SemanticWeb - Ontology Engineering and Reasoning in the Semantic Web

GEIST Research Group
http://geist.agh.edu.pl

AGH University of Science and Technology, POLAND

Using slides according to license from:

- P. Hitzler – "Knowledge Representation for the Semantic Web" *course based on*
- P. Hitzler, M. Krötzsch, S. Rudolph – Foundations of Semantic Web Technologies
- e-Lite: 01LHVIU - Semantic Web: Technologies, Tools, Applications

Outline

1 Ontology Engineering

2 Ontology Reasoning

3 The End

Outline

Initial Questions

WHY/WHAT FOR do you need an ontology?

- do you need it at all?
- various types, domains
- various levels of expressive power
- various use case scenarios

HOW to get one?

- manually develop your own
- reuse existing ones
- acquire with machine learning techniques

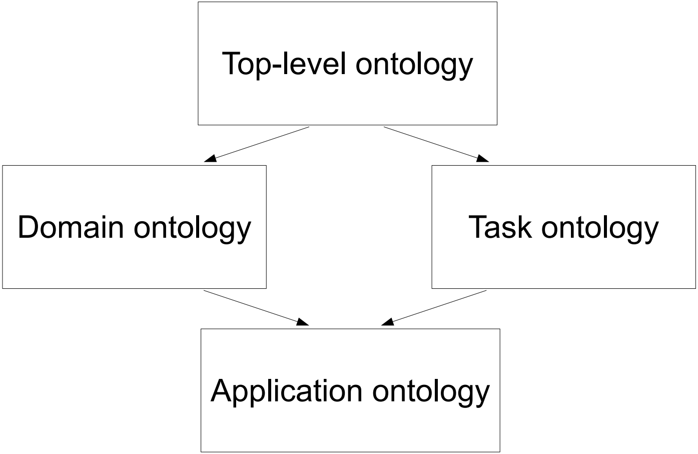
Requirement Analysis

Life cycle – an idea adopted from SE

Ontology Engineering, as SE, deals with challenges of designing complex systems (ontologies) by providing methodologies and auxiliary tools for their development, evaluation and maintenance.

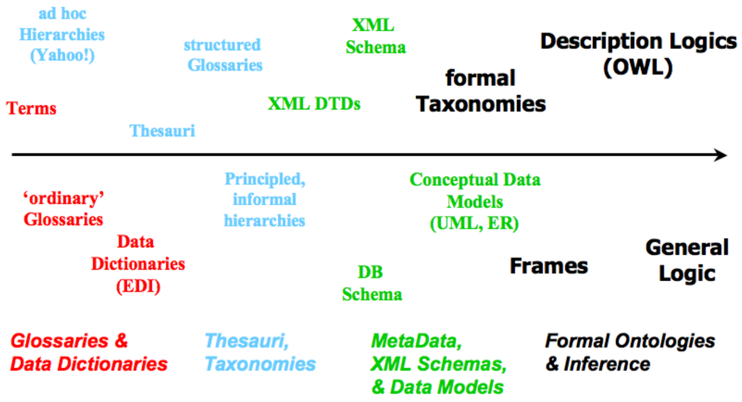
- A semantic description vs. classical solutions, e.g. database
  - knowledge exchange and integration
  - semantic specification + deduction → implicit knowledge
- Do we think logically?
- Available tools: license, support, maturity, interoperability
- What domain has to be modeled? What aspects of it?
- What are the tasks to be accomplished?
  - browsing, searching, querying, inferences – what kind of?
- If a semantic, logic-based formalism is used, then *which one*?

Types of ontologies

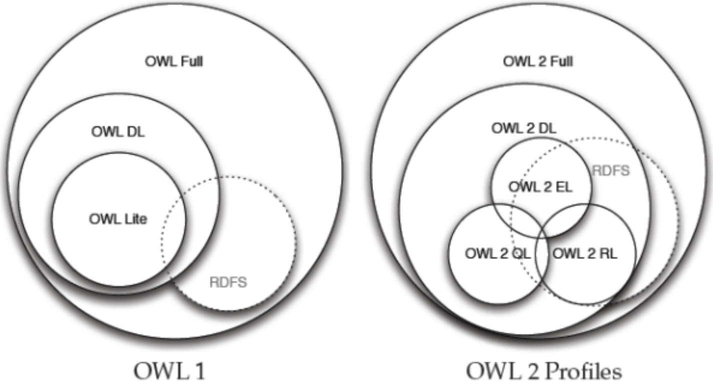


One of the ontology classification by N. Guarino (1998)

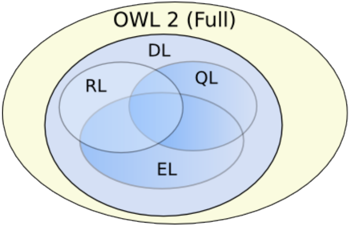
Formality levels of ontologies



OWL 1 vs. OWL 2


OWL Full
OWL DL
OWL Lite
RDFS
OWL 1
OWL 2 Full
OWL 2 DL
RDFS
OWL 2 EL
OWL 2 QL
OWL 2 RL
OWL 2 Profiles

OWL 2 languages



OWL 2 Profiles (EL, QL, RL)

- Reasoning in **polynomial time** w.r.t. the size of data
  → See http://www.w3.org/TR/owl2-profiles/
- substantial user communities
- expressiveness limitations (e.g. no negation, disjunction)

OWL 2 Profiles Revisited

OWL 2 EL

- large, class-expression oriented ontologies
- multiple classes/properties, complex structural descriptions
- heavy use of classification to manage and apply the terminology
- biohealth ontologies, e.g.: Galen , SNOMED-CT, the NCI thesaurus,
- other domains/applications: system configurations, product inventories, etc.

Examples of what CAN'T be stated

- "all children of a rich person are rich"
- "parentOf" and "childOf" are inverses of each other

OWL 2 Profiles Revisited

OWL 2 QL

- ontologies multiple instances, querying as the most important task
- can represent key features of ER and UML diagrams
- querying can be implemented with use of standard RDBS preprocessing layer
- for representing database schemas or integrating them via query rewriting
- supports inverse properties and subproperty hierarchies, restricts class axioms asymmetrically

Examples of what CAN'T be stated

- "every person has a *female* parent" (exist. quant. of roles)

OWL 2 Profiles Revisited

OWL 2 RL

- scalable reasoning, but still expressive
- reasoning can be implemented with rule-based engines
- ideal for enriching RDF data with rules
- limited class modelling possibilities
- restriction for class expressions (hard to express unknown individuals)
- restricts class axioms asymmetrically

Examples of what CAN'T be stated

- "every person has a parent" (the existence of an individual enforces the existence of another individual)

Modeling ontologies

- Heavyweight ontologies using Frames and First Order Logic
- Heavyweight ontologies using Description Logics
- Modeling ontologies with SE techniques
  - UML + OCL for lightweight ontologies
    - easy, popular, CASE tools
    - UML class diagrams for concepts, OCL for class axioms, object diagrams for instances
  - Entity/Relationship diagrams + integrity constraints

Outline

Where is your knowledge?

Creating and ontology – transferring knowledge into computer-accessible form → from what kinds of sources?

- human sources
- unstructured sources
- semi-structured sources
- structured sources

It's in Your Heads: Human Sources

- A person knowledgeable about the domain – a domain expert
- A person knowledgeable in the representation formalism – a knowledge engineer
- Interviews → cast knowledge into logical specification
  - human communication, misunderstandings, redundancy, feedback
  - knowledge acquisition with examples or machine learning techniques (decision tree learning, ILP)
  - knowledge authoring interface

It's in Your Books: Unstructured Sources

- Hard problem: extracting formal specification from arbitrary written texts
- AI $\rightarrow$ computational linguistics
  - grammatical structure analysis (parsing)
  - formalization: grammatical interdependencies $\rightarrow$ logical interdependencies with use of transformation rules
  - relevant lexical background knowledge needed (thesauri, e.g *WordNet*) to complement the *commonsense* knowledge

It's on the Web: Semi-structured Sources

- hyperlinks, wiki articles, semantic wikis
- file systems

Semi-structured Sources (cont.)

Author: betty_crocker

Publisher: wiley_publishing

Date: 2006

Language: english

Genre: handbook

Pages: 352

Keywords: christmas, cooking

Recommendation:

Books by this author:

betty_crocker_christmas_cookbook

Books in this genre:

betty_crocker_christmas_cookbook

logical_foundations_for_rule_based_systems

the_christmas_table

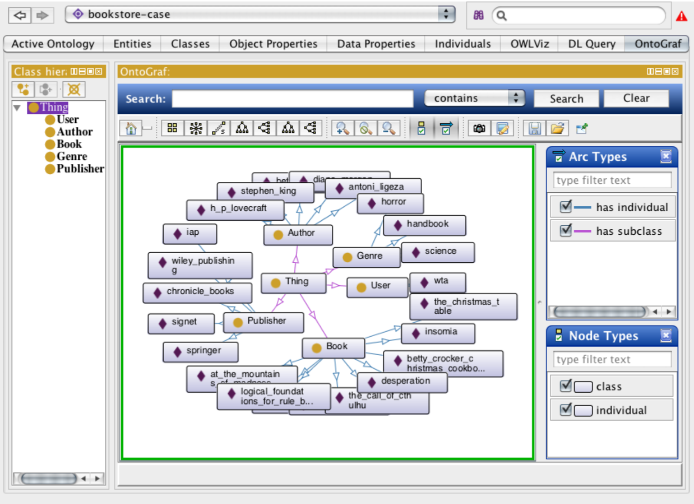
Books by this publisher:

betty_crocker_christmas_cookbook

Export to RDF/XML

Export to RDF

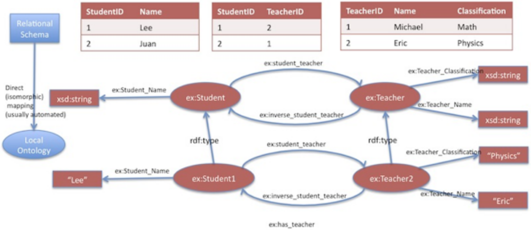
Semi-structured Sources (cont.)



Semantic Web Ontologies2
GEIST
Outline
Ontology Engineering
Introduction
Ontology Creation – Where Is Your Knowledge?
Constructing Ontologies Manually
Reusing Existing Ontologies
Semiautomatic Ontology Acquisition
Ontology Mapping
Ontology Design Patterns
Ontology Reasoning
Reasoning in OWL
DL revisited
Reasoning problems
The End

It's in the Databases: Structured Sources

- content of the RDB translated into ontology
  - ontology population (single individuals)
  - schema information → terminological axioms
- reusing existing ontologies:
  - ontology merging (several ontologies integrated into one)
  - ontology alignment (loose coupling) with ontology mapping



Outline

1. **Ontology Engineering**
   - Introduction
   - Ontology Creation – Where Is Your Knowledge?
   - Constructing Ontologies Manually
   - Reusing Existing Ontologies
   - Semiautomatic Ontology Acquisition
   - Ontology Mapping
   - Ontology Design Patterns
2. **Ontology Reasoning**
   - Reasoning in OWL
   - DL revisited
   - Reasoning problems
3. **The End**

Main Stages in Ontology Development

1. Determine scope
2. Consider reuse
3. Enumerate terms
4. Define taxonomy
5. Define properties
6. Define facets
7. Define instances
8. Check for anomalies

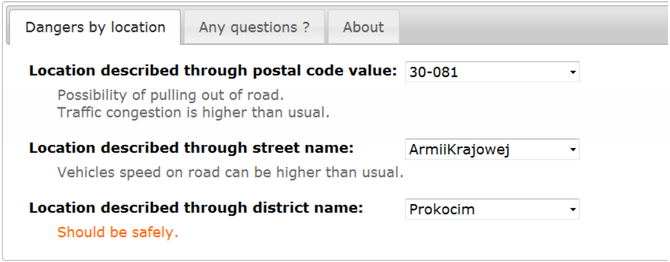
Not a linear process!

Determine Scope

- There is no correct ontology of a specific domain
  - An ontology is an abstraction of a particular domain, and there are always viable alternatives
- What is included in this abstraction should be determined by
  - the use to which the ontology will be put
  - by future extensions that are already anticipated

Example: TrafficDanger System

- **Main objective**: inform users in real-time about threats in urban environment
- **Ontology-driven system** based on TrafficDangerOntology
- **Ontology** describing traffic conditions and threats…
- …to be used with a **relational database** (concrete data)



Determine Scope (2)

- Basic questions to be answered at this stage are:
  - What is the domain that the ontology will cover?
  - For what we are going to use the ontology?
  - For what types of questions should the ontology provide answers?
  - Who will use and maintain the ontology?

Example: TrafficDangerOntology

- What are the traffic dangers on specific area ?
- Are there any dangers connected with low friction on specific area ?
- What are the subareas of specific location ?
- What kind of dangers are connected with bad atmospheric conditions ?
- Is there any danger connected with specific postal code on specific district ?
- Are there any traffic conditions provided for specific location ?

Consider Reuse

- With the spreading deployment of the Semantic Web, ontologies will become more widely available
- We rarely have to start from scratch when defining an ontology
  - There is almost always an ontology available from a third party that provides at least a useful starting point for our own ontology

Enumerate Terms

- Write down in an unstructured list all the relevant terms that are expected to appear in the ontology
  - Nouns form the basis for class names
  - Verbs (or verb phrases) form the basis for property names
- Traditional knowledge engineering tools (e.g. laddering and grid analysis) can be used to obtain
  - the set of terms
  - an initial structure for these terms

Example: TrafficDangerOntology

- **Classes**: Location, TrafficCondition, TrafficDanger, etc.
- **Object Properties**: hasLocation, hasTrafficCondition, etc.
- **Datatype Properties**: hasAverageDriversDensityPerKilometer

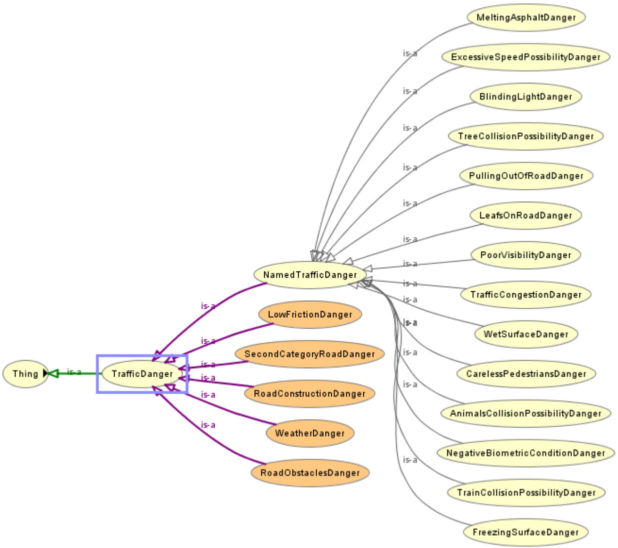
Define Taxonomy

- Relevant terms must be organized in a taxonomic hierarchy
  - Opinions differ on whether it is more efficient/reliable to do this in a top-down or a bottom-up fashion
- Ensure that hierarchy is indeed a taxonomy:
  - If A is a subclass of B, then every instance of A must also be an instance of B (compatible with semantics of **rdfs:subClassOf**

Example: TrafficDangerOntology

- Thing
  - Location
    - DistrictLocation
    - PostalCodeLocation
    - StreetLocation
  - TrafficCondition
    - AreaCondition
      - BuiltUpCondition
      - UnbuiltCondition
    - CongestionCondition
      - HighCongestionCondition
      - LowCongestionCondition
      - NormalCongestionCondition
    - RoadConstructionCondition
      - BuildingMaterialCondition
        - AsphaltCondition
        - ConcreteCondition
        - GravelCondition
      - CrosswalkCondition
        - AboveGroundCrosswalkCondition
        - GroundCrosswalkCondition
          - SignedCrosswalkCondition
          - UnsignedCrosswalkCondition
        - UndergroundCrosswalkCondition
      - RailwayCrossingCondition
        - SignedCrossingCondition
        - UnsignedCrossingCondition
      - RoadExtensionCondition
        - NoRoadsideCondition
        - NoSafetyBarriersCondition
        - RoadsideCondition
        - SafetyBarriersCondition
        - TreesAlongRoadCondition
      - SurfaceCondition
        - FirstCategorySurfaceCondition
          - SmoothCondition
        - SecondCategorySurfaceCondition
          - FullOfHolesCondition
          - FullOfWheelTracksCondition
          - InsufficientlySlopedCondition
          - PoorlyDrainedCondition
      - TrafficLighsCondition
        - BrokenLightsCondition
        - NoLightsCondition
        - WorkingLightsCondition
    - RoadInspectionCondition
      - NoInspectionCondition
      - PhotoRadarCondition
      - PoliceControlCondition
    - TimeCondition
      - DayCondition
      - DuskCondition
      - NightCondition
    - WeatherCondition
      - AirPressureCondition
        - HighPressureCondition
        - LowPressureCondition
      - AirTemperatureCondition
        - BelowFreezingTemperatureCondition
        - HighTemperatureCondition
        - LowTemperatureCondition
        - MediumTemperatureCondition
      - CloudinessCondition
        - CloudyCondition
        - SunnyCondition
      - PrecipitationCondition
        - FoggyCondition
        - NoPrecipitationCondition
        - RainyCondition
        - SnowyCondition
      - WindCondition
        - CrosswindCondition
        - HeadwindCondition
        - TailwindCondition
  - TrafficDanger
    - LowFrictionDanger
    - NamedTrafficDanger
      - AnimalsCollisionPossibilityDanger
      - BlindingLightDanger
      - CarelessPedestriansDanger
      - ExcessiveSpeedPossibilityDanger
      - FreezingSurfaceDanger
      - LeafsOnRoadDanger
      - MeltingAsphaltDanger
      - NegativeBiometricConditionDanger
      - PoorVisibilityDanger
      - PullingOutOfRoadDanger
      - TrafficCongestionDanger
      - TrainCollisionPossibilityDanger
      - TreeCollisionPossibilityDanger
      - WetSurfaceDanger
    - RoadConstructionDanger
    - RoadObstaclesDanger
    - SecondCategoryRoadDanger
    - WeatherDanger

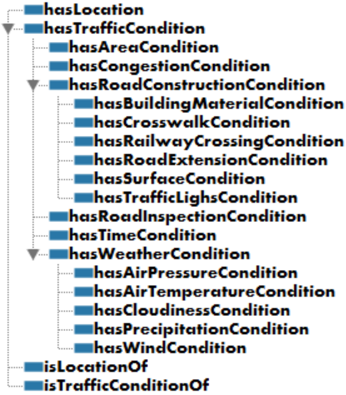
Example: TrafficDangerOntology



Define Properties

- Often interleaved with the previous step
- The semantics of **subClassOf** demands that whenever A is a subclass of B, every property statement that holds for instances of B must also apply to instances of A
  - It makes sense to attach properties to the highest class in the hierarchy to which they apply

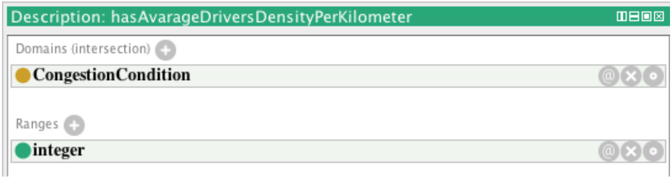
Example: TrafficDangerOntology



Define Properties (2)

- While attaching properties to classes, it makes sense to immediately provide statements about the domain and range of these properties
- There is a methodological tension here between generality and specificity:
  - Flexibility (inheritance to subclasses)
  - Detection of inconsistencies and misconceptions

Example: TrafficDangerOntology



Define Facets: From RDFS to OWL

- Cardinality restrictions
- Required values
  - **owl:hasValue**
  - **owl:allValuesFrom**
  - **owl:someValuesFrom**
- Relational characteristics
  - symmetry, transitivity, inverse properties, functional values

Example: TrafficDangerOntology

- WeatherDanger concept:

Equivalent classes

```
TrafficDanger
 and hasWeatherCondition some WeatherCondition
```

- LowFrictionDanger concept

Equivalent classes

```
TrafficDanger
 and (hasPrecipitationCondition some (RainyCondition
                                      or SnowyCondition)
   and hasSurfaceCondition some PoorlyDrainedCondition)
  or (hasAirTemperatureCondition some HighTemperatureCondition
     and hasBuildingMaterialCondition some AsphaltCondition)
  or (hasAirTemperatureCondition some BelowFreezingTemperatureCondition)
```

- DangersAtStareMiasto

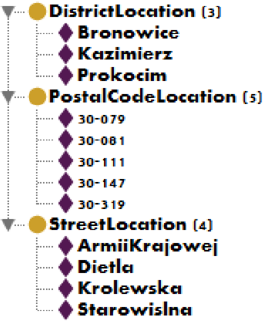
```
TrafficDanger and hasTrafficCondition some (CongestionCondition and hasLocation value StareMiasto)
```

Define Instances

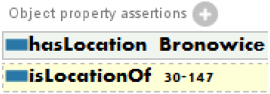
- Filling the ontologies with such instances is a separate step
- Number of instances >> number of classes
- Thus populating an ontology with instances is not done manually
  - Retrieved from legacy data sources (DBs)
  - Extracted automatically from a text corpus

Example: TrafficDangerOntology

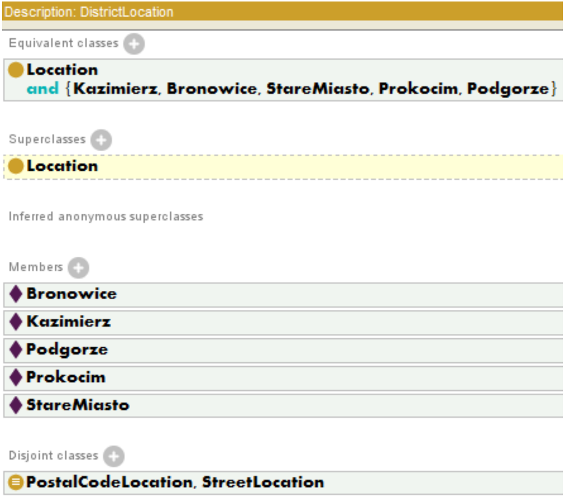
- Individuals by classes:



- Property assertions for ArmiiKrajowej inividual:



Example: TrafficDangerOntology



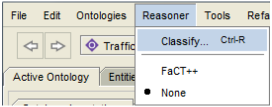
Check for Anomalies

- An important advantage of the use of OWL over RDF Schema is the possibility to detect inconsistencies
  - In ontology or ontology+instances
- Examples of common inconsistencies
  - incompatible domain and range definitions for transitive, symmetric, or inverse properties
  - cardinality properties
  - requirements on property values can conflict with domain and range restrictions

15

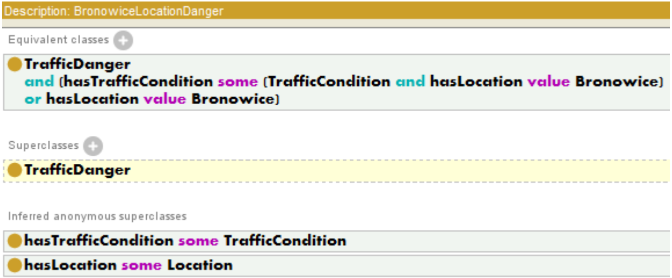
Chapter 7 A Semantic Web Primer

Example: TrafficDangerOntology

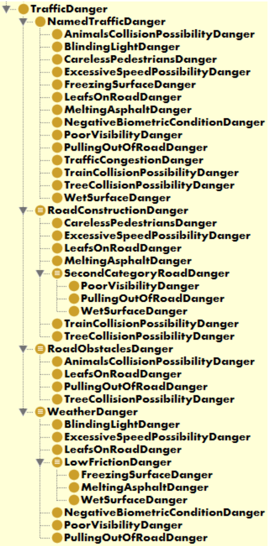


Inferred information about Bronowice:

Inferred information about BronowiceLocationDanger:



Example: TrafficDangerOntology



Outline

Existing Domain-Specific Ontologies

- **Medical domain**: Cancer ontology from the National Cancer Institute in the United States
- **Cultural domain**:
  - Art and Architecture Thesaurus (AAT) with 125,000 terms in the cultural domain
  - Union List of Artist Names (ULAN), with 220,000 entries on artists
  - Iconclass vocabulary of 28,000 terms for describing cultural images
- **Geographical domain**: Getty Thesaurus of Geographic Names (TGN), containing over 1 million entries

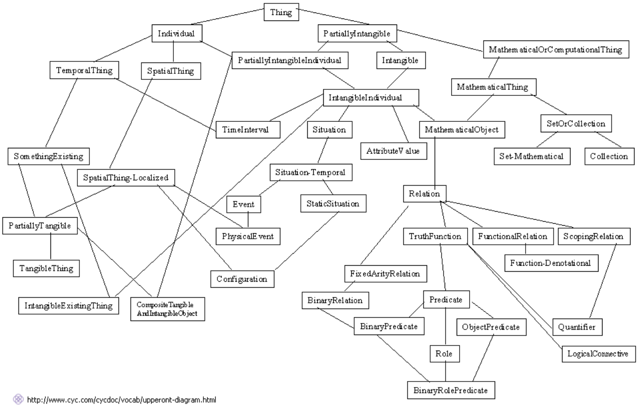
Upper-Level Ontologies

- Some attempts have been made to define very generally applicable ontologies
  - Mot domain-specific
- Cyc, with 60,000 assertions on 6,000 concepts
- Standard Upperlevel Ontology (SUO)

Example: CyC ontology

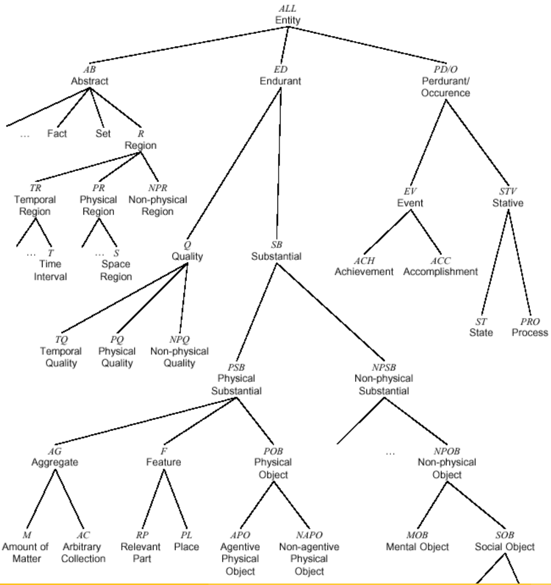
OpenCyc Selected Vocabulary and Upper Ontology

E-Mail Comments to: doc@cyc.com
Last update: March 27, 2002

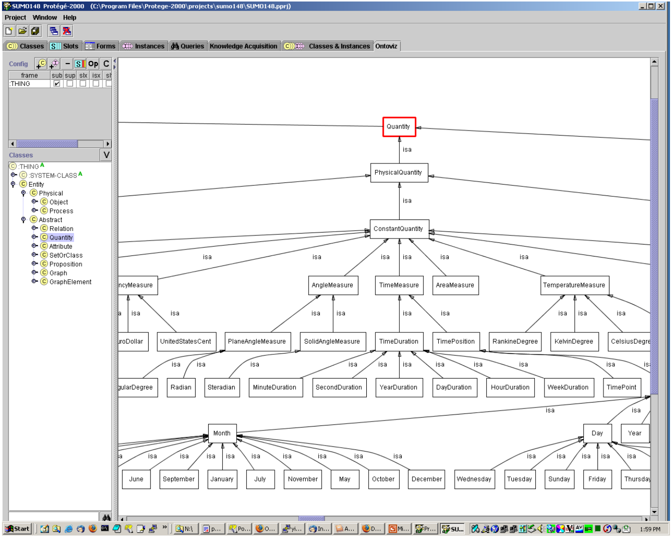


http://www.cyc.com/cycdoc/vocab/upperont-diagram.html

Example: DOLCE ontology



Example: SUMO ontology



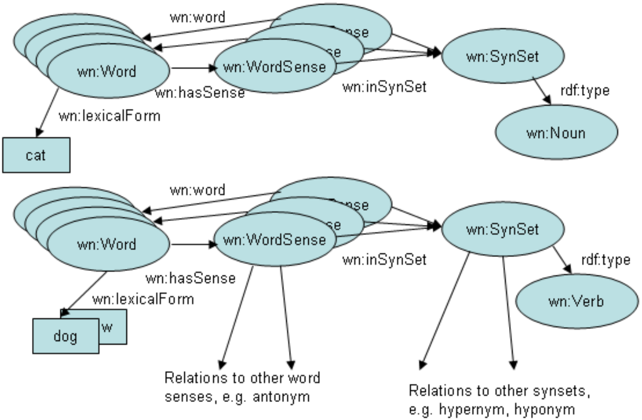
Topic Hierarchies

- Some "ontologies" do not deserve this name:
  - simply sets of terms, loosely organized in a hierarchy
- This hierarchy is typically not a strict taxonomy but rather mixes different specialization relations (e.g. is-a, part-of, contained-in)
- Such resources often very useful as starting point
- Example: Open Directory hierarchy, containing more then 400,000 hierarchically organized categories and available in RDF format

Linguistic Resources

- Some resources were originally built not as abstractions of a particular domain, but rather as linguistic resources
- These have been shown to be useful as starting places for ontology development
  - E.g. WordNet, with over 90,000 word senses

Example: WordNet



Outline

1. **Ontology Engineering**
   - Introduction
   - Ontology Creation – Where Is Your Knowledge?
   - Constructing Ontologies Manually
   - Reusing Existing Ontologies
   - Semiautomatic Ontology Acquisition
   - Ontology Mapping
   - Ontology Design Patterns
2. **Ontology Reasoning**
   - Reasoning in OWL
   - DL revisited
   - Reasoning problems
3. **The End**

The Knowledge Acquisition Bottleneck

- Manual ontology acquisition remains a time-consuming, expensive, highly skilled, and sometimes cumbersome task
- Machine Learning techniques may be used to alleviate
  - knowledge acquisition or extraction
  - knowledge revision or maintenance

Tasks Supported by Machine Learning

- Extraction of ontologies from existing data on the Web
- Extraction of relational data and metadata from existing data on the Web
- Merging and mapping ontologies by analyzing extensions of concepts
- Maintaining ontologies by analyzing instance data
- Improving SW applications by observing users

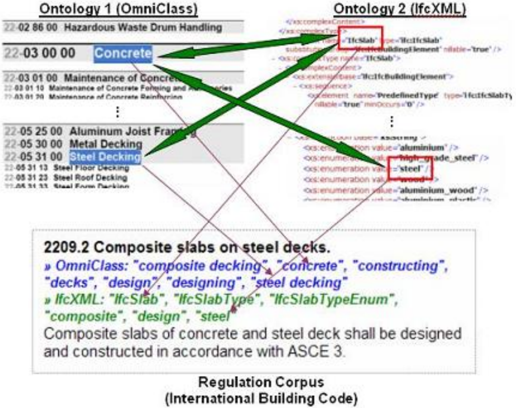
Outline

Ontology Mapping

- A single ontology will rarely fulfill the needs of a particular application; multiple ontologies will have to be combined
- This raises the problem of ontology integration (also called ontology alignment or ontology mapping)
- Current approaches deploy a whole host of different methods; we distinguish linguistic, statistical, structural and logical methods

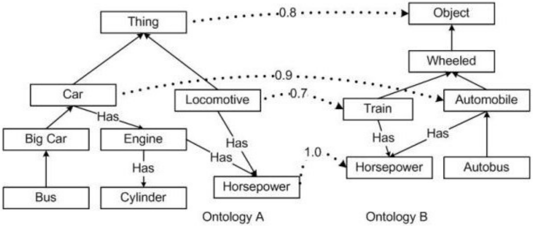
Ontology Mapping Methods

- Linguistic Methods
  - exploit the linguistic labels attached to the concepts → discover potential matches
  - specialized domain knowledge required, similarity is relative



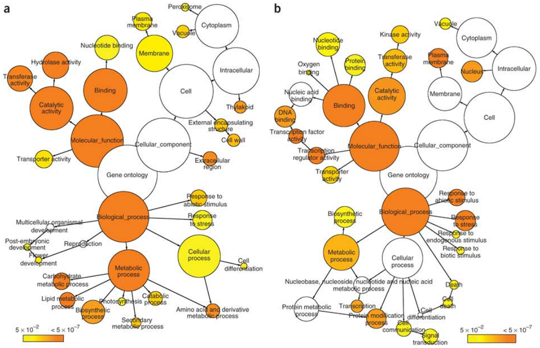
Ontology Mapping Methods

- Statistical Methods
  - analyze statistical correlation between the instances of source and target concepts
  - sufficiently large data corpus required



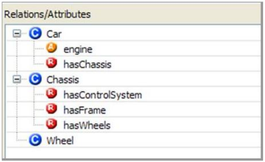
Ontology Mapping Methods

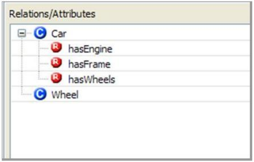
- Structural Methods
  - exploit the internal graph structure of the source and target ontologies
  - can be combined with linguistic methods



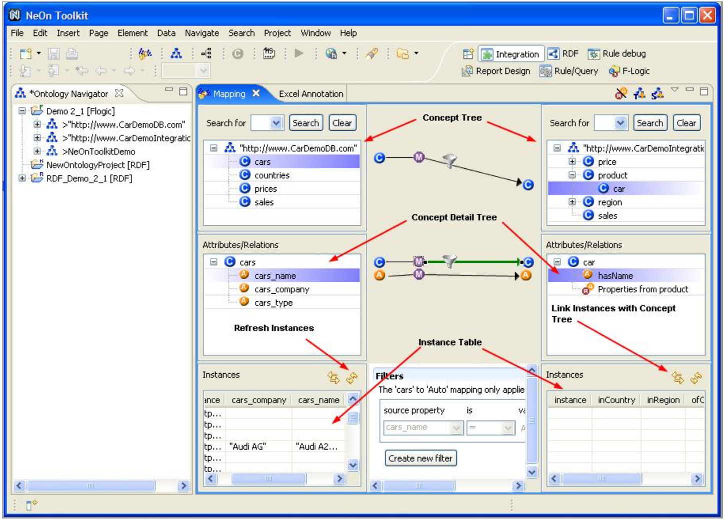
Ontology Mapping Methods

- Logical Methods
  - most specific, but most of the ontologies are semantically lightweight (do not carry much logical formalism)





Example: Neon toolkit



Ontology-Mapping Techniques Conclusion

- Although there is much potential, and indeed need, for these techniques to be deployed for Semantic Web engineering, this is far from a well-understood area
- No off-the-shelf techniques are currently available, and it is not clear that this is likely to change in the near future

Outline

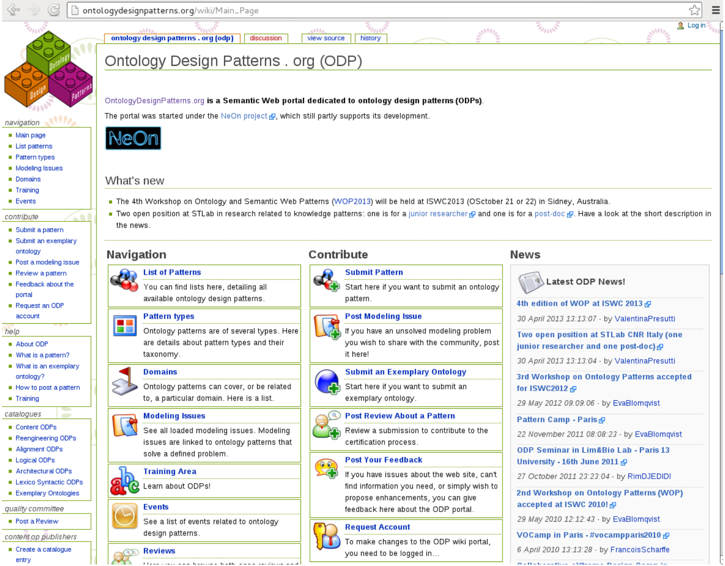
Rationale

- It is hard to reuse only the "useful pieces" of a comprehensive (foundational) ontology, and the cost of reuse may be higher than developing a new ontology from scratch
- Need for small (or cleverly modularized) ontologies with explicit documentation of design rationales, and best reengineering practices
- Hence, in analogy to software design patterns: ontology design patterns
- ODPs summarize the good practices to be applied within design solutions
- ODPs keep track of the design rationales that have motivated their adoption
- See `http://ontologydesignpatterns.org/wiki/Main_Page`

Types of Patterns

- Six families of ODPs: Structural OPs, Correspondence OPs, Content OPs (CPs), Reasoning OPs, Presentation OPs, and Lexico-Syntactic OPs
- CPs can be distinguished in terms of the domain they represent
- Correspondence OPs (for reengineering and mappings)
- Reasoning OPs are typical reasoning procedures
- Presentation OPs relate to ontology usability from a user perspective; e.g., we distinguish between Naming OPs and Annotation OPs
- Lexico-Syntactic OP are linguistic structures or schemas that permit to generalize and extract some conclusions about the meaning they express

Ontology Design Patterns Website



Outline

1 Ontology Engineering

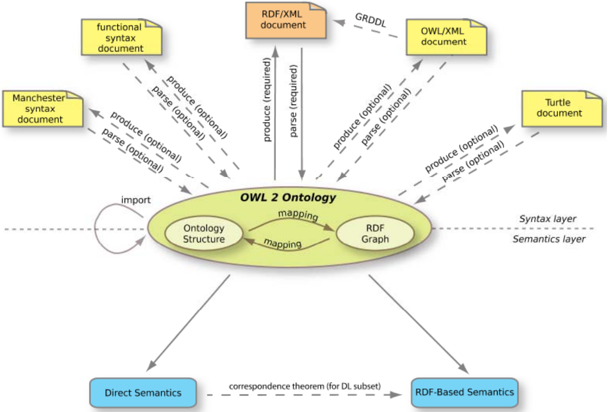
2 Ontology Reasoning
- Reasoning in OWL
- DL revisited
- Reasoning problems

3 The End

Web Ontology Language (OWL)

- What can be done with OWL?
  - Consistency checks – are there contradictions in the logical model?
  - Satisfiability checks – are there classes that cannot have any instances?
  - Classification – what is the type of a particular instance?

OWL2 structure



F. Corno, L. Farinetti - Politecnico di Torino 14

OWL2 semantics

- The Direct Semantics and the RDF-Based Semantics provide two alternative ways of assigning meaning to OWL 2 ontologies
  - A correspondence theorem provides a link between the two
- These two semantics are used by reasoners and other tools to answer class consistency, subsumption, instance retrieval queries, …

OWL 2 direct semantics

- Assigns meaning directly to ontology structures, resulting in a semantics compatible with the model theoretic semantics of the SROIQ description logic
  - SROIQ is a fragment of first order logic with useful computational properties
- The advantage of this close connection is that the extensive description logic literature and implementation experience can be directly exploited by OWL 2 tools
- However, some conditions must be placed on ontology structures in order to ensure that they can be translated into a SROIQ knowledge base
  - E.g, transitive properties cannot be used in number restrictions
- Ontologies that satisfy these syntactic conditions are called OWL 2 DL ontologies

OWL 2 RDF-based semantics

- Assigns meaning directly to RDF graphs and so indirectly to ontology structures via the Mapping to RDF graphs
- The RDF-Based Semantics is fully compatible with the RDF Semantics, and extends the semantic conditions defined for RDF
- The RDF-Based Semantics can be applied to any OWL 2 Ontology, without restrictions, as any OWL 2 Ontology can be mapped to RDF
- "OWL 2 Full" is used informally to refer to RDF graphs considered as OWL 2 ontologies and interpreted using the RDF-Based Semantics

Outline

Description Logics: Syntax

- *Concepts* corresponds to classes
- *Roles* correspond to class properties
- *Constructors* mix of set notation and FO quantification

  Booleans: $C \sqcap D$, $C \sqcup D$, $\neg C$
  Qualified quantification: $\forall R.C$, $\exists R.C$

- Variable free notation for concepts (classes)
  - $artist(x) = person(x) \wedge \exists y\ created(x, y) \wedge Artwork(y)$
    is written as $Artist \sqsubseteq Person \sqcap \exists created.Artwork$

OWL-DL Semantics

- The semantics of OWL-DL constructs is derived by the corresponding Description Logic operators.
- The formal definition is extremely concise
  - http://www.w3.org/TR/owl-semantics/semantics-all.html#3
  - ...but not so straightforward to understand!

Description Logic: Semantics

- Interpretations are pairs $(\Delta, \cdot^{\mathcal{I}})$, with a universe Δ and a mapping $\mathcal{I}$ from
  - concept names to subsets of Δ
  - role names to binary relations
- Booleans: $C \sqcap D$, $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
  $C \sqcup D$ $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
  $\neg C$ $(\neg C)^{\mathcal{I}} = \Delta \setminus C^{\mathcal{I}}$

  Qualified quantification:

  $\forall R.C$ $\forall R.C^{\mathcal{I}} = \{x \in \Delta \mid \forall y \in \Delta : R^{\mathcal{I}}(x,y) \rightarrow y \in C^{\mathcal{I}})\}$

  $\exists R.C$ $\exists R.C^{\mathcal{I}} = \{x \in \Delta \mid \exists y \in \Delta : R^{\mathcal{I}}(x,y) \& y \in C^{\mathcal{I}})\}$

Modular Definition of Description Logics

Constructor	Syntax	Semantics
concept name	C	$C^{\mathcal{I}}$
conjunction	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
univ. quant.	$\forall R.C$	$\{d_1 \mid \forall d_2 \in \Delta^{\mathcal{I}}.(R^{\mathcal{I}} d_1 d_2 \rightarrow d_2 \in C^{\mathcal{I}})\}$
top	$\top$	$\Delta^{\mathcal{I}}$
negation ($\mathcal{C}$)	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
disjunction ($\mathcal{U}$)	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
exist. quant. ($\mathcal{E}$)	$\exists R.C$	$\{d_1 \mid \exists d_2 \in \Delta^{\mathcal{I}}.(R^{\mathcal{I}} d_1 d_2 \wedge d_2 \in C^{\mathcal{I}})\}$
number restr. ($\mathcal{N}$)	$\geq nR$	$\{d_1 \mid \|\{d_2 \mid R^{\mathcal{I}} d_1 d_2\}\| \geq n\}$
	$\leq nR$	$\{d_1 \mid \|\{d_2 \mid R^{\mathcal{I}} d_1 d_2\}\| \leq n\}$
one-of ($\mathcal{O}$)	$\{a_1, \ldots, a_n\}$	$\{d \mid d = a_i^{\mathcal{I}}$ for some $a_i\}$
filler ($\mathcal{B}$)	$\exists R.\{a\}$	$\{d \mid d = R^{\mathcal{I}} d a^{\mathcal{I}}\}$
role name	R	$R^{\mathcal{I}}$
role conj. ($\mathcal{R}$)	$R_1 \sqcap R_2$	$R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}}$
inverse roles ($\mathcal{I}$)	R^{-1}	$\{(d_1, d_2) \mid R^{\mathcal{I}}(d_2, d_1)\}$

OWL Formal semantics

- http://www.w3.org/TR/owl-semantics/semantics-all.html#3
- Ingredients:
  - Vocabularies Vxx
  - Mapping functions (which provide ‘meaning’) EC, ER
  - Interpretation of syntax constructs
  - Interpretation of axioms and facts

Interpretation of constructs

Abstract Syntax	Interpretation (value of EC)
complementOf(c)	O - EC(c)
unionOf(c_1 … c_n)	EC(c_1) ∪ … ∪ EC(c_n)
intersectionOf(c_1 … c_n)	EC(c_1) ∩ … ∩ EC(c_n)
oneOf(i_1 … i_n), for i_j individual IDs	{S(i_1), …, S(i_n)}
oneOf(v_1 … v_n), for v_j literals	{S(v_1), …, S(v_n)}
restriction(p x_1 … x_n), for n > 1	EC(restriction(p x_1)) ∩…∩EC(restriction(p x_n))
restriction(p allValuesFrom(r))	{x ∈ O | <x,y> ∈ ER(p) implies y ∈ EC(r)}
restriction(p someValuesFrom(e))	{x ∈ O | ∃ <x,y> ∈ ER(p) ∧ y ∈ EC(e)}
restriction(p value(i)), for i an individual ID	{x ∈ O | <x,S(i)> ∈ ER(p)}
restriction(p value(v)), for v a literal	{x ∈ O | <x,S(v)> ∈ ER(p)}
restriction(p minCardinality(n))	{x ∈ O | card({y ∈ O∪LV : <x,y> ∈ ER(p)}) ≥ n}
restriction(p maxCardinality(n))	{x ∈ O | card({y ∈ O∪LV : <x,y> ∈ ER(p)}) ≤ n}
restriction(p cardinality(n))	{x ∈ O | card({y ∈ O∪LV : <x,y> ∈ ER(p)}) = n}
Individual(annotation(p_1 o_1) … annotation(p_k o_k) type(c_1) … type(c_m) pv_1 … pv_n)	EC(annotation(p_1 o_1)) ∩ … EC(annotation(p_k o_k)) ∩ EC(c_1) ∩ … ∩ EC(c_m) ∩ EC(pv_1) ∩…∩ EC(pv_n)
Individual(i annotation(p_1 o_1) … annotation(p_k o_k) type(c_1) … type(c_m) pv_1 … pv_n)	{S(i)} ∩ EC(annotation(p_1 o_1)) ∩ … EC(annotation(p_k o_k)) ∩ EC(c_1) ∩ … ∩ EC(c_m) ∩ EC(pv_1) ∩…∩ EC(pv_n)
value(p Individual(…))	{x ∈ O | ∃ y∈EC(Individual(…)) : <x,y> ∈ ER(p)}
value(p id) for id an individual ID	{x ∈ O | <x,S(id)> ∈ ER(p) }
value(p v) for v a literal	{x ∈ O | <x,S(v)> ∈ ER(p) }
annotation(p o) for o a URI reference	{x ∈ R | <x,S(o)> ∈ ER(p) }
annotation(p Individual(…))	{x ∈ R | ∃ y ∈ EC(Individual(…)) : <x,y> ∈ ER(p) }

Interpretation of axioms and facts I

Directive	Conditions on interpretations
Class(c complete annotation(p_1 o_1) … annotation(p_k o_k) $descr_1$ … $descr_n$)	S(c) ∈ EC(annotation(p_1 o_1)) … S(c) ∈ EC(annotation(p_k o_k)) EC(c) = EC($descr_1$) ∩…∩ EC($descr_n$)
Class(c partial annotation(p_1 o_1) … annotation(p_k o_k) $descr_1$ … $descr_n$)	S(c) ∈ EC(annotation(p_1 o_1)) … S(c) ∈ EC(annotation(p_k o_k)) EC(c) ⊆ EC($descr_1$) ∩…∩ EC($descr_n$)
EnumeratedClass(c annotation(p_1 o_1) … annotation(p_k o_k) i_1 … i_n)	S(c) ∈ EC(annotation(p_1 o_1)) … S(c) ∈ EC(annotation(p_k o_k)) EC(c) = { S(i_1), …, S(i_n) }
Datatype(c annotation(p_1 o_1) … annotation(p_k o_k))	S(c) ∈ EC(annotation(p_1 o_1)) … S(c) ∈ EC(annotation(p_k o_k)) EC(c) ⊆ LV
DisjointClasses(d_1 … d_n)	EC(d_i) ∩ EC(d_j) = { } for 1 ≤ i < j ≤ n
EquivalentClasses(d_1 … d_n)	EC(d_i) = EC(d_j) for 1 ≤ i < j ≤ n
SubClassOf(d_1 d_2)	EC(d_1) ⊆ EC(d_2)
DatatypeProperty(p annotation(p_1 o_1) … annotation(p_k o_k) super(s_1) … super(s_n) domain(d_1) … domain(d_n) range(r_1) … range(r_n) [Functional])	S(p) ∈ EC(annotation(p_1 o_1)) … S(p) ∈ EC(annotation(p_k o_k)) ER(p) ⊆ O×LV ∩ ER(s_1) ∩…∩ ER(s_n) ∩ EC(d_1)×LV ∩…∩ EC(d_n)×LV ∩ O×EC(r_1) ∩…∩ O×EC(r_n) [ER(p) is functional]
…	…

Interpretation of axioms and facts II

Directive	Conditions on interpretations
ObjectProperty(p annotation(p_1 o_1) … annotation(p_k o_k) super(s_1) … super(s_n) domain(d_1) … domain(d_n) range(r_1) … range(r_n) [inverse(i)] [Symmetric] [Functional] [InverseFunctional] [Transitive])	S(p) ∈ EC(annotation(p_1 o_1)) … S(p) ∈ EC(annotation(p_k o_k)) ER(p) ⊆ O×O ∩ ER(s_1) ∩…∩ ER(s_n) ∩ EC(d_1)×O ∩…∩ EC(d_n)×O ∩ O×EC(r_1) ∩…∩ O×EC(r_n) [ER(p) is the inverse of ER(i)] [ER(p) is symmetric] [ER(p) is functional] [ER(p) is inverse functional] [ER(p) is transitive]
AnnotationProperty(p annotation(p_1 o_1) … annotation(p_k o_k))	S(p) ∈ EC(annotation(p_1 o_1)) … S(p) ∈ EC(annotation(p_k o_k))
OntologyProperty(p annotation(p_1 o_1) … annotation(p_k o_k))	S(p) ∈ EC(annotation(p_1 o_1)) … S(p) ∈ EC(annotation(p_k o_k))
EquivalentProperties(p_1 … p_n)	ER(p_i) = ER(p_j) for $1 \le i < j \le n$
SubPropertyOf(p_1 p_2)	ER(p_1) ⊆ ER(p_2)
SameIndividual(i_1 … i_n)	S(i_j) = S(i_k) for $1 \le j < k \le n$
DifferentIndividuals(i_1 … i_n)	S(i_j) ≠ S(i_k) for $1 \le j < k \le n$
Individual([i] annotation(p_1 o_1) … annotation(p_k o_k) type(c_1) … type(c_m) pv_1 … pv_n)	EC(Individual([i] annotation(p_1 o_1) … annotation(p_k o_k) type(c_1) … type(c_m) pv_1 … pv_n)) is nonempty

Reasoning

- With the definition of the semantics, we may now define some reasoning metods
  - Reasoning on the structure of the ontology
  - Reasoning on relationships among classes
  - Reasoning on instances

ESSLLI 2005 | Stefan Schlobach & Heiner Stuckenschmidt | 20

Concept Reasoning

Based on these semantics there are two basic reasoning services:

- *Concept satisfiability*, $\models C \neq \bot$.
  - Check whether for some interpretation $\mathcal{I}$ we have $C^{\mathcal{I}} \neq \emptyset$.
  - $\models \forall creates.Sculpture \sqcap \exists creates.(Artwork \sqcap \neg Sculpture) = \bot$.
- *Concept subsumption*, $\models C_1 \sqsubseteq C_2$.
  - Check whether for all interpretations $\mathcal{I}$ we have $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
  - $\forall creates.Painting \sqcap \exists creates.\top \sqsubseteq \exists creates.Painting$.

Terminological Reasoning

$$\mathcal{T} = \{ Painting \sqsubseteq Artwork \sqcap \neg Sculpture, \\ Painter \sqsubseteq \exists creates.Paintings, \\ Sculpturer \sqsubseteq \exists creates.Artwork \sqcap \forall creates.Sculpture \}$$

- *Concept satisfiability*, $\Sigma \models C \neq \bot$.
  - Check whether there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \Sigma$ and $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
  - *Concept unsatisfiability:* $\Sigma \models Painter \sqcap Sculpturer = \bot$.
- *Subsumption*, $\Sigma \models C_1 \sqsubseteq C_2$.
  - Check whether for all interpretations $\mathcal{I}$ such that $\mathcal{I} \models \Sigma$ we have $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
  - *Subsumption:* $\Sigma \models Painter \sqsubseteq \neg Sculpurer$

Assertional reasoning

$\mathcal{A} = \{ rembrandt{:}Artist,\ nightwatch{:}Painting,\ (rembrandt,nightwatch){:}created\}$ and $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$

- *Consistency*, $\Sigma \not\models \bot$.
  - Check whether there exists $\mathcal{I}$ such that $\mathcal{I} \models \Sigma$.
  - $\Sigma \models \mathcal{A} \neq \bot$ but $\Sigma \models \mathcal{A} \cup \{rembrandt{:}Sculpturor\} = \bot$
- *Instance Checking*, $\Sigma \models a : C$.
  - Check whether $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all interpretations $\mathcal{I} \models \Sigma$.
  - $rembrandt \in_{\Sigma} Painter$.
- Defined reasoning tasks:
  - *Retrieval:* $retrieve(Artists) = \{rembrandt\}$.
  - *Realization:* find most specific concepts in $\mathcal{T}$ for instances in $\mathcal{A}$ $realize(rembrandt) = Painter$

Outline

A Reasoning Problem

A is a logical consequence of K
  written K ⊨ A
if and only if
  every model of K is a model of A.

- To show an entailment, we need to check all models?
- But that‘s infinitely many!!!

A Reasoning Problem

Kno.e.sis

We need algorithms which do not apply the model-based definition of logical consequence in a naive manner.

These algorithms should be syntax-based.
(Computers can only do syntax manipulations.)

Luckily, such algorithms exist!

However, their correctness (soundness and completeness) needs to be proven formally.
Which is often a non-trivial problem requiring substantial mathematical build-up.

We won‘t do the proofs here.

Important Inference Problems

Kno.e.sis

- Global consistency of a knowledge base. KB $\models$ **false**?
  - Is the knowledge base meaningful?
- Class consistency $C \equiv \bot$?
  - Is C necessarily empty?
- Class inclusion (Subsumption) $C \sqsubseteq D$?
  - Structuring knowledge bases
- Class equivalence $C \equiv D$?
  - Are two classes in fact the same class?
- Class disjointness $C \sqcap D = \bot$?
  - Do they have common members?
- Class membership C(a)?
  - Is a contained in C?
- Instance Retrieval „find all x with C(x)“
  - Find all (known!) individuals belonging to a given class.

Reduction to Unsatisfiability

- Global consistency of a knowledge base. — KB unsatisfiable
  - Failure to find a model.
- Class consistency — $C \equiv \bot$?
  - KB ∪ {C(a)} unsatisfiable
- Class inclusion (Subsumption) — $C \sqsubseteq D$?
  - KB ∪ {C ⊓ ¬D(a)} unsatisfiable (a new)
- Class equivalence — $C \equiv D$?
  - $C \sqsubseteq D$ und $D \sqsubseteq C$
- Class disjointness — $C \sqcap D = \bot$?
  - KB ∪ {(C ⊓ D)(a)} unsatisfiable (a new)
- Class membership — C(a)?
  - KB ∪ {¬C(a)} unsatisfiable
- Instance Retrieval „find all x with C(x)“
  - Check class membership for all individuals.

If you want to know more...

Reading:

- N. Guarino: "Formal Ontology and Information Systems"
- http://www.co-ode.org/resources/papers/ekaw2004.pdf
- http://www.betaversion.org/~stefano/linotype/news/57/ - SemWeb for the XML people
- http://www.cs.man.ac.uk/~horrocks/ISWC2003/Tutorial/ - OWL Tutorial from 2003
- http://ontologydesignpatterns.org/wiki/Main_Page - Ontology Design patterns

Tutorials:

- http://www.co-ode.org/resources/tutorials/
- http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/

Additional resources:

- http://www.co-ode.org/ontologies/

If you want to know more...

http://ai.ia.agh.edu.pl/wiki/pl:dydaktyka:semantic_web:start

OWL pictures used according to license:

- Squeezyboy: http://www.flickr.com/photos/squeezyboy/154171735/
- One Speckled Frog / Eric Hoffmann: http://www.flickr.com/photos/speckledfrog
- Larry McQueen - OWLS 1: http://www.natureartists.com/artists/artist_artwork.asp?ArtistID=1374&ArtworkID=18081
- Tasshu Rikimara: http://www.flickr.com/photos/tasshu113/4956888274/
- Johan J.Ingles-Le Nobel: http://www.flickr.com/photos/jingleslenobel/4971486823/
- Mara 1: http://www.flickr.com/photos/flametree/4517245491/
- Gabor Kovacs: http://www.flickr.com/photos/21923568@N00/2520990696/
- Rhys's Piece Is: http://www.flickr.com/photos/rhysasplundh/5105241656/

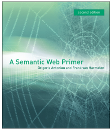
Questions

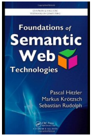
Any questions?

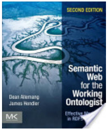
Thank you

Thank you for your attention!

http://geist.agh.edu.pl
GEIST Research Group







Powered by LaTeX