Reprezentacja wiedzy: Ontologie w Semantic Web

Wojciech Jaworski

Instytut Informatyki
Uniwersytet Warszawski

Semantic Web

- Inicjatywa, która ma przyczynić się do utworzenia i rozpowszechnienia standardów opisywania treści w Internecie w sposób, który umożliwi maszynom i programom przetwarzanie informacji w sposób odpowiedni do ich znaczenia.
- Do standardów Semantic Web należy OWL służący do opisu ontologii.
- OWL stanowi rozszerzenie RDF, który z kolei korzysta z XML.
- Oprócz powyższych Semantic Web dostarcza:
  - języki regułowe, mające większą siłę wyrazu niż OWL i RDF;
  - logikę deskrypcyjną (podzbiór FOL) służącą do definiowania semantyki RDF, OWL i języków regułowych oraz wnioskowania;
  - zasoby - ontologie i narzędzia do ich przetwarzania.

Resource Description Framework (RDF)

- Język służący do reprezentowania informacji o zasobach w sieci, w szczególności do opisu metadanych.
  - Zasób jest to cokolwiek co ma tożsamość, zarówno byt fizyczny jak i pobieralny byt cyfrowy; odpowiednik indywiduum.
  - Metadane są to dane o danych, np. autor, tytuł tekstu, itp.

```
<?xml version= 1.0"?>
<rdf:RDF
    xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
    xmlns:dc='http://purl.org/dc/elements/1.1/'>
  <rdf:Description rdf:about='http://www.cat.com/docs#R20301'>
    <dc:creator rdf:resource='http://www.cat.com/auth#R051156'/>
    <dc:title>Karin Homepage</dc:title>
    <dc:date>2021-01-20</dc:date>
  </rdf:Description>

</rdf:RDF>
```

- RDF wyraża koniunkcję relacji dwuargumentowych.
- Pierwszy argument nazywany jest podmiotem a drugi przedmiotem.
- Dziedzina pierwszego argumentu to *domain*, dziedzina drugiego to *range*.

Składnia symboli w RDF

```
<?xml version='1.0'?>
<rdf:RDF
    xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
    xmlns:dc='http://purl.org/dc/elements/1.1/'>
  <rdf:Description rdf:about='http://www.cat.com/docs#R20301'>
    <dc:creator rdf:resource='http://www.cat.com/auth#R051156'/>
    <dc:title>Karin Homepage</dc:title>
    <dc:date>2021-01-20</dc:date>
  </rdf:Description>
</rdf:RDF>
```

- Stałe i nazwy relacji to identyfikatory zasobów URI (=URL+URN).
- Uniform Resource Identifier (URI): identyfikuje zasób.
- URI Reference (URIref): URI z opcjonalnym fragmentem identyfikatora.
- Namespace: przestrzeń nazw, identyfikowana przez URIref.
- Qualified Name (QName): identyfikuje nazwę w przestrzeni nazw, jest postaci *Namespace*:*Name*, rozwija się do URIref identyfikującego *Namespace* skonkatenowanego z *Name*.
- Definicje przestrzeni nazw występują na początku dokumentu, są postaci `xmlns`:*Namespace*=*URI*.

Podstawowe przestrzenie nazw

```
rdf   http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs  http://www.w3.org/2000/01/rdf-schema#
daml  http://www.daml.org/2001/03/daml+oil#
owl   http://www.w3.org/2002/07/owl#
dc    http://purl.org/dc/elements/1.1/
skos  http://www.w3.org/2009/08/skos-reference/skos.r
```

Odniesienie przedmiotowe w RDF

- Ze względu na zastosowanie w WWW wprowadzono specjalną semantykę symboli:
  - modelem, w którym interpretujemy stałe jest internet;
  - istnieje ustalona kanoniczna interpretacja stałych w modelu (algorytm znajdowania zasobu dla danego adresu), inaczej mówiąc nazwa wyznacza odniesienie przedmiotowe.
- Powyższa konwencja daje RDF zdolność do zdefiniowania obiektu poprzez wskazanie go w sieci.
- Języki naturalne mają podobną właściwość: za ich pomocą można wskazać konkretne obiekty w świecie.
- Logiki, których semantyka oparta jest na teorii modeli nie mają tej właściwości, choć surogaty są metodą jej symulacji.
- RDF może odwoływać się do samego siebie - można w nim zapisać formułę logiczną stanowiącą o sobie samej.
- RDF stanowi swój własny metajęzyk, przez co niektóre definicje są niejawne.

Składnia RDF/XML

```
<rdf:RDF>
  <rdf:Description rdf:about=subject1>
    <predicate1 rdf:resource=object1>
    <predicate2>object2</predicate2>
  </rdf:Description>
  <class1 rdf:ID=subject2>
    <predicate3 rdf:datatype=datatype1>object3</predicate3>
  </class1>
</rdf:RDF>
```

- $subject_1$, $object_1$, $subject_2$, $datatype_1$ to URIref będące surogatami dla konkretnych zasobów.
- $predicate_1$, $predicate_2$, $class_1$, $predicate_3$ to QName będące surogatami dla predykatów i typów.
- $object_2$ to zasób (tekstowy) zdefiniowany przez podanie swojej treści.
- $object_3$ to zasób typu $datatype_1$ zdefiniowany przez podanie swojej treści.

Semantyka RDF/XML

```
<rdf:RDF>
  <rdf:Description rdf:about=subject1>
    <predicate1 rdf:resource=object1>
    <predicate2>object2</predicate2>
  </rdf:Description>
  <class1 rdf:ID=subject2>
    <predicate3 rdf:datatype=datatype1>object3</predicate3>
  </class1>
</rdf:RDF>
```

$$predicate_1(subject_1, object_1) \wedge predicate_2(subject_1, object_2) \wedge$$

$$\texttt{rdf:type}(subject_2, class_1) \wedge (\exists x) predicate_3(subject_2, x) \wedge$$

$$\texttt{rdf:datatype}(x, datatype_1) \wedge \texttt{rdf:value}(x, object_3)$$

Typy, czyli elementy `rdfs:Class`

- `rdfs:Resource` najbardziej ogólny typ.
  - `rdfs:Class` typ zawierający wszystkie typy
    - ★ `rdfs:Datatype` typ zawierający typy danych.
  - `rdfs:Literal` literały np. napisy i liczby
    - ★ `rdf:PlainLiteral` literał nie mający ustalonego typu, element `rdfs:Datatype`.
    - ★ `rdf:XMLLiteral` literał którego wartością jest XML, element `rdfs:Datatype`.
  - `rdfs:Container` kolekcje (kontenery).
    - ★ `rdf:Bag` kolekcje nieuporządkowane.
    - ★ `rdf:Seq` kolekcje uporządkowane.
    - ★ `rdf:Alt` kolekcje zasobów alternatywnych.
  - `rdf:List` listy.
  - `rdf:Property` relacje (własności, atrybuty, cechy).
    - ★ `rdf:ContainerMembershipProperty` relacje należenia do kolekcji
  - `rdf:Statement` typ zawierający wszystkie formuły w języku RDF.

Atrybuty, czyli elementy `rdf:Property`

- `rdf:about` wskazuje podmiot w formule RDF, poprzez podanie jego URIref
- `rdf:ID` wskazuje podmiot w formule RDF, jest nim URIref uzyskany z połączenia bazowego URI, symbolu „#” i wartości `rdf:ID`.
- `rdf:resource` wskazuje przedmiot w formule RDF, poprzez podanie jego URIref
- `rdf:datatype` wartością jest URIref definiujący typ danych np string jakim jest wartość atrybutu

Powyższe atrybuty nie są zdefiniowane w przestrzeni nazw `rdf`.

Atrybuty, czyli elementy `rdf:Property`

- `rdf:type` $\subseteq$ `rdfs:Resource`$\times$`rdfs:Class`
  wyznacza typ podmiotu.
- `rdfs:subClassOf` $\subseteq$ `rdfs:Class`$\times$`rdfs:Class`
  relacja bycia podklasą.
- `rdfs:subPropertyOf` $\subseteq$ `rdf:Property`$\times$`rdf:Property`
  relacja bycia podrelacją.
- `rdfs:comment` $\subseteq$ `rdfs:Resource`$\times$`rdfs:Literal`
  opis.
- `rdfs:label` $\subseteq$ `rdfs:Resource`$\times$`rdfs:Literal`
  zrozumiała dla człowieka nazwa.
- `rdfs:domain` $\subseteq$ `rdf:Property`$\times$`rdfs:Class`
  dziedzina pierwszego argumentu relacji.
- `rdfs:range` $\subseteq$ `rdf:Property`$\times$`rdfs:Class`
  dziedzina drugiego argumentu relacji.
- `rdfs:seeAlso` $\subseteq$ `rdfs:Resource`$\times$`rdfs:Resource`
  dodatkowe informacje o zasobie.

Typy w OWL

- `owl:Class` typ zawierający kategorie ontologiczne; jest jednocześnie podklasą i elementem `rdfs:Class`
- `owl:Thing` typ zawierający wszystkie indywidua, elt `owl:Class`
- `owl:Nothing` typ pusty, podzbiór `owl:Thing`, elt `owl:Class`
- relacje na typach traktowanych jako zbiory

`owl:equivalentClass`(A, B)	$A = B$
`rdfs:subClassOf`(A, B)	$A \subseteq B$
`rdf:type`(v, A)	$v \in A$
`owl:disjointWith`(A, B)	$A \cap B = \emptyset$
`owl:intersectionOf`$(A, \{B_1, \ldots, B_n\})$	$A = B_1 \cap \cdots \cap B_n$
`owl:unionOf`$(A, \{B_1, \ldots, B_n\})$	$A = B_1 \cup \cdots \cup B_n$
`owl:complementOf`(A, B)	$A = \texttt{owl:Thing} \setminus B$
`owl:oneOf`$(A, \{v_1, \ldots, v_n\})$	$A = \{v_1, \ldots, v_n\}$

- relacje na indywiduach, czyli elementach `owl:Thing`

`owl:sameAs`(a, b)	$a = b$
`owl:differentFrom`(a, b)	$a \neq b$

Ekstensja i intensja

- Pojęcia (typy, klasy) stanowią abstrakcyjną specyfikację, nie są zbiorami obiektów.
- Każdemu typowi przyporządkowany jest zbiór *desygnatów* — obiektów, które należą do typu.
- Zbiór ten nazywamy *denotacją*, *ekstensją*, albo *odniesieniem przedmiotowym*.
- Na przykład denotacją nazwy „kot” jest zbiór wszystkich (przeszłych, obecnych i przyszłych) kotów.
- *Intensja* (*znaczenie*) jest to relacja wiążąca obserwowany obiekt z pojęciem.
- Intensja ma charakter algorytmiczny: dla zadanego obiektu generuje zbiór pojęć, do których ekstensji ten obiekt należy.

Ekstensjonalność

- Ekstensjonalność jest aksjomatem mówiącym, że równość ekstensji implikuje równość intensji.
- Teoria mnogości jest ekstensjonalna: jeśli dwa zbiory mają te same elementy, to są równe (identyczne).
- Teoriomnogościowa równość $A = B$ oznacza, że A i B są tym samym (dwoma nazwami na to samo).
- Rachunek lambda nie jest ekstensjonalny, chyba że dodamy do niego η-redukcję

$$(\lambda x)M(x) \xrightarrow{\eta} M$$

- Na przykład definicja $A(x) \stackrel{\text{def}}{=} B(x)$, czyli $A = (\lambda x)B(x)$ mówi, że A i procedura wywołująca B są tym samym bytem.
- Natomiast dodanie η-redukcji pozwoli wywnioskować, że z $A = (\lambda x)B(x)$ wynika $A = B$.

Nieekstensjonalna równoważność

- $(\forall x)A(x) \Leftrightarrow B(x)$, oznacza, że A i B są dwiema procedurami zwracającymi dla zadanego wejścia takie same wyniki.
- Na przykład niech

$$A(n) = \begin{cases} \sum_{k=0}^{n} k & \text{gdy } n \in \mathbb{N} \\ 0 & \text{wpp.} \end{cases}, \text{ a } B(n) = \begin{cases} \frac{n(n+1)}{2} & \text{gdy } n \in \mathbb{N} \\ 0 & \text{wpp.} \end{cases}$$

- A i B są równoważne, ale nie są sobie równe, bo w inny sposób obliczają wynik.
- Równoważność oznaczamy $A \approx B$.

Równość ekstensji

$$A \approx B \stackrel{def}{\iff} (\forall x) A(x) \Leftrightarrow B(x)$$

- W definicji równoważności występuję kwantyfikator $(\forall x)$.
- Oznacza on kwantyfikację po wszystkich możliwych bytach
- Jeżeli zdefiniujemy

$$A(n) = \sum_{k=0}^{n} k \quad \text{i} \quad B(n) = \frac{n(n+1)}{2},$$

  to A i B nie będą sobie równoważne.
- Jeśli jednak ograniczymy się do świata zawierającego wyłącznie liczby naturalne, to będą miały tą samą ekstensję, co oznaczymy jako $A \sim B$.

Paradoks Statku Tezeusza

- Identyczność przedmiotów fizycznych zależy od tego jakie cechy uznamy za istotne.
- Zgodnie z grecką legendą, podaną przez Plutarcha:

  Statek, którym Tezeusz powrócił wraz z młodymi ateńczykami, miał trzydzieści rzędów wioseł i został zachowany przez ateńczyków aż do czasów Demetriusza z Faleronu, ponieważ gdy stare deski gniły, zastępowali je nowymi, z twardszego drewna, tak że statek stał się sztandarowym przykładem wśród filozofów problemu istnienia rzeczy, które ulegają zmianie.
- Czy jeśli wymienimy wszystkie elementy jakiegoś złożonego obiektu na nowe, a więc cały obiekt zostanie zastąpiony nowym, to czy pozostaje on tym samym obiektem?
- Jeśli odpowiedź jest twierdząca, wynika z niej, że dokładna kopia jakiegoś obiektu (np. obrazu) jest tym samym obiektem.
- Jeśli odpowiedź jest przecząca, to żadne obiekty, których elementy z czasem ulegają wymianie (w szczególności ludzie), nie zachowują swojej tożsamości.

Paradoks Statku Tezeusza w amerykańskiej armii

- Amerykańska armia śledzi dystrybucję karabinów, ale nie śledzi dystrybucji ich poszczególnych części.
- Żołnierz, którego karabin jest uszkodzony może pobrać część na wymianę z minimum formalności.
- Ale zasada niekontrolowanej wymiany części umożliwiałaby kradzież karabinu poprzez pobieranie kolejnych jego części do wymiany.
- Aby zapobiec takim sytuacjom wprowadzono zasadę mówiącą, że tożsamość karabinu jest określona przez jego kolbę, która jest jedyną częścią nie podlegającą wymianie.

Identyczność bytów fizycznych

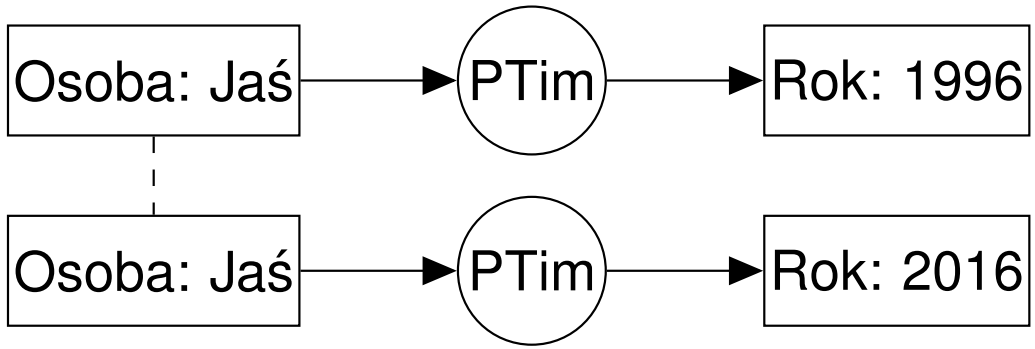
- Wszystkie byty fizyczne zmieniają się pod wpływem czasu.
- Czy Wisła dzisiaj jest tą samą rzeką co 100 lat temu?
- Czy Wisła dziś i Wisła 100 lat temu to ten sam obiekt?
- Składa się z innych atomów, ma inny kształt, ale znajduje się z grubsza w tym samym miejscu i ma te same właściwości.
- Z naszej perspektywy zmiana była stopniowa.
- Czy talerz przed i po stłuczeniu to ten sam obiekt?
- Składa się z tych samych atomów, zmieniła się konfiguracja części z nich oraz funkcjonalność obiektu.
- Z naszej perspektywy zmiana była nagła.

Identyczność bytów fizycznych

- To czy byt jest stały, czy zmienny w czasie zależy bardziej od intencji obserwatora niż od niego samego.
- Dla geografa Wisła dziś i 100 lat temu to ten sam obiekt, ale dla chemika o dwa zupełnie różne byty.
- Podobnie talerz przed i po stłuczeniu jest tym samym z punktu widzenia utylizacji odpadów, a czym innym jako wyposażenie kuchni.

Komplementarność

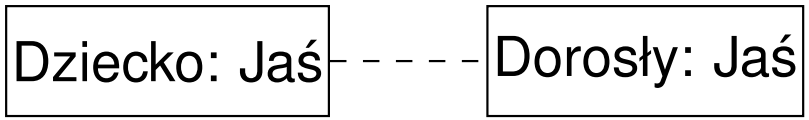
- W logice identyczność (równość) jest definiowana jako relacja łącząca obiekty, których wszystkie cechy są identyczne.
- Jaś 20 lat temu był dzieckiem, dziś jest dorosły.
- Czy Jaś jest dziś tą samą osobą co 20 lat temu?
- Z punktu widzenia prawa jednocześnie jest: domniemuje się brak zmian cech (np. to co było jego własnością 20 lat temu przez domniemanie nadal do niego należy).



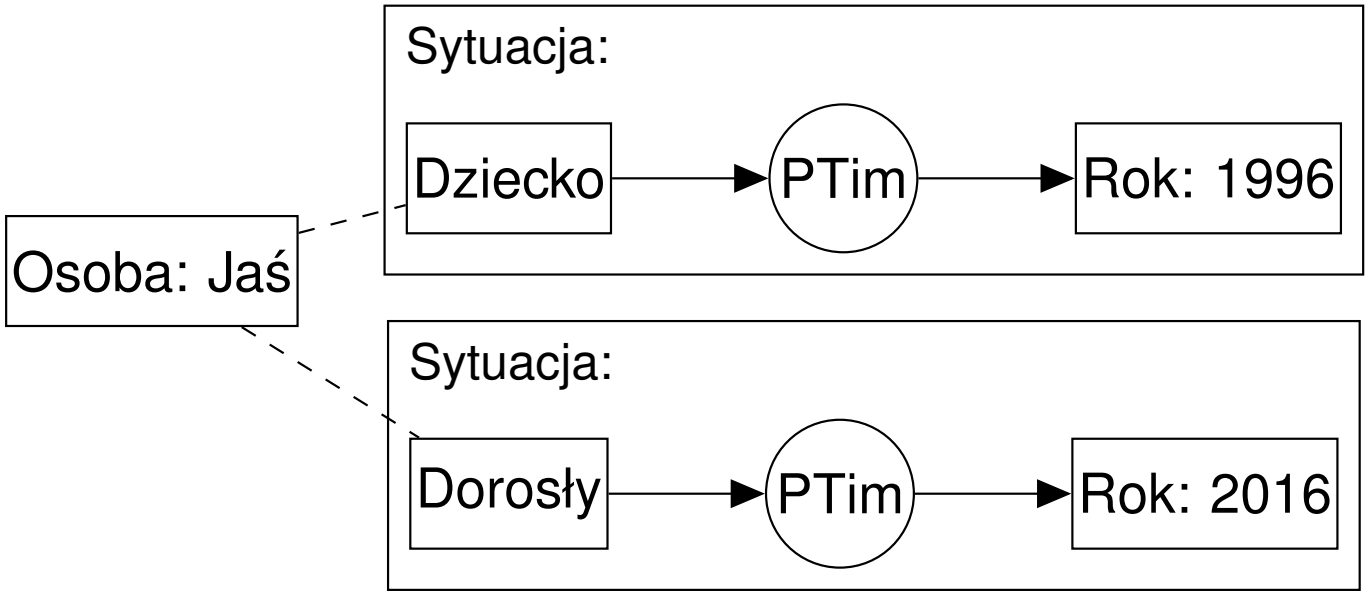
- I nie jest: konkretne cechy uległy zmianie (np. był małotelni a jest pełnoletni).
- W konkretnych przypadkach każdy z tych punktów widzenia ma zastosowanie.

Komplementarność

- Nie wolno jednak ich mieszać, gdyż prowadzi to do sprzeczności.
- Z powyższych grafów moglibyśmy wywnioskować:



- Co jest sprzeczne z tym, że dziecko nie może być jednocześnie dorosłym.
- Komplementarne sposoby opisu świata możemy mieszać jeśli jawnie wskażemy kontekst.



Hierarchia pojęć

- Równość ekstensji nie oznacza identyczności pojęć, np. „człowiek” i „bezpióry dwunóg” mają tą samą ekstensję, ale inne znaczenie, podobnie „gwiazda zaranna” i „gwiazda wieczorna”.
- Z kolei „jednorożec” ma pustą ekstensję, ale nie chcemy, by był podtypem „drzewa”.
- Hierarchię pojęć budujemy na zasadzie zawierania intensji.
- Typy będziemy rozumieć jako algorytmy, które dla zadanego obiektu na wejściu zwracają prawdę, albo fałsz.
- Kwantyfikację będziemy przeprowadzać po wszystkich możliwych bytach (obserwacjach)
- Np aby powiedzieć, że jednorożec ma róg rozpatrzymy m.in. jednorożca istniejącego w kontekście jakiejś w powieści.

Równoważność, równość i zawieranie w OWL

- `rdfs:subClassOf`(A, B) interpretujemy jako

$$(\forall x)A(x) \Rightarrow B(x) \text{ i oznaczamy } A \leq B.$$

- `owl:equivalentClass`(A, B) interpretujemy jako

$$(\forall x)A(x) \Longleftrightarrow B(x) \text{ i oznaczamy jako } A \approx B.$$

- Pojęcia równoważne mogą się różnić sposobem wyznaczania ekstensji.
- `owl:sameAs`(A, B) rozumiemy jako dwie stałe oznaczające ten sam byt i oznaczamy jako

$$A = B.$$

- Synonimy to słowa lub frazy mające to samo znaczenie; dwie nazwy tego samego pojęcia, np. dentysta i stomatolog.

Spójniki definiujące typy

- `owl:intersectionOf`$(A, \{B_1, \ldots, B_n\})$ jest definicją przecięcia typów

$$A(x) \iff (B_1 \cap \cdots \cap B_n)(x) \iff B_1(x) \wedge \cdots \wedge B_n(x).$$

- `owl:unionOf`$(A, \{B_1, \ldots, B_n\})$ jest definicją sumy typów

$$A(x) \iff (B_1 \cup \cdots \cup B_n)(x) \iff B_1(x) \vee \cdots \vee B_n(x).$$

- `owl:complementOf`(A, B) jest definicją typu dopełniającego zadany typ

$$A(x) \iff (\texttt{owl:Thing} \setminus B)(x) \iff \neg B(x).$$

Rozłączność i definiowanie przez wyliczenie

- `owl:disjointWith`(A, B) stwierdza, że A i B są rozłączne

$$A \cap B \approx \texttt{owl:Nothing}, \text{ czyli że } \neg(\exists x)A(x) \wedge B(x)$$

- `owl:oneOf`$(A, \{v_1, \ldots, v_n\})$ definiuje typ przez wyliczenie indywiduów do niego należących, czyli przez ekstensję

$$A(x) \iff x \in \{v_1, \ldots, v_n\} \iff x = v_1 \vee \cdots \vee x = v_n$$

Relacje

- `owl:ObjectProperty` typ zawierający relacje binarne na indywiduach, podklasa `rdf:Property`
- podtypy `owl:ObjectProperty`:
  `owl:AsymmetricProperty`, `owl:SymmetricProperty`,
  `owl:IrreflexiveProperty`, `owl:ReflexiveProperty`,
  `owl:FunctionalProperty`,
  `owl:InverseFunctionalProperty`,
  `owl:TransitiveProperty`
- relacje na relacjach (elementach `owl:ObjectProperty`):
  - `owl:equivalentProperty` równoważność,
  - `rdfs:subPropertyOf` bycie podrelacją,
  - `owl:inverseOf` odwrotność,
  - `owl:propertyDisjointWith` rozłączność.
- relacja pełna: `owl:topObjectProperty`,
- relacja pusta: `owl:bottomObjectProperty`.

Znaczenie: przypisywanie bytom pojęć

- Metody nadawania bytom typów:
  - wskazanie istniejącego obiektu i przypisania mu typu-etykiety (wymaga sensorów, percepcji świata i języka pozwalającego się odwołać do konkretnych bytów w świecie);
  - zdefiniowanie typu — opisanie go za pomocą prostszych pojęć, np: relacja symetryczna to relacja która spełnia aksjomat

$$(\lambda R)(\forall x)(\forall y)xRy \Longrightarrow yRx$$

- Metody sprawdzania przynależności bytów do pojęć:
  - klasyfikatory, np: algorytm rozpoznający twarze jest znaczeniem pojęcia twarz;
  - wnioskowanie w logice (za pomocą systemu dowodowego), gdy byt jest zdefiniowany aksjomatycznie;
  - badanie czy formuła definiująca jest spełniona.
- W powyższe metody można łączyć:
  - na opis pojęcia może składać się zarówno formalna definicja, jak i pewna ilość przykładów
  - rozpoznawanie twarzy może być związane z ich strukturą, wtedy mamy klasyfikatory rozpoznające poszczególne elementy twarzy i definicję, która łączy je w całość.

Świat poza ontologią

- Analizując teksty lub obrazy możemy natrafić na dane nie pasujące do przyjętej ontologii.
- Byty, dla których nie ma pojęć w ontologii nie dają się umieścić w strukturach danych i są niewidzialne dla logicznego wnioskowania.
- Dla określonych danych możemy dojść do różnych wniosków zależnie od algorytmów wyznaczania znaczenia, zakresu posiadanych pojęć i relacji miedzy nimi (ich aksjomatyzacji).
- Na przykład ontologia przestrzeni oparta na pikslach, albo siatce sześciokątnej.
- Dyskretyzacja, uciąglenie cech zmienia wyniki wnioskowania.
- Ontologia umożliwia logiczne wnioskowanie, a jednocześnie ogranicza jego zakres.
- Ten problem dotyczy też ludzi, tyle że go nie zauważają, bo nie mogą.

Restrykcje

`owl:Restriction` typy utworzone przez wybranie elementów innych typów spełniających określony warunek, podklasa `owl:Class`

```
<owl:Class rdf:about=C >
  ...
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource=P>
      ...  declaration of restriction R...
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

$$C \leq (\lambda x) R(x)$$

Restrykcja kwantyfikowana egzystencjalnie

```
<owl:Class rdf:about='HeldInEuropeConf'>
  <rdfs:subClassOf rdf:resource='Conf'/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource='heldIn'/>
      <owl:someValuesFrom
          rdf:resource='EuropeanCity'/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

$$\mathrm{HeldInEuropeConf}(x) \Longrightarrow \mathrm{Conf}(x) \wedge (\exists y)\mathrm{heldIn}(x,y) \wedge \mathrm{EuropeanCity}(y)$$

Dla zadanej relacji dwuargumentowej f i pojęcia A przeciwobraz A względem f definiujemy jako $(f^{-1}(A))(x) \Longleftrightarrow (\exists y)f(x,y) \wedge A(y)$ otrzymując

$$\mathrm{HeldInEuropeConf} \leq \mathrm{Conf} \cap \mathrm{heldIn}^{-1}(\mathrm{EuropeanCity})$$

Restrykcja kwantyfikowana uniwersalnie

```
<owl:Class rdf:about='EuropeanConf'>
  <rdfs:subClassOf rdf:resource='Conf'/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource='heldIn'/>
      <owl:allValuesFrom
          rdf:resource='EuropeanCity'/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

$$\mathrm{EuropeanConf}(x) \Longrightarrow \mathrm{Conf}(x) \wedge (\forall y)\mathrm{heldIn}(x,y) \Longrightarrow \mathrm{EuropeanCity}(y)$$

Dla zadanej relacji dwuargumentowej f i pojęcia A obraz A względem f definiujemy jako $(f(A))(y) \Longleftrightarrow (\exists x)f(x,y) \wedge A(x)$. Następnie na podstawie tego, że formuła $(\forall x)(A(x) \Rightarrow (\forall y)f(x,y) \Rightarrow B(y))$ jest równoważna formule $(\forall y)((f(A))(y) \Rightarrow B(y))$ otrzymujemy

$$\mathrm{EuropeanConf} \leq \mathrm{Conf} \qquad \mathrm{heldIn}(\mathrm{EuropeanConf}) \leq \mathrm{EuropeanCity}$$

Restrykcja określająca liczność relacji

```
<owl:Class rdf:about='Publication'>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource='noPages'/>
      <owl:cardinality
          rdf:datatype= 'xsd:nonNegativeInteger'>
          7
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

$$\text{Publication}(x) \implies \text{card}(\{y : \text{noPages}(x, y)\}) = 7$$

Analogicznie: `owl:maxCardinality` i `owl:minCardinality`.

Restrykcja określająca wartość relacji

```
<owl:Class rdf:about='ConfInBerlin'>
  <rdfs:subClassOf rdf:resource='Conf'/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource='heldIn'/>
      <owl:hasValue rdf:resource='Berlin'/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

$$\mathrm{ConfInBerlin}(x) \Longrightarrow \mathrm{Conf}(x) \wedge \mathrm{heldIn}(x, '\mathrm{Berlin}')$$

$$\begin{aligned} \mathrm{ConfInBerlin} &\leq \mathrm{Conf} \\ \mathrm{ConfInBerlin} &\leq \mathrm{heldIn}^{-1}(\{'\mathrm{Berlin}'\}) \end{aligned}$$

Przykład użycia przecięcia typów

```
<owl:Class rdf:about='TrulyEuropeanConf'>
  <owl:intersectionOf rdf:parseType='Collection'>
    <owl:Class rdf:about='Conf'/>
    <owl:Restriction>
      <owl:onProperty rdf:resource='heldIn'/>
      <owl:allValuesFrom
          rdf:resource='EuropeanCity'/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

$$\text{TrulyEuropeanConf}(x) \Longrightarrow (\text{Conf} \cap (\forall y)\text{heldIn}(x,y) \Rightarrow \text{EuropeanCity}(y))(x)$$

$$\begin{aligned} \text{TrulyEuropeanConf} &\leq \text{Conf} \\ \text{heldIn}(\text{TrulyEuropeanConf}) &\leq \text{EuropeanCity} \end{aligned}$$

Przykład użycia dopełnienia typów

```
<owl:Class rdf:about='NonEuropeanConf'>
  <owl:intersectionOf rdf:parseType='Collection'>
    <owl:Class rdf:about='Conf'/>
    <owl:Class>
      <owl:complementOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource='heldIn'/>
          <owl:allValuesFrom
            rdf:resource='EuropeanCity'/>
        </owl:Restriction>
      </owl:complementOf>
    </owl:Class>
  </owl:intersectionOf>
</owl:Class>
```

$$\text{NonEuropeanConf}(x) \Longleftrightarrow (\text{Conf} \cap$$

$$\cap \neg(\forall y)\text{heldIn}(x, y) \Rightarrow \text{EuropeanCity}(y))(x)$$