Reprezentacja wiedzy: Logika

Wojciech Jaworski

Instytut Informatyki
Uniwersytet Warszawski

Co to jest reprezentacja wiedzy?

- Wyrażenie w wiedzy o danej dziedzinie w sposób jawny, uporządkowany i zrozumiały dla komputera.
- Reprezentacja wiedzy jest zastosowaniem logiki i ontologii do konstruowania obliczalnych modeli dla zadanych dziedzin.
  - **Logika** zapewnia formalny język opisu i reguły wnioskowania
  - **Ontologia** definiuje typy bytów
  - **Obliczalne modele** (np. relacyjny model danych) umożliwiają zaimplementowanie logiki i ontologii w programach komputerowych.
- Przykłady zastosowań
  - projektowanie baz danych
  - modelowanie obiektowe
  - przetwarzanie języka naturalnego
  - organizacja wiedzy z danej dziedziny

Spis treści

Reprezentacja wiedzy w logice

- Logika pozwala reprezentować w obliczalnej postaci każdą informację, którą da się sformułować precyzyjnie w dowolnym innym języku.
- Logika pozwala wyrazić każdą informację, która może być przechowywana w pamięci komputera.
- Wiedza nie reprezentowalna w logice nie może być przetwarzana za pomocą komputera niezależnie od użytej notacji.

Rachunek zdań

Każdy samochód ma 4 koła.

Zapis powyższego zdania za pomocą rachunku zdań:

$$p$$

- najprostsza forma reprezentacji
- brak informacji o szczegółach takich jak samochód, koła, liczba 4 i relacjach między nimi

W niektórych zastosowaniach brak szczegółów jest korzystny.

Logika pierwszego rzędu

- Rozpatrzmy następujący sylogizm (starożytną regułę wnioskowania)

  Każdy samochód ma 4 koła.
  Niektóre Syrenki są samochodami.
  Zatem, niektóre Syrenki mają 4 koła.

- Możemy wyrazić go w logice w następujący sposób

$$(\forall x)(\text{samochód}(x) \Rightarrow \text{czterokołowiec}(x))$$
$$(\exists x)(\text{Syrenka}(x) \wedge \text{samochód}(x))$$
$$(\exists x)(\text{Syrenka}(x) \wedge \text{czterokołowiec}(x))$$

- Zauważmy, że zdania typu *Każde A jest B* tłumaczymy na

$$\forall x(A(x) \Rightarrow B(x))$$

  zaś zdania typu *Niektóre A są B* tłumaczymy na

$$\exists x(A(x) \wedge B(x))$$

Dobór predykatów

- Predykat „czterokołowiec" nie pozwala wyrazić wprost liczby 4, ani jej związku z kołami.
- Możemy go zastąpić predykatem dwuargumentowym:

$$\text{liczbaKół}(x, n),$$

który oznacza „liczba kół dla x wynosi n", albo „x ma n kół".

- Zdanie *Każdy samochód ma 4 koła* otrzyma formę logiczną

$$(\forall x)(\text{samochód}(x) \Rightarrow \text{liczbaKół}(x, 4))$$

co możemy odczytać jako *Dla każdego x, jeśli x jest samochodem, to liczba kół x wynosi 4*

Dobór predykatów

- W predykacie

$$\text{liczbaKół}(x, n)$$

  x odnosi się do samochodu, n odnosi się do liczby, ale nie ma zmiennej odnoszącej się do kół.
- Z punktu widzenia systemu komputerowego nazwy predykatów są pozbawionymi znaczenia etykietami.
- Jeśli w aplikacji potrzebne jest pojęcie koła oraz fakt jego istnienia w samochodzie, musimy wprowadzić bardziej szczegółowe predykaty.

Dobór predykatów

- Rozpatrzmy predykaty:

samochód(x)	x jest samochodem
koło(x)	x jest kołem
część(x, y)	y jest częścią x
zbiór(s)	s jest zbiorem
liczność(s, n)	Liczność zbioru s wynosi n
element(x, s)	x jest elementem zbioru s

- Otrzymujemy następującą formę logiczną:

$$(\forall x)(\text{samochód}(x) \Rightarrow (\exists s)(\text{zbiór}(s) \wedge \text{liczność}(s, 4)$$

$$\wedge(\forall w)(\text{element}(w, s) \Rightarrow (\text{koło}(w) \wedge \text{część}(x, w)))))$$

- Możemy ją odczytać następująco:
  Dla każdego x, jeśli x jest samochodem, to istnieje s, które jest zbiorem o mocy 4 i dla każdego w, jeśli w jest elementem s, to w jest kołem i w jest częścią x.

Definicje

- Formuła

$$(\exists s)(\text{zbiór}(s) \wedge \text{liczność}(s, 4)$$

$$\wedge(\forall w)(\text{element}(w, s) \Rightarrow (\text{koło}(w) \wedge \text{część}(x, w)))))$$

  opisuje własność posiadania 4 kół.
- Możemy za jej pomocą zdefiniować predykat czterokołowiec(x):

$$\text{czterokołowiec}(x) = (\exists s)(\text{zbiór}(s) \wedge \text{liczność}(s, 4)$$

$$\wedge(\forall w)(\text{element}(w, s) \Rightarrow (\text{koło}(w) \wedge \text{część}(x, w)))))$$

- Definicja jest skrótem notacyjnym, odpowiednikiem procedury lub makra w języku programowania.
- x — zmienną wolną formuły definicji — nazywamy parametrem formalnym.
- Rozwijając definicję dokonujemy α-konwersji: zmieniamy nazwy zmiennych lokalnych na unikalne.

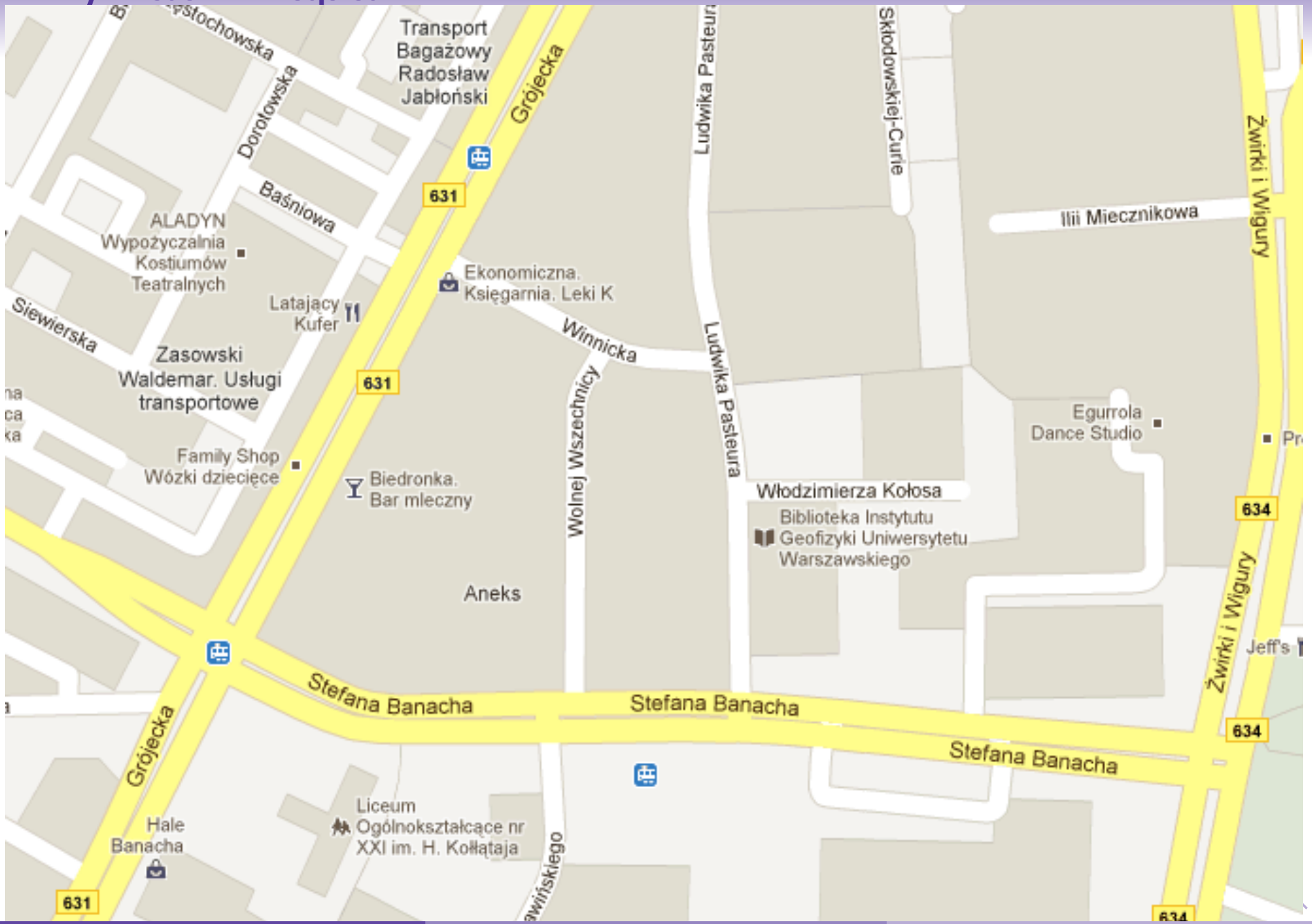
Logika i ontologia

- Nawet proste zdania w języku naturalnym niosą ze sobą wiele ukrytych informacji.
- Podanie wprost wszystkich szczegółów utrudnia odnalezienie tego co istotne w przekazie.
- Nadmiar szczegółów wynika po części z jawnego wprowadzania zmiennych.
- Poziom dokładności zależy od doboru predykatów, czyli wyboru ontologii.
- Predykaty możemy podzielić na
  - specyficzne dla danej dziedziny: samochód, koło(x)
  - niezależne od dziedziny:
    część(x, y), zbiór(s), liczność(s, n), element(x, s)

Języki specjalistyczne

- Podzbiory logiki z własną notacją i wbudowaną ontologią
- Przykłady: notacja muzyczna, rozkłady jazdy, systemy informacyjne, mapy, plany, schematy.
- Krótsze czytelniejsze niż logika (i język naturalny)
- Logika jest ontologicznie neutralną notacją, która może być użyta do reprezentowania informacji z dowolnej dziedziny poprzez dodanie predykatów opisujących tą dziedzinę.
- Języki specjalistyczne mogą być sprowadzane do postaci formuł logicznych w celu komputerowego przetwarzania zawartych w nich treści.

Przykład: mapa



Przykład: mapa

- Mamy ulice, które możemy reprezentować jako segmenty prostych i łuków.
- Ulice przecinają się na skrzyżowaniach tworząc graf.
- Nazwy ulic są etykietami krawędzi w tym grafie
- Oprócz tego mamy punkty takie jak restauracje, przystanki itp.
- Strukturalna implementacja mapy polegałaby na stworzeniu grafu ulic itp., dodanie nowego komponentu mapy wymaga zmodyfikowania struktur danych, czyli napisaniu programu od nowa.
- Obiektowo możemy stworzyć odpowiednie klasy dla elementów mapy dzieląc je na punktowe, liniowe itp., dodanie nowego komponentu może być realizowane przez dodanie podklasy, ale nie zawsze jest to możliwe, np. wprowadzenie etykiet ulic.
- Możemy też wyrazić treść mapy za pomocą logiki, dodanie nowego komponentu polega na wprowadzeniu nowego predykatu.

Logika egzystencjalno-koniunkcyjna

- Pozwala wyrażać fakty o konkretnych bytach.
- Wystarcza jako reprezentacja większości notacji specjalistycznych.
- Jest reprezentacją informacji przechowywanej w bazach danych zarówno relacyjnych jak i obiektowych.
- Nie może reprezentować generalizacji, negacji, implikacji, ani alternatywy.
- Przykład generalizacji: zasada *przystanek musi znajdować się na ulicy*.

Spis treści

Wariacje logik

Logiki mogą różnić się od siebie następującymi cechami

- składnia: wpływa na czytelność, nie zmienia siły wyrazu
- podzbiór: dozwolone operatory i sposoby ich łączenia, np logika $\exists\wedge$, czy rachunek zdań
- teoria dowodu: ograniczenie systemu dowodowego (np. logika intuicjonistyczna, liniowa), lub jego rozszerzenie (np. logika niemonotoniczna); semantyka logiki wynika wtedy z systemu dowodowego — odwrotnie niż w FOL
- teoria modeli: zmiana pojęcia prawdy, np logika trójwartościowa, czy rozmyta
- ontologia: zbiór predykatów i aksjomatów jakie traktujemy jako podstawowe, niezależne od dziedziny np. $=$, teoria mnogości, teoria czasu
- metajęzyk: język, który służy do mówienia o języku (np. FOL, gramatyka bezkontekstowa)

Logika z typami

- Nadajemy zmiennym typy:

$$(\forall x : t)P(x) \equiv (\forall x)(t(x) \Rightarrow P(x))$$

$$(\exists x : t)P(x) \equiv (\exists x)(t(x) \wedge P(x))$$

- Na przykład:

Każdy samochód ma 4 koła.
$(\forall x : \text{Samochód})\text{czterokołowiec}(x)$
Niektóre Syrenki są samochodami.
$(\exists x : \text{Syrenka})\text{samochód}(x))$
Zatem, niektóre Syrenki mają 4 koła.
$(\exists x : \text{Syrenka})\text{czterokołowiec}(x)$

Logika z typami

- Stosujemy również specjalną notację dla predykatów niezależnych od dziedziny
- Zdanie

$$(\forall x)(\text{samochód}(x) \Rightarrow (\exists s)(\text{zbiór}(s) \wedge \text{liczność}(s, 4)$$

$$\wedge(\forall w)(\text{element}(w, s) \Rightarrow (\text{koło}(w) \wedge \text{część}(x, w)))))$$

zapiszemy jako

$$(\forall x : \text{Samochód})(\exists s : \text{Zbiór})(s@4$$

$$\wedge(\forall w \in s)(\text{koło}(w) \wedge \text{część}(x, w)))))$$

Definiowanie funkcji i relacji anonimowych

- Zamiast

$$\text{czterokołowiec}(x) =$$

$$= (\exists s : \text{Zbiór})(s@4 \wedge (\forall w \in s)(\text{koło}(w) \wedge \text{część}(x, w)))$$

możemy napisać

$$\text{czterokołowiec} =$$

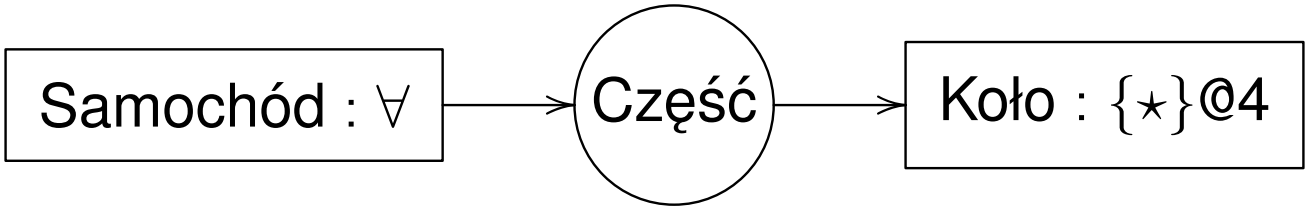
$$= (\lambda x)((\exists s : \text{Zbiór})(s@4 \wedge (\forall w \in s)(\text{koło}(w) \wedge \text{część}(x, w))))$$

- Notacja $(\lambda x)\Psi(x)$ jest definicją funkcji anonimowej, w której x jest parametrem formalnym, a Ψ formułą definiującą funkcję.
- Aplikacja $((\lambda x)\Psi(x))(a)$ powoduje wygenerowanie formuły $\Psi(a)$.
- λ będziemy stosować m.in. przy definiowaniu typów anonimowych

$$(\exists x : \text{Czterokołowiec})\ \text{samochód}(x)$$

$$(\exists x : (\lambda x)((\exists s : \text{Zbiór})(s@4 \wedge (\forall w \in s)(\text{koło}(w) \wedge \text{część}(x, w)))))\text{samochód}(x)$$

Grafy pojęć



Każdy samochód ma 4 koła

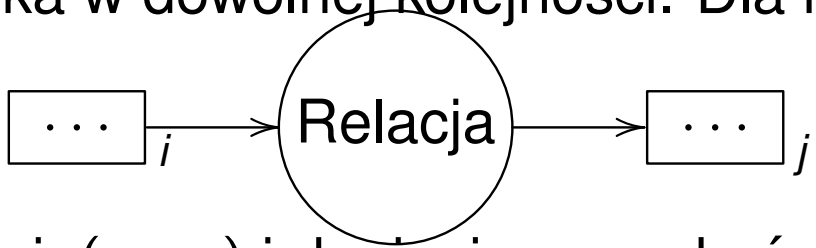
- Notacja dla logiki pozwalająca uniknąć jawnego wprowadzania zmiennych.
- Prostokąty nazywamy to pojęciami, a kółka relacjami pojęciowymi.
- Wewnątrz prostokąta znajduje się opis pojęcia (nazwa lub definicja) oraz opis odniesienia, który może zawierać nazwę, kwantyfikator $\forall$ lub specyfikację ilości, oddzielone dwukropkiem.
- Relacje pokazują w jaki sposób odniesienia pojęć są powiązane.

Semantyka grafów pojęć

- Każdy prostokąt etykietujemy unikalną liczbą naturalną.
- Konstruujemy formułę FOL przetwarzając prostokąty w kolejności rosnących etykiet:
  - jeżeli napotkamy na $\boxed{\text{Pojęcie}}_i$, dopisujemy na końcu formuły $(\exists x_i : \text{pojęcie})$
  - $\boxed{\text{Pojęcie} : \forall}_i$ tłumaczymy na $(\forall x_i : \text{pojęcie})$
  - $\boxed{\text{Pojęcie} : \{\star\}@n}_i$ tłumaczymy na

$$(\exists s_i : \text{zbiór}) s_i @ n \wedge (\forall x_i \in s_i) \text{pojęcie}(x_i)$$

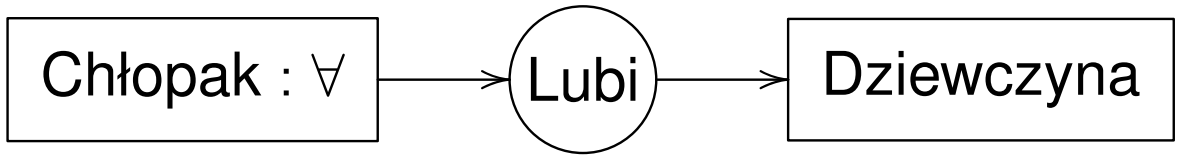
  gdzie i jest etykietą prostokąta.
- Przetwarzamy kółka w dowolnej kolejności. Dla relacji postaci



  generujemy $\wedge\text{relacja}(x_i, x_j)$ i dopisujemy na końcu formuły („$\wedge$” pomijamy przy pierwszej relacji)

Niejednoznaczność semantyki

- Jeśli ograniczymy się do logiki egzystencjalno-koniunktywnej, kolejność predykatów i kwantyfikatorów w formule nie ma znaczenia, co zapewnia jednoznaczną translację grafu na formułę FOL.
- Jeśli występuje $\forall$ translacja przestaje być jednoznaczna.
- Na przykład



możemy zinterpretować jako

$$(\forall y : \text{chłopak})(\exists x : \text{dziewczyna})\text{lubi}(y, x)$$

albo

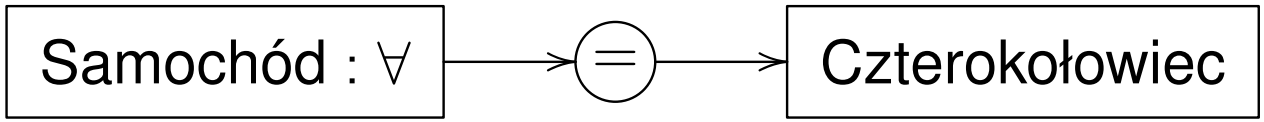
$$(\exists x : \text{dziewczyna})(\forall y : \text{chłopak})\text{lubi}(y, x)$$

- Zjawisko to występuje również w językach naturalnych (*Quantifier Scope Ambiguity*).

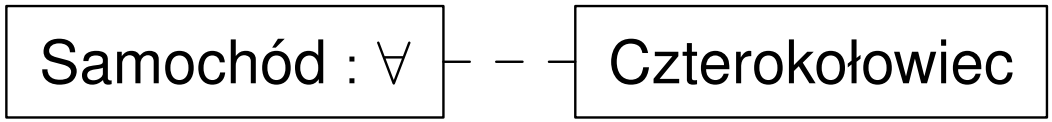
Koreferencja

Każdy samochód ma 4 koła.
$(\forall x : \text{Samochód})\text{czterokołowiec}(x)$

- W powyższym przykładzie dwa pojęcia mają ten sam desygnat (odniesienie)
- Możemy zapisać to za pomocą grafu jako:



- Co odpowiada formule:
  $(\forall x : \text{Samochód})(\exists y : \text{czterokołowiec})x = y$
- Możemy też skorzystać ze skrótu notacyjnego:



Knowledge Interchange Format (KIF)

- Logika z typami mająca składnię w stylu Lisp'a korzystająca jedynie z symboli ASCII.
- Przykład

```
(forall (?x samochod)
  (exists (?s set)
    (and (count ?s 4)
      (forall (?w in ?s) (and (kolo ?w)
            (part ?x ?w))) )))
```

Logiki modalne

- Logika posiadająca operatory wyrażające modalności.
- $\Diamond\varphi$ oznacza, że φ jest możliwe.
- $\Box\varphi$ oznacza, że φ jest konieczne.
- Semantykę operatorów modalnych można w skrócie wyrazić następująco:
  - mamy zbiór możliwych światów, z wyróżnionym światem aktualnym;
  - φ oznacza, że φ jest prawdziwe w świecie aktualnym;
  - $\Diamond\varphi$ oznacza, że istnieje świat, w którym φ jest prawdziwe;
  - $\Box\varphi$ oznacza, że φ jest prawdziwe w każdym możliwym świecie.
- Istnieją różne aksjomatyzacje modalności mające różną siłę wyrazu.
- W bazach danych
  - informacje przechowywane w bazie są uznawane za prawdziwe w świecie aktualnym
  - więzy są uznawane za koniecznie prawdziwe.

Logiki wyższego rzędu

- Logika pierwszego rzędu pozwala kwantyfikować po indywiduach.
- Logika drugiego rzędu pozwala kwantyfikować po relacjach na indywiduach.
- Logika trzeciego rzędu pozwala kwantyfikować po relacjach na relacjach na indywiduach, itd ...
- Na przykład, aksjomat indukcji dla liczb naturalnych:

$$(\forall P : \text{Predykat})((P(0) \wedge (\forall n \in \mathbb{N})(P(n) \Rightarrow P(n+1)) \Rightarrow (\forall n \in \mathbb{N})P(n))$$

- Logiki wyższego rzędu można zdefiniować za pomocą FOL zaopatrzonej w ontologię dla relacji (np. ZFC)

Metajęzyk

- Język używany do mówienia o języku.
- Na przykład język naturalny używany do definiowania logiki pierwszego rzędu.
- Rozpatrzmy zdanie: *Zdanie „Jest prawdą, że Jan jest wysoki” znaczy to samo co „Jan jest wysoki.”*
- Cudzysłowy wskazują na użycie metajęzyka, podobnie słowo *prawdą*.
- Zatem, cała rozpatrywana wypowiedź należy do metametajęzyka, a slajd który o niej mówi do metametametajęzyka.
- W logice możemy wprowadzić specjalny predykat, który pełni rolę cudzysłowów i łączy język z metajęzykiem.

Spis treści

Nazwy własne

- Wyrażenie *kot Mruczek* możemy reprezentować jako

$$\text{kot(Mruczek)}$$

- Jest to prosta reprezentacja, ale zawodna, gdy
  - nazwa jest niejednoznaczna, np. wiele kotów ma na imię „Mruczek”
  - jeden obiekt ma wiele nazw, np. kot znany w jednym domu jako „Mruczek”, jest w drugim domu nazywany „Filon”.
- Rozwiązaniem są **surogaty**, czyli jednoznaczne identyfikatory przypisywane indywiduom.
- W Polsce w przypadku ludzi surogatem jest PESEL.
- Nazwy własne stają się zwykłymi cechami obiektów:

$$\text{Imię}(\text{„Mruczek”}) \wedge (\exists x : \text{Kot})\text{nazywaSię}(x, \text{„Mruczek”})$$

Konwencja unikalnych nazw

- Jeżeli stałe a i b mają różne nazwy to $a \neq b$.
- KIF ma dwa typy stałych dla jednych zakłada się unikalność, dla drugich nie.
- W grafach pojęć stałe nie są traktowane jako unikalne, ale symbole postaci #liczba są unikalne i służą za surogaty.
- Węzeł grafu postaci $\boxed{\text{Pojęcie} : \text{Nazwa}}$ zapisujemy w języku logiki jako

  $$\text{Pojęcie}(\text{Nazwa})$$

  i oznacza to, że obiekt o nazwie „Nazwa” jest typu „Pojęcie”.
- $\boxed{\text{Imię} : \text{„Mruczek”}}$ to imię o brzmieniu „Mruczek”
- $\boxed{\text{Kot} : \#2563}$ to kot skatalogowany w systemie pod identyfikatorem #2563.

Skolemizacja

- Jeśli wprowadzimy surogat dla każdego indywiduum, o którym wiemy, że istnieje, będziemy mogli usunąć wszystkie $\exists$ zastępując wystąpienia kwantyfikowanych zmiennych surogatami.
- Jest to instancjacja egzystencjalna, zwana też skolemizacją.
- Skolemizacja zmienia nieco znaczenie, bo zmienna kwantyfikowana egzystencjalnie nie wskazuje jednoznacznie obiektu.
  - Jeśli wiemy, że
    $$(\exists x)\mathrm{kot}(x)$$
    to na pytanie „Jaki jest kolor futra tego kota?” możemy opowiedzieć podając kolor futra dowolnego istniejącego kota.
  - Z kolei, jeśli wiemy, że
    $$\mathrm{kot}(\#2563)$$
    to odpowiadając na pytanie „Jaki jest kolor futra tego kota?” musimy podać kolor futra obiektu wskazanego przez $\#2563$.

Rzeczowniki pospolite

- Zdanie *Koty lubią ryby* możemy zapisać w logice jako

$$(\forall x : \text{Kot})(\forall y : \text{Ryba})\text{lubi}(x, y)$$

- Zaś *Kot lubi rybę* możemy zapisać jako

$$(\exists x : \text{Kot})(\exists y : \text{Ryba})\text{lubi}(x, y)$$

- Rzeczownikom (nazwom) pospolitym odpowiadają typy, albo predykaty.

Typ „Typ”

- Logika pierwszego rzędu pozwala kwantyfikować jedynie po indywiduach.
- Aby móc kwantyfikować po typach wprowadzimy typ „Typ”.
- Zmiennymi typu „Typ” będą typy.
- Jest on typem drugiego rzędu.

Koty to ssaki. Kot jest ssakiem. Koty są ssakami

$$(\forall x : \text{Kot})\text{ssak}(x)$$

Koty to gatunek.

$$(\exists x : \text{Gatunek})(\text{typ}(\text{Kot}) \wedge x = \text{Kot})$$

Mruczek jest kotem.

$$(\exists x : \text{Kot})x = \text{Mruczek}$$

kot Mruczek.

$$\text{kot}(\text{Mruczek})$$

Predykat „rodzaj”

- Wprowadzimy też predykat „rodzaj(x, t)”, który oznacza, że obiekt x jest typu t.
- Stosowanie predykatu „rodzaj” do określania typów nazywamy reifikacją:

$$(\forall x)(\forall y)(\text{rodzaj}(x, \text{Kot}) \wedge \text{rodzaj}(y, \text{Ryba})) \Rightarrow \text{lubi}(x, y)$$

- Nazwy typów mogą mieć aliasy podobnie jak nazwy indywiduów, więc w praktyce również są stosowane dla nich surogaty.
- Możemy teraz na przykład zdefiniować typy niepuste mówiąc, że *Każdy typ jest niepusty, gdy zawiera przynajmniej jedno indywiduum*:

$$(\forall t : \text{TypNiepusty})(\exists x : \text{Indywiduum})\text{rodzaj}(x, t)$$

Reprezentacja miar

Waldemar i Zofia otrzymują po 30000zł honorarium.

$$(\exists x, y : \text{Zarobić})(\exists z, w : \text{Honorarium})$$

$$(\text{Osoba}(\text{Waldemar}) \wedge \text{Osoba}(\text{Zofia}) \wedge$$

$$\text{agnt}(x, \text{Waldemar}) \wedge \text{przedmiot}(x, z) \wedge \text{ilość}(z, 30000\text{zł}) \wedge$$

$$\text{agnt}(y, \text{Zofia}) \wedge \text{przedmiot}(y, w) \wedge \text{ilość}(w, 30000\text{zł}))$$

- Powyższa formuła nie uwzględnia faktu, że honoraria są rozłączne (co podkreśla przyimek „po”).
- Możemy zawyrokować rozłączność na następujące sposoby:
  - dodać do formuły warunek: $z \neq w$
  - ustanowić surogaty dla honorariów i ustanowić konwencję mówiącą, że każdy surogat wskazuje inne indywiduum.