not need to have any toppings at all."@en
Description: VegetarianPizza
Equivalent classes
Pizza
and not (hasTopping some FishTopping)
and not (hasTopping some MeatTopping)
Superclasses
Inferred anonymous superclasses
hasBase some PizzaBase
Members

Dziękuję za uwagę