

# Tvorba obsahu (doménových) ontologií

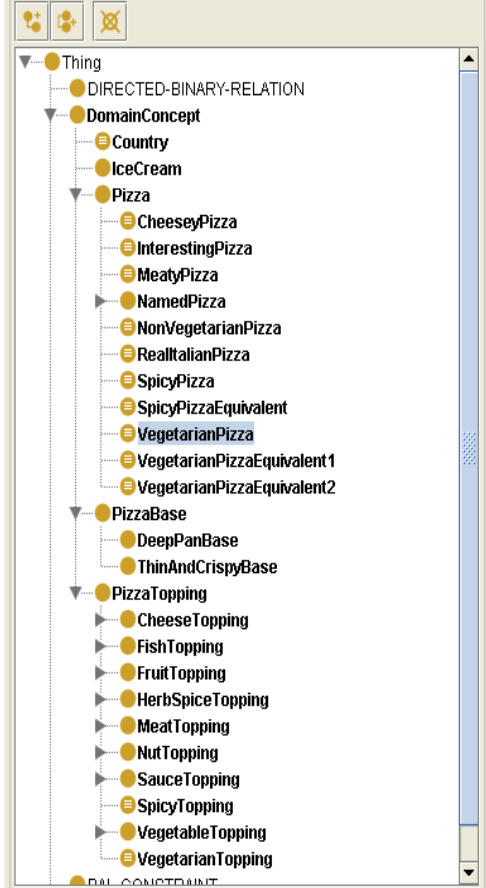
# Doménové ontologie

- Vyvíjené pro dílčí oblast (doménu)
  - Příklady viz text ve sborníku
- Dnes se často předpokládá (a realizuje) tvorba informatiky bez dlouhodobých zkušeností
- Běžně známé nástroje (editory, odvozovací systémy) zajistí jen **syntaktickou** a **formálně-logickou** správnost ontologií
- Hlavními prostředky pro podporu adekvátnosti **věcného obsahu** jsou
  - Strukturované metodiky
  - Základní ontologie a obsahové (návrhové) vzory
  - Metody učení ontologií z textů

# Editors ontologií

- Nejznámější **Protégé**
- Dále SWOOP, TopBraid, SemanticWorks, OntoEdit/OntoStudio, OKS (pro Topic Maps)...
- Většinou umožňují propojení s odvozovacími nástroji (viz dále)
- Zejména Protégé dále vybaven zásuvnými moduly pro veškeré další myslitelné úlohy ontologického inženýrství
  - Velmi rozsáhlá uživatelská komunita, pořádání vlastních konferencí atd.

Asserted Class Hierarchy: VegetarianPizza



Class Annotations: VegetarianPizza

Annotations +

- label: PizzaVegetariana@pt
- comment: 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

Class Annotations Class Usage

Class Description: VegetarianPizza

Equivalent classes +

- Pizza  
and not (hasTopping some FishTopping)  
and not (hasTopping some MeatTopping)

Superclasses +

Inherited anonymous classes

- hasBase some PizzaBase

Instances +

Disjoint classes +

- NonVegetarianPizza

Asserted class hierarchy | Inferred class hierarchy

### Object Properties: hasBase

- FROM
- hasCountryOfOrigin
- hasIngredient
  - hasBase
  - hasTopping
- hasSpiciness
- isIngredientOf
  - isBaseOf
  - isToppingOf
- TO

### Annotations: hasBase

Annotations +

Annotations Object Property Usage

### Characteristics: has

- Functional
- Inverse functional
- Transitive
- Symmetric
- Antisymmetric
- Reflexive
- Irreflexive

### Description: hasBase

Equivalent object properties +

Super properties +

- hasIngredient

Inverse properties +

- isBaseOf

Disjoint properties +

Property chains +

### Domains and ranges: hasBase

Domains (intersection) +

- Pizza

Ranges (intersection) +

- PizzaBase

# Strukturované metodiky

- Popisují optimální proces vývoje a životního cyklu (zejména doménové) ontologie
- Zárodky se objevují se již cca od r. 1990
- Zralé metodiky v polovině 90. let
  - Uschold & Grüninger (metodika vycházející ze zkušenosti s tvorbou „ontologií firmy“)
  - Gomez-Perez et al. (METHONTOLOGY)
- Od té doby se výrazně nezměnily, jen podrobnější pokrytí speciálních fází vývoje
  - kolaborativní tvorba ontologií, učení ontologií z textů, využití základních ontologií (např. OntoClean) a návrhových vzorů, mapování ontologií...

# Obvyklá posloupnost kroků

- Ujasnění účelu a rozsahu ontologie
- Specifikace terminologické části
- Odlišení ontologických typů
- Specifikace taxonomie
- Vytvoření netaxonomických relací, atributů a instancí
- Specifikace pokročilých axiomů
- Nasazení a údržba ontologie

# Ujasnění účelu a rozsahu

- Formulace obecných scénářů a případů užití
- Konkrétní vzorové **kompetenční otázky**
  - Co by mělo být možné s pomocí ontologie zodpovědět?
- Vyvarovat se bezbřehosti (viz „hugeness problem“)
- Současně ale zachovat otevřenost pro prozatím neřešené aplikace (znovupoužití)

# Specifikace terminologické části

- Východiskem vývoje ontologie bývá seznam relevantních **termínů**
  - Možnost již v této fázi využít podporu automatickým nástrojem
    - text mining / **učení ontologií**
- Slovní definice pojmů – **glosář** – usnadňují i komunikaci mezi více vývojáři navzájem

# Odlišení ontologických typů

- Třídy, instance, relace, atributy
- Zpravidla existuje více možností!
- Rozhodnutí, co jak modelovat se může řídit mj. **logickými vzory**

# Specifikace taxonomie

- **Taxonomie** zpravidla tvoří páteř ontologie
- Budovat se dá
  - Shora dolů
    - velmi obecné doménové ontologie, přímo navázané na základní ontologie
  - Zdola nahoru
    - např. ontologie orientované na používanou terminologii
  - Ze středu „ven“
    - od nejfrekventovanějších pojmů – často nejefektivnější způsob
- Nejobecnější pojmy z doménové ontologie je každopádně vhodné navázat na některou **základní ontologii** resp. **obsahový vzor**
- Vedle taxonomie nadřazenosti pojmů se často uplatní **partonomie** (celek-část), struktury **závislostí** apod.

# Netaxonomické relace, atributy, instance

- Důležité je jejich umístění na vhodnou hierarchickou úroveň
- **Relace** a **atributy** v ontologických jazycích někdy do značné míry splývají
  - Např. objektové a datové vlastnosti v OWL
- **Instance** se zpravidla zařazují jen tehdy, pokud mají v doméně zásadní postavení
  - Zejména jsou-li potřebné pro definování tříd

# Specifikace pokročilých axiomů

- Nedělá se vždy, jen pokud se nad ontologií předpokládá sofistikovanější odvozování
- Např. výrazy definující příslušnost instancí ke třídě, případně relaci; ekvivalence tříd; disjunktnost tříd; matematické vlastnosti relací...
- Zpravidla nejprve v **predikátovém** jazyce, teprve následně se řeší např. „ořezání“ do deskripční logiky

# Nasazení a údržba

- Ontologie zpravidla nejprve vzniká „na papíře“ (nejprve v podobě textových glosářů, později tabulek), teprve po ustálení struktury se **kóduje** ve zvoleném jazyce
- Před vlastním nasazením je potřeba provést **evaluaci**
  - v případě ontologií je objektivní formální evaluace velmi obtížná, spíše vstupují do hry subjektivní faktory
- Dlouhodobě se doména může vyvíjet, bývá potřeba např. **verzování**

# Učení ontologií

- Textové materiály jsou téměř vždy významným vstupem; vedle využití jako vodítka pro lidského návrháře na ně lze aplikovat i automatické techniky
- Pokročilé metody učení ontologií do jisté míry replikují kroky ruční tvorby ontologií
  - nalezení terminologie; rozlišení tříd a instancí; konstrukce taxonomie; nalezení a pojmenování netaxonomických relací; učení axiomů
- Základem jsou techniky zejména
  - z oblasti IR (TFIDF apod.)
  - z oblasti NLP („Hearst patterns“ apod.)
- Lze použít od začátku, nebo i pro doplnění již existující ontologie

# OntoClean

- Specializovaná metodika, jejímž cílem je omezit výskyt nejběžnějších modelovacích chyb, a zajistit konzistenci výsledné ontologie
- Vyvinutá N. Guarinem a C. Weltym
- Vychází z obecně teoretických výzkumů v oblasti filosofické ontologie
- Viz článek
  - <http://www.loa-cnr.it/Papers/GuarinoWeltyOntoCleanv3.pdf>

# OntoClean

- Metavlastnosti
  - anotují jednotlivé součásti navrhované ontologie, zejména třídy (zde označované jako „vlastnosti“ – de facto unární predikáty)
- Pravidla (integritní omezení) nad metavlastnostmi
  - umožňují vykázat existující nekonzistence v dané ontologii
  - umožňují omezit další vývoj ontologie pouze na konzistentní rozšíření

# Metavlastnosti v OntoClean

- Esencialita
  - Vlastnost je pro *danou entitu* esenciální, pokud ji daná entita musí mít, jinak by přestala být sama sebou (např. „být člověkem“ nebo „mít mozek“, pro konkrétního člověka); neboli musí pro ni platit v každém „možném světě“
  - Vlastnost, kterou entita může, ale nemusí mít, je tzv. *akcidentální*
- Rigidnost
  - Vlastnost je *rigidní*, pokud je esenciální pro všechny entity, které ji mají (např. „být člověkem“)
  - Vlastnost je *semi-rigidní*, pokud je pro některé entity esenciální, a pro některé ne (např. „být červený“)
  - Vlastnost je *anti-rigidní*, pokud není esenciální pro žádné entity (např. „být studentem“, „být potravou“)

# Metavlastnosti v OntoClean

- Identita
  - Vlastnost může
    - být nositelem vlastního *kritéria identity* („být člověkem“)
    - zdědit ho („být mužem“)
    - vůbec ho nemít („být červenou věcí“)
  - Vztah identity a esenciality
    - Identita je potřebná i k určení esenciality: kdy je ještě entita sama sebou a kdy už ne?
    - Naopak když víme, že se objekty liší v esenciální vlastnosti, nemohou být identické (vlastnost „tvar“ pro sochu nebo dům vs. hromadu hlíny/cihel)

# Metavlastnosti v OntoClean

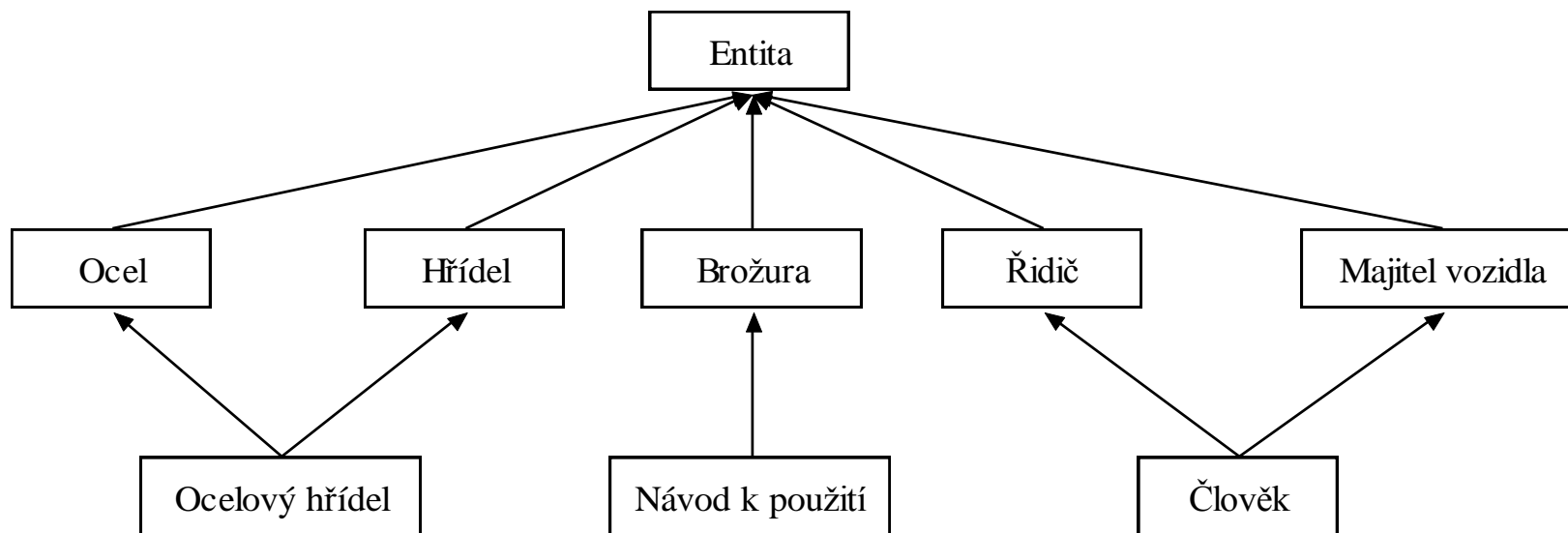
- Jednota
  - v jakém vztahu jsou části určité entity k celku
    - Topologicky (kus uhlí)
    - Morfologicky (suhvězdí)
    - Funkčně (dvoudílné plavky)
  - *Kritérium jednoty* je relace, která spojuje části do jednoho celku
    - Vlastnost má kritérium jednoty, pokud všechny entity, které ji mají, jsou „ohraničené celky“
    - Příklad: vlastnost „být množstvím vody“ nemá kritérium jednoty, „být oceánem“ ho má

# Odvozování nadmetavlastnostmi

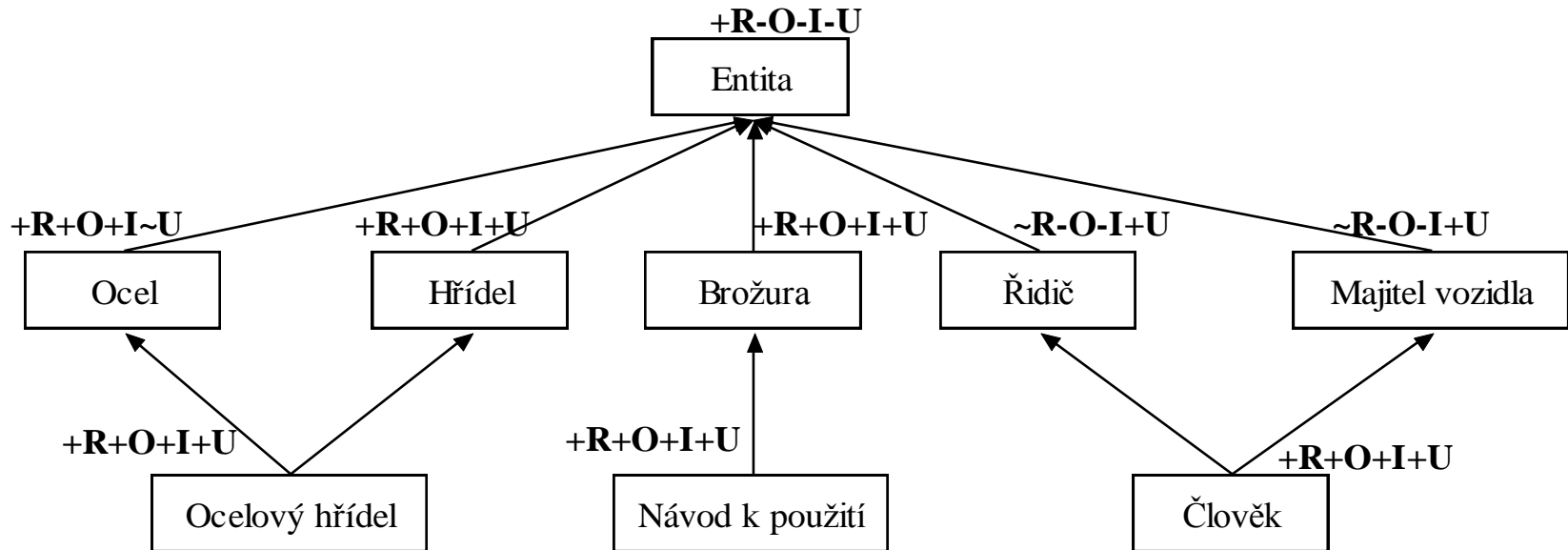
- Páteřní taxonomie ontologie
  - zahrnuje všechny rigidní vlastnosti
  - ostatní vlastnosti mají typicky charakter „rolí“
- Rigidní vlastnost nesmí být podřazena anti-rigidní
  - třída „Student“ nemůže mít podtřídy „Muž“ a „Žena“
- Podřazená vlastnost musí mít stejná kritérium identity a jednoty jako nadřazená
  - Třída „Množství vody“ nemůže mít podtřídu „Oceán“

# Příklad „čištění“ ontologie

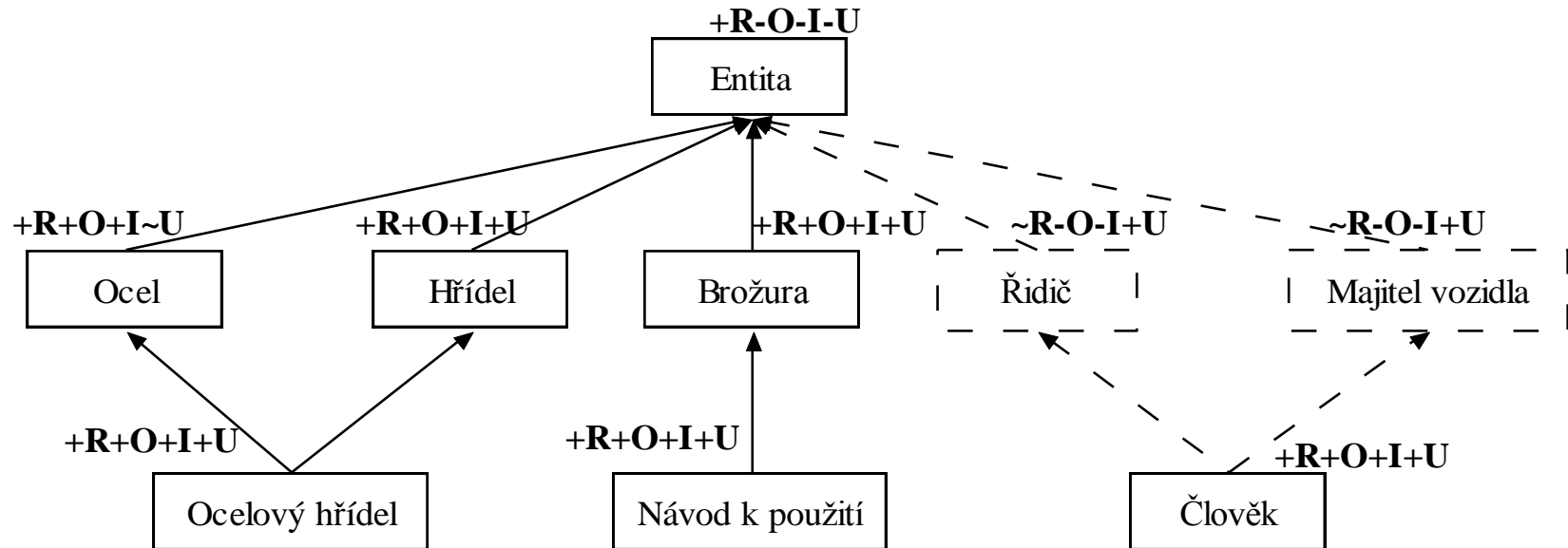
(Z. Zdráhal, 2010)



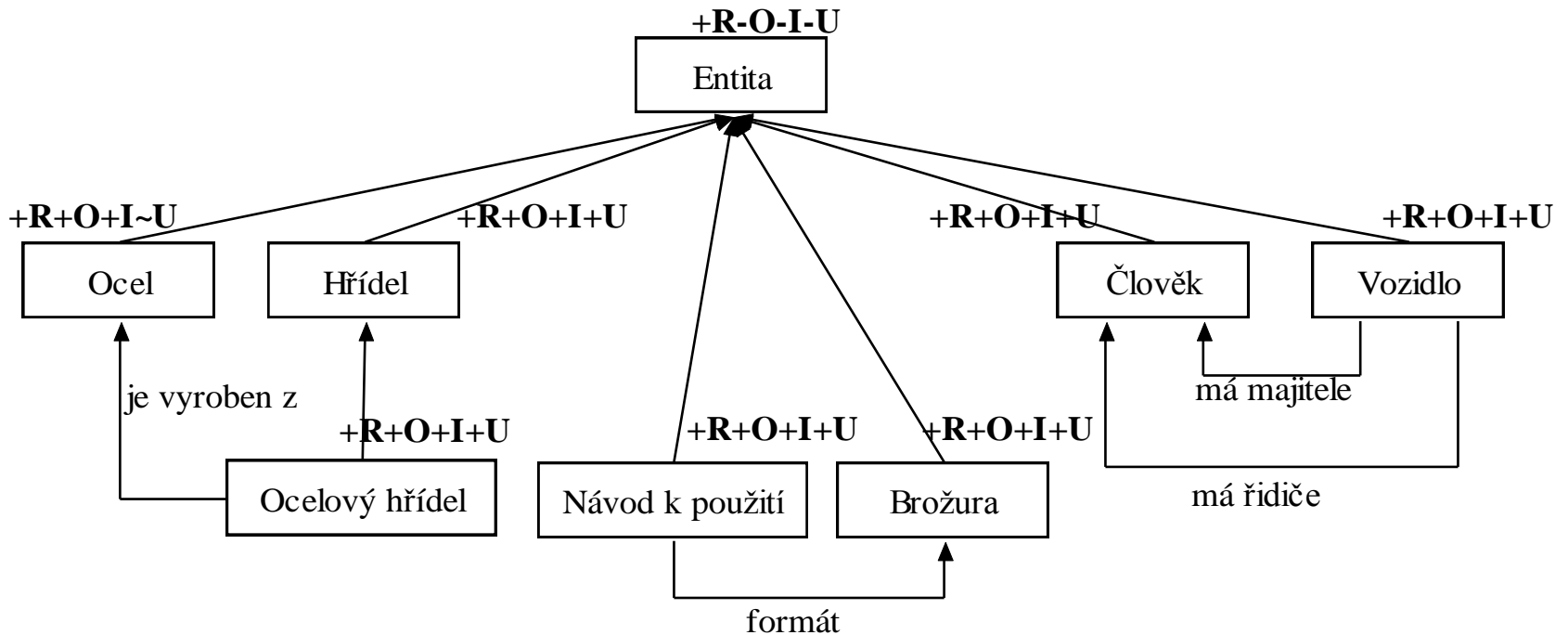
# Přiřazení metavlastností



# Vyznačení páteřní taxonomie



# Opravená taxonomie



# Implementace OntoClean

- Plugin pro ontologický editor Protégé
  - Ruční přiřazování metavlastností
- Spojení s extrakcí informací z WWW
  - Automatická podpora přiřazování metavlastností
  - např. když se v dokumentech často objevuje
    - výraz „trochu X“, pak X nemá kritérium jednoty
    - výraz „Y už není X“, pak X není rigidní vlastnost



Classes

Slots

Forms

Instances

Queries

PAL Constraints

Relationship

Sup...



Group of people (type=OntoClean property)

:THING A

:SYSTEM-CLASS A

Entity

Amount of matter

Red

Agent

Group

Group of people

Social entity

Organization

Location

Name

Documentation

Constraints

Group of people

Role

Concrete

Carrying Identity

+I:carries\_IC

Supplying Identity

-O:does\_not\_supply\_IC

Unity

~U:anti-unity

Rigidity

+R:rigid

Dependency

-D:not\_dependent

Social entity (type=OntoClean property)

Name

Documentation

Constraints



nality

Oth

Social entity

Role

Concrete

Carrying Identity

-I:does\_not\_carry\_IC

Supplying Identity

Unity

+U:carries\_UC

Rigidity

+R:rigid

Dependency

-D:not\_dependent