

Znalostní technologie I

ZÁKLADNÍ STAVEBNÍ PRVKY ONTOLOGIÍ

OBSAH PŘEDNÁŠKY

- × Typy tříd
- × Třídy a jejich vymezení
- × Sémantika tříd
- × Odvozovací stroje
- × Princip otevřeného světa
- × Princip jedinečného jména

TYPY TŘÍD

TŘÍDA

- ✗ Základní konstrukční prvek
- ✗ Slouží jako předpis pro jedince (nikoliv objekty)
- ✗ Analogie s třídou z objektového programování
- ✗ Protégé implicitně obsahuje nejméně jednu základní, zcela obecnou třídu owl:Thing, od níž jsou odvozeny všechny ostatní třídy

ROZDĚLENÍ TŘÍD

Název třídy	Anglický ekvivalent
Primitivní třída	Primitive Class
Definovaná třída	Definable Class
Přídavná třída	Modifier Class
Anonymní třída	Anonymous Class
Testovací třída	Test Class

PRIMITIVNÍ TŘÍDA (PRIMITIVE CLASS)

- ✗ Výchozí typ třídy
- ✗ Obsahuje:
 - + **Název** (musí být zřejmé, jakou část reality reprezentuje)
 - + **Popis** (vlastnosti, které ji charakterizují)
- ✗ V přirozeném jazyce se obvykle jedná o podstatné jména
- ✗ Modeluje se jako neúplná třída – pouze omezení v bloku *necessary*

DEFINOVANÁ TŘÍDA (DEFINABLE CLASS) (1)

- ✗ Třída vymezená přesnou definicí
- ✗ Může u ní být uveden **popis**
- ✗ Obecně nemusí existovat pouze jeden správný popis definované třídy, věci lze definovat různými způsoby
- ✗ Def. třída vytváří kategorie, do nichž se po aplikaci klasifikátoru (odvození) odvodí třídy odpovídající definici

DEFINOVANÁ TŘÍDA (DEFINABLE CLASS) (2)

- ✗ Obvykle odpovídá konceptu, k jehož vymezení jsou použity jiné koncepty, např.
 - + Stařec (člověk, který je starý)
 - + Student (člověk, který studuje)
 - + Matka (člověk ženského pohlaví, který má alespoň jednoho potomka)
- ✗ Modeluje se pomocí omezení v bloku *necessary and sufficient*, případně dále i pomocí omezení v bloku *necessary*

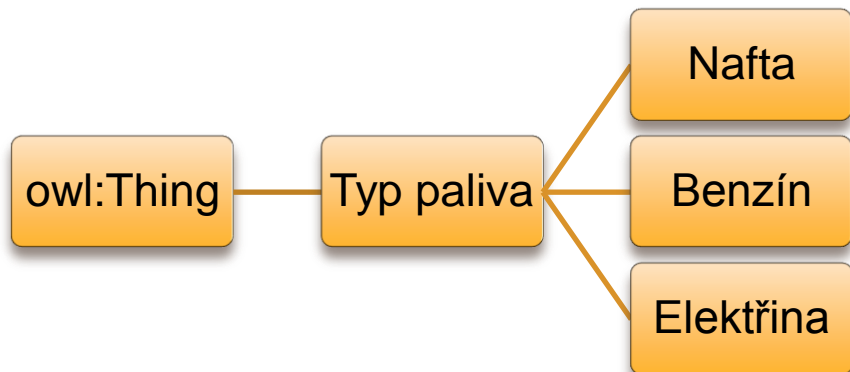
PŘÍDAVNÁ TŘÍDA (MODIFIER CLASS)

- ✗ Je typem třídy, která slouží k upřesnění jiných pojmů
- ✗ Obvykle v přirozeném jazyce odpovídá
 - + Přídatnému jménu (šedý, mladý, náročné, dlouho, levně) – otázka JAKÝ?
 - + Příslovci – otázka JAK?
- ✗ Modeluje se použitím návrhového vzoru „Rozklad třídy na podtřídy“

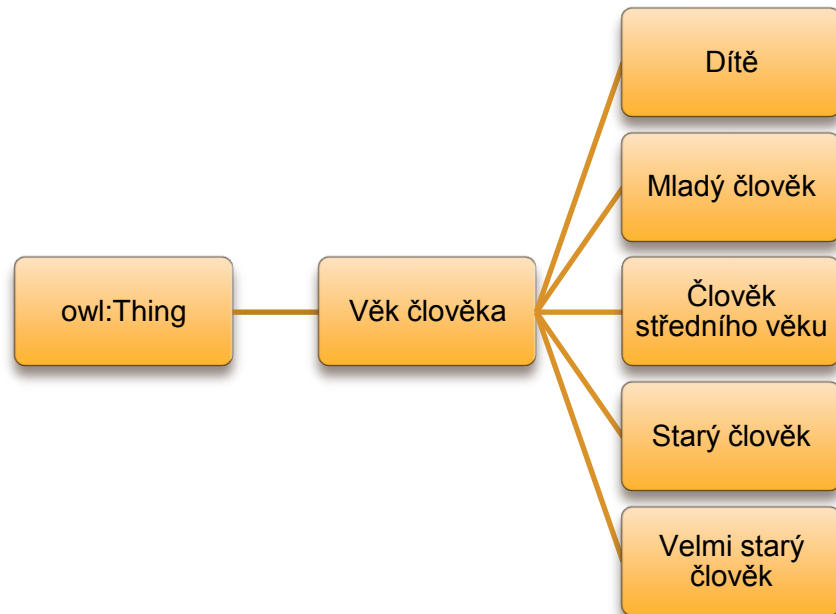
ROZKLAD TŘÍDY NA PODTŘÍDY

- ✗ Nadřazená třída představuje dimenzi (nějakou vlastnost, atribut), kterou chceme modelovat
- ✗ Podtřídy představují možné hodnoty, kterých tato dimenze může nabývat
- ✗ Hodnoty dimenze jsou navzájem různé, musí být disjunktní

NESPORNÝ PŘÍKLAD



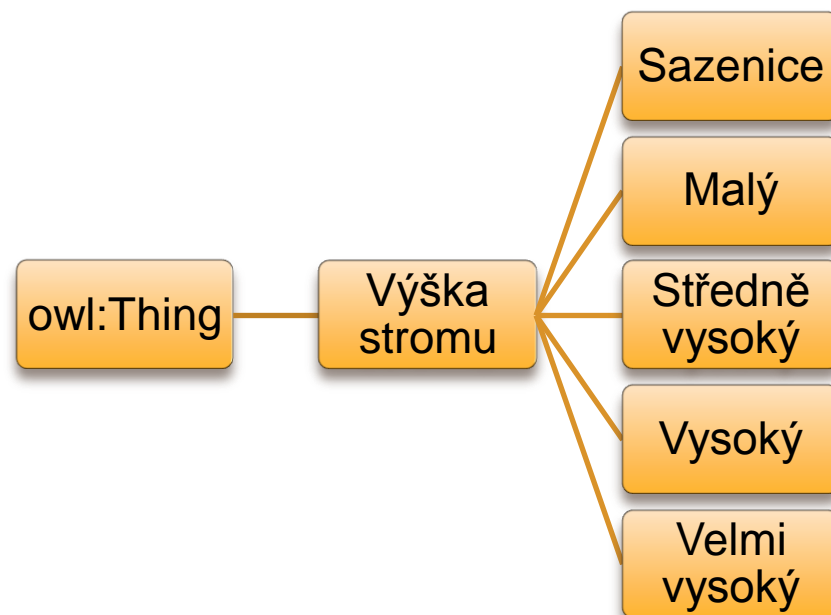
SPORNÝ PŘÍKLAD



PŘÍKLADY PŘÍDAVNÝCH TŘÍD

DŮLEŽITOST KOMENTÁŘŮ

- ✗ Komentáře ujasňují chápání ontologických konstruktů
- ✗ Měly by zodpovídat otázku: Co tím autor myslel?
- ✗ Stručné, jasné, bezesporné vyjadřování
- ✗ Pozor na subjektivitu v chápání pojmů – komentáře by ji měly odstranit
- ✗ Řadě potíží se dá předejít vhodným pojmenováním tříd



ANONYMNÍ TŘÍDA (ANONYMOUS CLASS)

- ✗ Je bezejmenná (odtud název)
- ✗ Vymezena pomocí logických výrazů:
 - + Sjednocení (OR)
 - + Průnik (AND)
 - + Doplněk (NOT)
 - + Výčtem (specifikuje členství v dané třídě)
 - + Omezeními (vztahují se k použité vlastnosti)
- ✗ Třída nemá explicitní vyjádření v hierarchii tříd
- ✗ Do anonymní třídy spadají jedinci, kteří splňují určitou logickou definici

VYMEZENÍ TŘÍDY

- ✗ Tři možné přístupy
 - + Vymezení popsáním (částečné)
 - + Vymezení definováním (úplné)
 - + Vymezení výčtem (úplné)
- ✗ Důležité je mít jasno v otázkách:
 - + Jaký je význam třídy?
 - + Kteří jedinci do třídy patří a kteří ne?
 - + Jak posoudit, zda jedinec do třídy patří?
- ✗ Vymezení určuje význam třídy v kontextu jiných tříd

VYMEZENÍ TŘÍDY POPSÁNÍM

- ✗ Třída je vymezena použitím jejích charakteristik
- ✗ Je uveden výčet vlastností, které třída má
- ✗ Hrozí rozpor mezi výčtem charakteristik a tím, co se nám do třídy řadí – popsání třídy nemusí nutně stačit pro jednoznačné a správné zařazení jedince do třídy
- ✗ Výčet vlastností musí být dosti detailní, aby zařazení fungovalo správně

DŮLEŽITOST POPISU TŘÍD

- ✗ Jazyk OWL je založený na deskripční logice
- ✗ Popis třídy dovoluje ověřit, zda jedinec skutečně do dané třídy patří, zda splňuje daná kritéria
- ✗ Uvažování s použitím klasifikátoru – nástroje pro odvozování implicitních souvislostí v rámci ontologie

VYMEZENÍ TŘÍDY DEFINOVÁNÍM

- ✗ Jednoznačné zařazení jedince do třídy

- ✗ Příklad:

„Každá sýrová pizza má sýrovou přísadu“

„CheesePizza hasTopping some CheeseTopping“

Definice třídy

Jestliže je něco sýrová pizza, pak

patří do třídy Pizza a zároveň

patří do třídy všech jedinců, kteří obsahují sýrovou přísadu

\leq

Jestliže je něco Pizza a obsahuje sýrovou přísadu, pak patří do třídy sýrová pizza

- ✗ Tzn. třída je vymezená popsáním + zavádí se ekvivalence







„Sýrová pizza je totéž, co pizza se sýrovou přísadou“

VYMEZENÍ TŘÍDY VÝČTEM

- ✗ Je uveden výčet jedinců, kteří do třídy patří
- ✗ Tím je třída vymezena jednoznačně, není sporu co do třídy patří a co ne

SÉMANTIKA TŘÍD – TYPY OMEZENÍ

- ✗ Třída sama o sobě nenese sémantickou hodnotu – je pouze symbolem
- ✗ Sémantickou hodnotu přinášejí **omezení hodnot vlastností**, které mezi třídami existují (viz vpravo)

	someValuesFrom
	allValuesFrom
	hasValue
	minCardinality
	maxCardinality
	Cardinality

OMEZENÍ SOME VALUES FROM

- ✗ Alespoň jedna hodnota vlastnosti musí být uvedeného typu
- ✗ Mohou existovat i další hodnoty
- ✗ Příklad: Sýrová pizza je taková pizza, že alespoň jedna její přísada je sýrová
Z toho mj. plyne, že sýrová pizza má alespoň jednu přísadu (sýr)



OMEZENÍ ALL VALUES FROM

- ✗ Všechny hodnoty vlastnosti musí být uvedeného typu
- ✗ Pozor na triviální splnění omezení:
 - + Jestliže jedinec nemá danou vlastnost, splňuje omezení **allValuesFrom**, což znamená
 - + Ze splnění omezení **allValuesFrom** neplyne splnění omezení **someValuesFrom**
- ✗ Příklad: vegetariánská pizza je taková pizza, že všechny její přísady jsou zeleninové nebo sýrové
 - + Nevíme, zda nějakou přísadu vůbec má



OMEZENÍ HAS VALUE

- ✗ Přiřazuje konkrétního jedince/hodnotu
- ✗ Lze přiřadit víc konkrétních jedinců/hodnot (neimplikuje funkčnost vlastnosti)
- ✗ Podobné omezení jako **someValuesFrom**, ale **pro jedince a primitivní (datotypové) hodnoty**
- ✗ Příklad: Pro třídu „přísada Mozzarella“ je alespoň jedna hodnota vlastnosti „krajina původu“ Itálie

The screenshot shows a SPARQL query editor interface. At the top, there are icons for query, results, insert, and a help icon. Below the icons, the text "Asserted Conditions" is displayed. The main area shows a list of conditions for a class named "CheeseTopping". The conditions are:

- hasCountryOfOrigin **has** Italy
- hasSpiciness **some** Mild

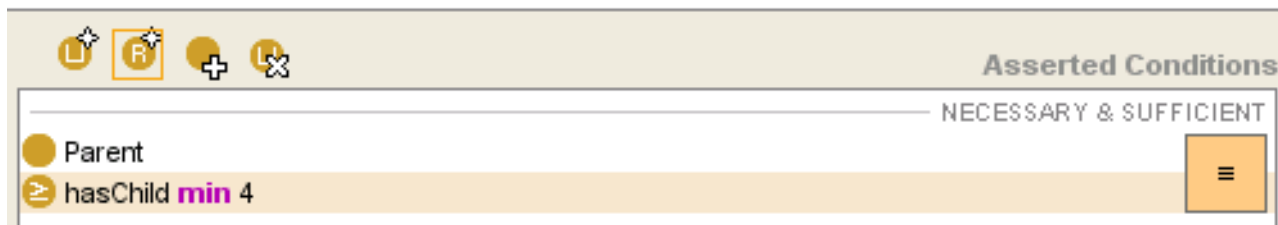
On the right side, there are three yellow buttons with the symbol \sqsubseteq (subset or equal to), indicating the logical relationship between the conditions.

OMEZENÍ CARDINALITY

- ✗ Vlastnost musí nabývat alespoň/právě/nanejvýš x hodnot
- ✗ Příklady:
 - + Šťastný rodič je rodič, který má právě dvě děti (tj. kardinalita vlastnosti hasChild je přesně dva)



- + Utahaný rodič je rodič, který má alespoň čtyři děti (tj. kardinalita vlastnosti hasChild je alespoň čtyři)



LOGICKÉ/MNOŽINOVÉ VLASTNOSTI TŘÍD

× Definice třídy použitím jiných tříd

+ sjednocení (or) 

+ průnik (and) 

+ doplněk (not) 

× Lze libovolně vnořovat

„(A and (B or C) and not D)“

RELACE MEZI TŘÍDAMI – SJEDNOCENÍ (OR)

- ✗ Třída všech jedinců, kteří patří do třídy A **nebo (or)** do třídy class B (nebo do obou)



RELACE MEZI TŘÍDAMI – PRŮNIK (AND)

- ✗ Třída všech jedinců, kteří patří do třídy A a zároveň (and) do třídy B

The screenshot shows a logic editor interface. At the top, there is a toolbar with icons for class creation (L, R), addition (+), and deletion (X). Below the toolbar, a class hierarchy is displayed: a yellow circle labeled 'Pizza' is the parent, and a yellow circle labeled 'E' labeled 'hasTopping' is a child. The 'hasTopping' property is further defined as 'some (PizzaTopping and (hasSpiciness some Hot))'. On the right side, under the heading 'Asserted Conditions', the text 'NECESSARY & SUFFICIENT' is displayed. A small orange button with a hamburger menu icon is located at the bottom right of the interface.

IMPLICITNÍ PRŮNIK

- ✗ Jestliže je třída vymezená několika podmínkami, pak tyto podmínky kombinujeme logickou operací průnik

		NECESSARY
NamedPizza		\sqsubseteq
hasTopping some MozzarellaTopping		\sqsubseteq
hasTopping some TomatoTopping		\sqsubseteq

RELACE MEZI TŘÍDAMI – DOPLNĚK (NOT)

- ✖ Třída všech jedinců, kteří nepatří do nějaké třídy
- ✖ Příklad: nevegetariánská pizza je pizza, která není vegetariánská

		NECESSARY
NamedPizza		
hasTopping some MozzarellaTopping		
hasTopping some TomatoTopping		

RELACE MEZI TŘÍDAMI – DISJUNKCE (DISJOINT)

- ✗ Jedinci patřící do dvou vzájemně disjunktních tříd nemohou být v obou těchto třídách současně (bez ohledu na vzájemnou podobnost těchto tříd – viz dole)

Margherita Pizza

Americana Pizza

Asserted Conditions	
NECESSARY & SUFFICIENT	
NECESSARY	
NamedPizza	<input type="checkbox"/>
hasTopping some MozzarellaTopping	<input type="checkbox"/>
hasTopping some TomatoTopping	<input type="checkbox"/>

Asserted Conditions	
NECESSARY & SUFFICIENT	
NECESSARY	
NamedPizza	<input type="checkbox"/>
hasTopping some MozzarellaTopping	<input type="checkbox"/>
hasTopping some TomatoTopping	<input type="checkbox"/>
hasTopping some PepperoniTopping	<input type="checkbox"/>

PODMÍNKY PŘÍSLUŠNOSTI DO TŘÍDY

✗ **Nutné podmínky** (Necessary Conditions)

(Primitivní / částečné třídy)

“Jestliže něco patří do třídy X,
pak splňuje podmínky...”



✗ **Nutné a dostačující podmínky** (Necessary & Sufficient Conditions):

(Definované / úplné classes)

“Jestliže něco patří do třídy X,
pak splňuje podmínky...”

A ZÁROVEŇ

“Jestliže něco splňuje podmínky...,
pak patří do třídy X.”



ODVOZOVÁNÍ V ONTOLOGII - KLASIFIKACE

ODVOZOVACÍ STROJE (KLASIFIKÁTORY)

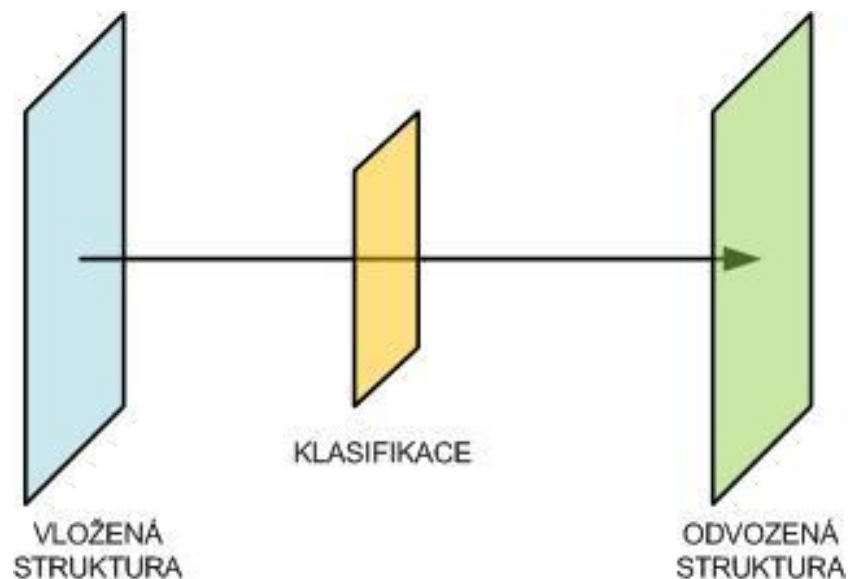
- ✗ Odvozovač je usuzovací program, který je schopen z ontologie odvodit informace/znalosti, které v ní nejsou explicitně vyjádřeny
- ✗ Jiné pojmenování:
 - + klasifikátor, odvozovací stroj (cz)
 - + reasoner, classifier, inference engine (en)

VYUŽITÍ KLASIFIKÁTORŮ

- ✗ Strojové zpracování znalostí bez asistence člověka (sémantický web)
- ✗ Zvyšování kvality ontologie:
 - + Smysluplnost – všechny pojmenované třídy mohou mít instance
 - + Korektnost – zachycení znalostí doménových expertů
 - + Minimální redundance – žádná neúmyslná synonyma
 - + Bohatá axiomatizace – detailní popis tříd

ÚČEL KLASIFIKÁTORU (1)

- ✗ Usnadnění tvorby ontologie
- ✗ Klasifikací jsou odvozeny nové vztahy, které nebyly přímo vloženy tvůrcem



ÚČEL KLASIFIKÁTORU (2)

✗ Klasifikátor provádí:

- + Kontrolu konzistentnosti (zda si jednotlivé části ontologie logicky vzájemně neodporují)
- + Klasifikaci tříd (kontrola příslušnosti tříd, tj. isA relací – tvoří hierarchickou strukturu ontologie)
- + Klasifikace jedinců

VLOŽENÁ STRUKTURA



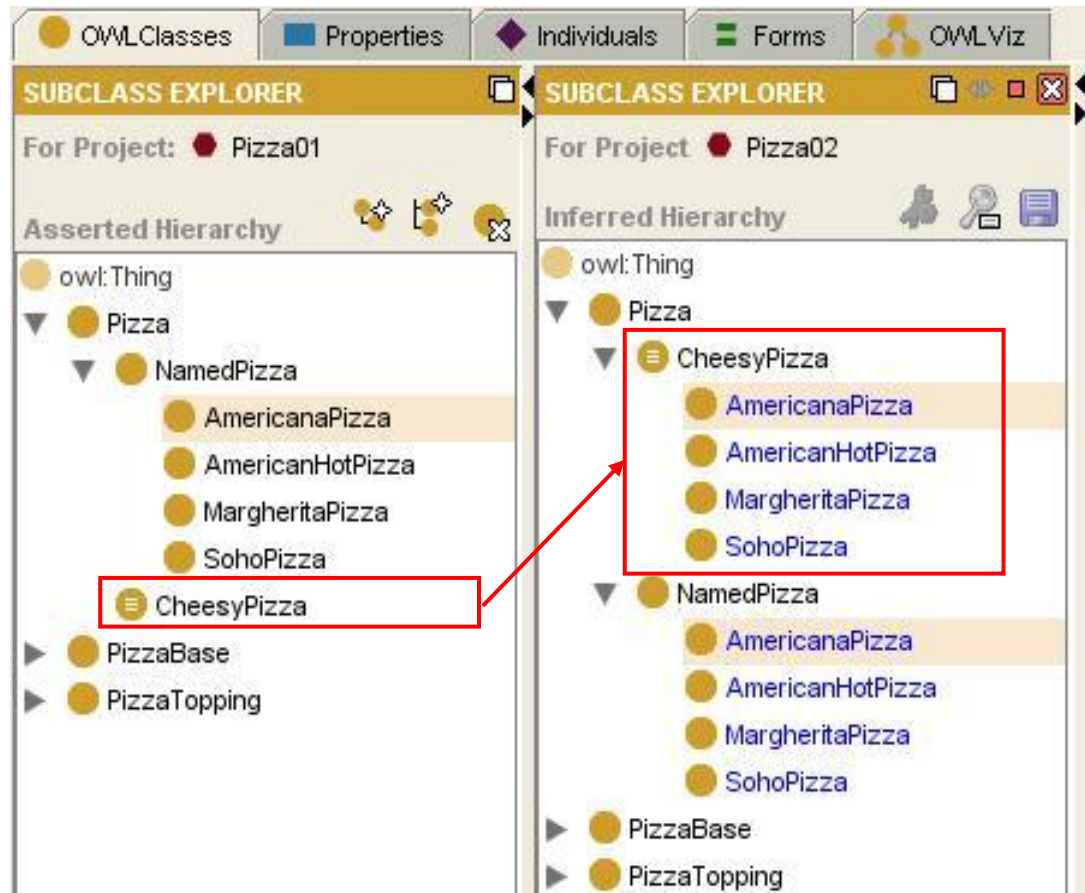
ODVOZENÁ STRUKTURA



APLIKACE KLASIFIKÁTORU

Input: vložené definice tříd

Output: odvozené vztahy „je podtřídou“



DALŠÍ VLASTNOSTI ONTOLOGICKÉHO MODELU

PRINCIP UZAVŘENÉHO SVĚTA

PRINCIP CWA (CLOSE WORLD ASSUMPTION)

- ✗ Někaké tvrzení o světě může být pravdivé, nepravdivé nebo nerozhodnutelné.
- ✗ V uzavřeném světě nepředpokládáme, že budou zjišťovány nové skutečnosti, které by mohly změnit náš aktuální pohled na svět.
- ✗ Předpoklady uzavřeného světa (Prolog):
 - + „Jestliže o tvrzení X neumím prokázat, že je pravdivé, pak přijmu závěr, že tvrzení X je nepravdivé.“
 - + „Jestliže se neprokáže nevina, pak je obžalovaný vinen.“

PRINCIP OTEVŘENÉHO SVĚTA

PRINCIP OWA (OPEN WORLD ASSUMPTION)

- V otevřeném světě předpokládáme, že časem bude možné přidat další informace k aktuálnímu stavu zkoumání dané domény
- ✗ Předpoklad otevřeného světa:
 - + „Jestliže tvrzení X (obžalovaný je nevinen) není pravdivé, pak tvrzení X může být jak nepravdivé (tudíž obžalovaný je vinen) tak nerozhodnutelné.“
 - + „Jestliže se neprokáže nevina, pak nelze usuzovat, že obžalovaný je vinen.“
- ✗ Opatrný přístup
 - + „Jestliže je nějaké tvrzení nerozhodnutelné, a později získám znalosti k určení jeho pravdivosti, není nutné přehodnotit předchozí závěry.“

PRINCIP OTEVŘENÉHO SVĚTA A KLASIFIKACE

- ✗ PizzaMargherita (Mozzarella, rajče) může být klasifikována jako vegetariánská pizza pouze tehdy, jestliže model obsahuje omezení, že PizzaMargherita žádné další přísady neobsahuje

OWA VS. CWA VE VZTAHU K ODVOZOVÁNÍ

- ✗ Databáze je příkladem uzavřeného světa:
 - + Hledáme osobu X v živnostenském rejstříku, která provozuje činnost Y
 - + Možný výsledek: „Osoba nenalezena.“
 - + V principu OWA bychom řekli, že daná osoba není živnostník
 - + To, co není v databázi, neexistuje
- ✗ Při použití klasifikátoru bychom jen na základě neobdržení dat nemohli konstatovat, že živnostník neexistuje
- ✗ Klasifikátor může rozhodnout, když má explicitně vyjádřené informace

PRINCIP UNA (UNIQUE NAME ASSUMPTION)

- ✗ Jedinec má jedinečné jméno (žádné přezdívký)
- ✗ Jazyk OWL, verze DL, princip UNA považuje za nepravdivý
- ✗ Odvození, že dva jedinci s různými jmény reprezentují tentýž objekt skutečnosti, není klasifikátorem označeno za nekonzistentní
- ✗ Dva různé názvy mohou označovat ten samý objekt

DĚKUJI ZA POZORNOST
