Semantic Web Technologies

Lecture 3: Top-down ontology development

Maria Keet

email: keet -AT- inf.unibz.it

home: http://www.meteck.org

blog:

http://keet.wordpress.com/category/computer-science/72010-semwebtech/

KRDB Research Center
Free University of Bozen-Bolzano, Italy

23 November 2009

Outline

So, we have OWL and OWL 2 as W3C standardised ontology languages—but what is an ontology, how dow you develop one, and what do you do with it?

Outline

Background

- Aristotle and colleagues: **O**ntology
- Engineering: ontolog**ies** (count noun)

- Investigating reality, representing it
- Putting an engineering artifact to use

What then, is this engineering artifact?

some slides based on

http://ontolog.cim3.net/file/resource/presentation/NicolaGuarino_20060202/DOLCE–NicolaGuarino_20060202.pdf

A few definitions

- Most quoted: "An ontology is a specification of a conceptualization" (by Tom Gruber, 1993)
- More detailed: "An ontology is a logical theory accounting for the *intended meaning* of a formal vocabulary, i.e. its *ontological commitment* to a particular *conceptualization* of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models." (Guarino, 1998)
- JWS03 paper: "with an ontology being equivalent to a Description Logic knowledge base" (Horrocks et al, 2003)

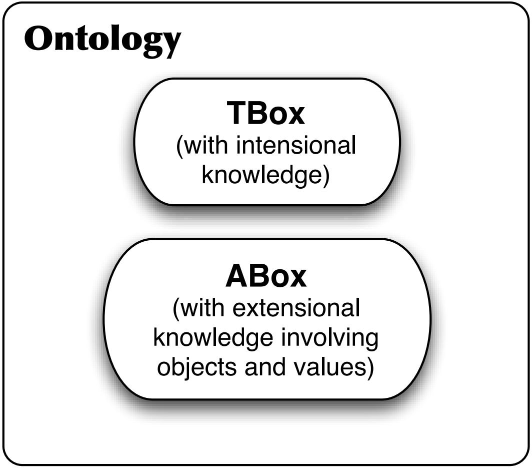
A few definitions

- Most quoted: "An ontology is a specification of a conceptualization" (by Tom Gruber, 1993)
- More detailed: "An ontology is a logical theory accounting for the *intended meaning* of a formal vocabulary, i.e. its *ontological commitment* to a particular *conceptualization* of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models." (Guarino, 1998)
- JWS03 paper: "with an ontology being equivalent to a Description Logic knowledge base" (Horrocks et al, 2003)

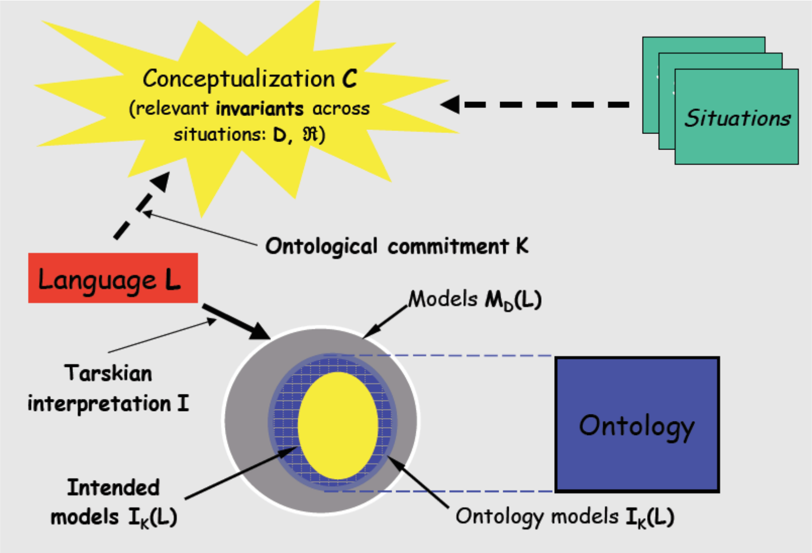
A few definitions

- Most quoted: "An ontology is a specification of a conceptualization" (by Tom Gruber, 1993)
- More detailed: "An ontology is a logical theory accounting for the *intended meaning* of a formal vocabulary, i.e. its *ontological commitment* to a particular *conceptualization* of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models." (Guarino, 1998)
- JWS03 paper: "with an ontology being equivalent to a Description Logic knowledge base" (Horrocks et al, 2003)

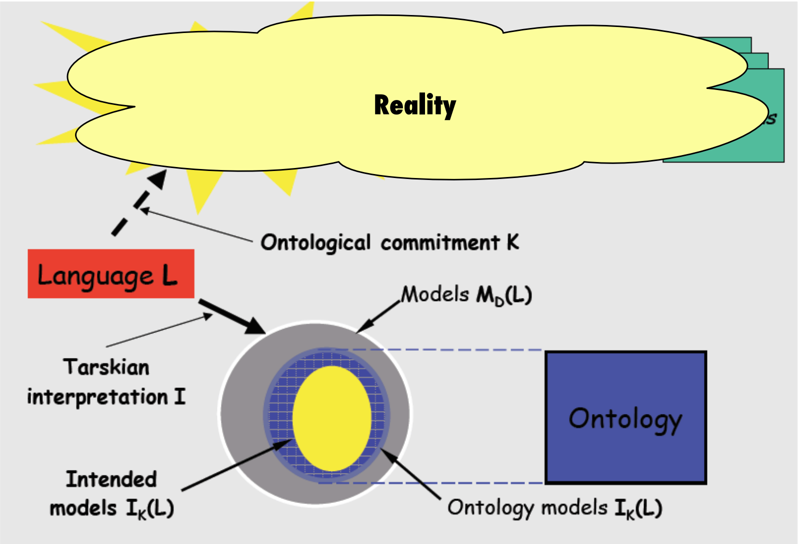
Description Logic knowledge base



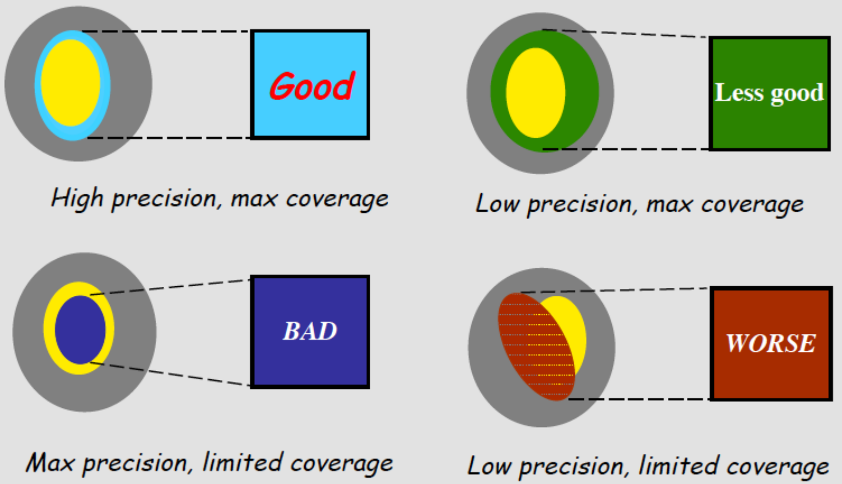
Ontologies and meaning



Ontologies and reality



Either way... Quality of the ontology



Either way... Quality of the ontology

- "Bad ontologies are (inter alia) those whose general terms lack the relation to corresponding universals in reality, and thereby also to corresponding instances."
- "Good ontologies are reality representations, and the fact that such representations are possible is shown by the fact that, as is documented in our scientific textbooks, very many of them have already been achieved, though of course always only at some specific level of granularity and to some specific degree of precision, detail and completeness"

(Smith, 2004)

Either way... Quality of the ontology

- "Bad ontologies are (inter alia) those whose general terms lack the relation to corresponding universals in reality, and thereby also to corresponding instances."
- "Good ontologies are reality representations, and the fact that such representations are possible is shown by the fact that, as is documented in our scientific textbooks, very many of them have already been achieved, though of course always only at some specific level of granularity and to some specific degree of precision, detail and completeness"

(Smith, 2004)

Initial Ontology Dimensions that have Evolved

- Semantic
  - Degree of Formality and Structure
  - Expressiveness of the Knowledge Representation Language
  - Representational Granularity
- Pragmatic
  - Intended Use
  - Role of Automated Reasoning
  - Descriptive vs. Prescriptive
  - Design Methodology
  - Governance

slide from, and more details available in:
http://ontolog.cim3.net/file/work/OntologySummit2007/symposium/OntologyFramework_symposium–Gruninger-Obrst_20070424.ppt

Initial Ontology Dimensions that have Evolved

- Semantic
  - Degree of Formality and Structure
  - Expressiveness of the Knowledge Representation Language
  - Representational Granularity
- Pragmatic
  - Intended Use
  - Role of Automated Reasoning
  - Descriptive vs. Prescriptive
  - Design Methodology
  - Governance

slide from, and more details available in:
http://ontolog.cim3.net/file/work/OntologySummit2007/symposium/
OntologyFramework_symposium–Gruninger-Obrst_20070424.ppt

Outline

General notion

- Provide a top-level with basic categories of kinds of things
- Principal choices
  - Endurantist vs. Perdurantist
  - Universals vs. Particulars
- Formal...
  - ... logic: connections between truths – neutral wrt truth
  - ... ontology: connections between things – neutral wrt reality

General notion

- Provide a top-level with basic categories of kinds of things
- Principal choices
  - Endurantist vs. Perdurantist
  - Universals vs. Particulars
- Formal...
  - ... logic: connections between truths – neutral wrt truth
  - ... ontology: connections between things – neutral wrt reality

General notion

- Provide a top-level with basic categories of kinds of things
- Principal choices
  - Endurantist vs. Perdurantist
  - Universals vs. Particulars
- Formal...
  - ... logic: connections between truths – neutral wrt **truth**
  - ... ontology: connections between things – neutral wrt **reality**

Outline

Descriptive Ontology for Linguistic and Cognitive Engineering

- Strong cognitive/linguistic bias:
  - Descriptive (as opposite to prescriptive) attitude
  - Categories mirror cognition, common sense, and the lexical structure of natural language.
- Emphasis on cognitive invariants
- Categories as conceptual containers: no 'deep' metaphysical implications
- Focus on design rationale to allow easy comparison with different ontological options
- Rigorous, systematic, interdisciplinary approach
- Rich axiomatization
  - 37 basic categories
  - 7 basic relations
  - 80 axioms, 100 definitions, 20 theorems
- Rigorous quality criteria
- Documentation

Descriptive Ontology for Linguistic and Cognitive Engineering

- Strong cognitive/linguistic bias:
  - Descriptive (as opposite to prescriptive) attitude
  - Categories mirror cognition, common sense, and the lexical structure of natural language.
- Emphasis on cognitive invariants
- Categories as conceptual containers: no ‘deep’ metaphysical implications
- Focus on design rationale to allow easy comparison with different ontological options
- Rigorous, systematic, interdisciplinary approach
- Rich axiomatization
  - 37 basic categories
  - 7 basic relations
  - 80 axioms, 100 definitions, 20 theorems
- Rigorous quality criteria
- Documentation

Descriptive Ontology for Linguistic and Cognitive Engineering

- Strong cognitive/linguistic bias:
  - Descriptive (as opposite to prescriptive) attitude
  - Categories mirror cognition, common sense, and the lexical structure of natural language.
- Emphasis on cognitive invariants
- Categories as conceptual containers: no 'deep' metaphysical implications
- Focus on design rationale to allow easy comparison with different ontological options
- Rigorous, systematic, interdisciplinary approach
- Rich axiomatization
  - 37 basic categories
  - 7 basic relations
  - 80 axioms, 100 definitions, 20 theorems
- Rigorous quality criteria
- Documentation

Descriptive Ontology for Linguistic and Cognitive Engineering

- Strong cognitive/linguistic bias:
  - Descriptive (as opposite to prescriptive) attitude
  - Categories mirror cognition, common sense, and the lexical structure of natural language.
- Emphasis on cognitive invariants
- Categories as conceptual containers: no 'deep' metaphysical implications
- Focus on design rationale to allow easy comparison with different ontological options
- Rigorous, systematic, interdisciplinary approach
- Rich axiomatization
  - 37 basic categories
  - 7 basic relations
  - 80 axioms, 100 definitions, 20 theorems
- Rigorous quality criteria
- Documentation

Descriptive Ontology for Linguistic and Cognitive Engineering

- Strong cognitive/linguistic bias:
  - Descriptive (as opposite to prescriptive) attitude
  - Categories mirror cognition, common sense, and the lexical structure of natural language.
- Emphasis on cognitive invariants
- Categories as conceptual containers: no 'deep' metaphysical implications
- Focus on design rationale to allow easy comparison with different ontological options
- Rigorous, systematic, interdisciplinary approach
- Rich axiomatization
  - 37 basic categories
  - 7 basic relations
  - 80 axioms, 100 definitions, 20 theorems
- Rigorous quality criteria
- Documentation

Descriptive Ontology for Linguistic and Cognitive Engineering

- Strong cognitive/linguistic bias:
  - Descriptive (as opposite to prescriptive) attitude
  - Categories mirror cognition, common sense, and the lexical structure of natural language.
- Emphasis on cognitive invariants
- Categories as conceptual containers: no ‘deep’ metaphysical implications
- Focus on design rationale to allow easy comparison with different ontological options
- Rigorous, systematic, interdisciplinary approach
- Rich axiomatization
  - 37 basic categories
  - 7 basic relations
  - 80 axioms, 100 definitions, 20 theorems
- Rigorous quality criteria
- Documentation

Descriptive Ontology for Linguistic and Cognitive Engineering

- Strong cognitive/linguistic bias:
  - Descriptive (as opposite to prescriptive) attitude
  - Categories mirror cognition, common sense, and the lexical structure of natural language.
- Emphasis on cognitive invariants
- Categories as conceptual containers: no ‘deep’ metaphysical implications
- Focus on design rationale to allow easy comparison with different ontological options
- Rigorous, systematic, interdisciplinary approach
- Rich axiomatization
  - 37 basic categories
  - 7 basic relations
  - 80 axioms, 100 definitions, 20 theorems
- Rigorous quality criteria
- Documentation

Descriptive Ontology for Linguistic and Cognitive Engineering

- Strong cognitive/linguistic bias:
  - Descriptive (as opposite to prescriptive) attitude
  - Categories mirror cognition, common sense, and the lexical structure of natural language.
- Emphasis on cognitive invariants
- Categories as conceptual containers: no 'deep' metaphysical implications
- Focus on design rationale to allow easy comparison with different ontological options
- Rigorous, systematic, interdisciplinary approach
- Rich axiomatization
  - 37 basic categories
  - 7 basic relations
  - 80 axioms, 100 definitions, 20 theorems
- Rigorous quality criteria
- Documentation

Descriptive Ontology for Linguistic and Cognitive Engineering

- Strong cognitive/linguistic bias:
  - Descriptive (as opposite to prescriptive) attitude
  - Categories mirror cognition, common sense, and the lexical structure of natural language.
- Emphasis on cognitive invariants
- Categories as conceptual containers: no ‘deep’ metaphysical implications
- Focus on design rationale to allow easy comparison with different ontological options
- Rigorous, systematic, interdisciplinary approach
- Rich axiomatization
  - 37 basic categories
  - 7 basic relations
  - 80 axioms, 100 definitions, 20 theorems
- Rigorous quality criteria
- Documentation

Descriptive Ontology for Linguistic and Cognitive Engineering

- Strong cognitive/linguistic bias:
  - Descriptive (as opposite to prescriptive) attitude
  - Categories mirror cognition, common sense, and the lexical structure of natural language.
- Emphasis on cognitive invariants
- Categories as conceptual containers: no 'deep' metaphysical implications
- Focus on design rationale to allow easy comparison with different ontological options
- Rigorous, systematic, interdisciplinary approach
- Rich axiomatization
  - 37 basic categories
  - 7 basic relations
  - 80 axioms, 100 definitions, 20 theorems
- Rigorous quality criteria
- Documentation

Descriptive Ontology for Linguistic and Cognitive Engineering

- Strong cognitive/linguistic bias:
  - Descriptive (as opposite to prescriptive) attitude
  - Categories mirror cognition, common sense, and the lexical structure of natural language.
- Emphasis on cognitive invariants
- Categories as conceptual containers: no 'deep' metaphysical implications
- Focus on design rationale to allow easy comparison with different ontological options
- Rigorous, systematic, interdisciplinary approach
- Rich axiomatization
  - 37 basic categories
  - 7 basic relations
  - 80 axioms, 100 definitions, 20 theorems
- Rigorous quality criteria
- Documentation

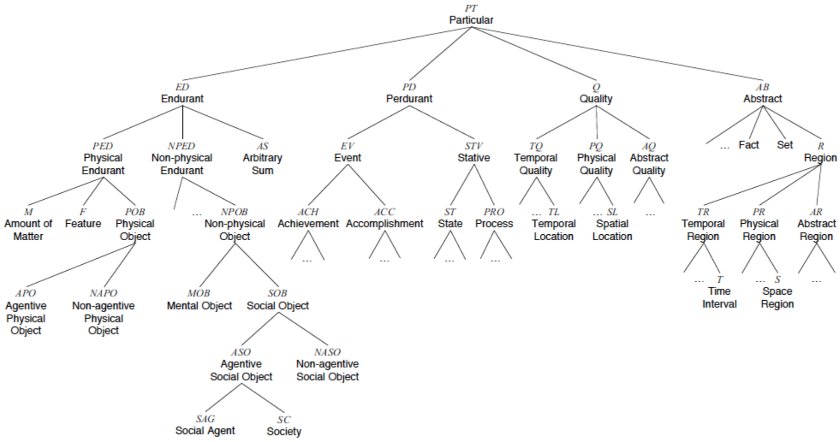
Descriptive Ontology for Linguistic and Cognitive Engineering

- Strong cognitive/linguistic bias:
  - Descriptive (as opposite to prescriptive) attitude
  - Categories mirror cognition, common sense, and the lexical structure of natural language.
- Emphasis on cognitive invariants
- Categories as conceptual containers: no 'deep' metaphysical implications
- Focus on design rationale to allow easy comparison with different ontological options
- Rigorous, systematic, interdisciplinary approach
- Rich axiomatization
  - 37 basic categories
  - 7 basic relations
  - 80 axioms, 100 definitions, 20 theorems
- Rigorous quality criteria
- Documentation

Descriptive Ontology for Linguistic and Cognitive Engineering

- Strong cognitive/linguistic bias:
  - Descriptive (as opposite to prescriptive) attitude
  - Categories mirror cognition, common sense, and the lexical structure of natural language.
- Emphasis on cognitive invariants
- Categories as conceptual containers: no ‘deep’ metaphysical implications
- Focus on design rationale to allow easy comparison with different ontological options
- Rigorous, systematic, interdisciplinary approach
- Rich axiomatization
  - 37 basic categories
  - 7 basic relations
  - 80 axioms, 100 definitions, 20 theorems
- Rigorous quality criteria
- Documentation

Rough outline of DOLCE categories



DOLCE's basic relations

- Parthood
  - Between quality regions (immediate)
  - Between arbitrary objects (temporary)
- Dependence: Specific/generic constant dependence
- Constitution
- Inherence (between a quality and its host)
- Quale
  - Between a quality and its region (immediate, for unchanging entities)
  - Between a quality and its region (temporary, for changing entities)
- Participation
- Representation

DOLCE's basic relations

- Parthood
  - Between quality regions (immediate)
  - Between arbitrary objects (temporary)
- Dependence: Specific/generic constant dependence
- Constitution
- Inherence (between a quality and its host)
- Quale
  - Between a quality and its region (immediate, for unchanging entities)
  - Between a quality and its region (temporary, for changing entities)
- Participation
- Representation

DOLCE's basic relations

- Parthood
  - Between quality regions (immediate)
  - Between arbitrary objects (temporary)
- Dependence: Specific/generic constant dependence
- Constitution
- Inherence (between a quality and its host)
- Quale
  - Between a quality and its region (immediate, for unchanging entities)
  - Between a quality and its region (temporary, for changing entities)
- Participation
- Representation

DOLCE's basic relations

- Parthood
  - Between quality regions (immediate)
  - Between arbitrary objects (temporary)
- Dependence: Specific/generic constant dependence
- Constitution
- Inherence (between a quality and its host)
- Quale
  - Between a quality and its region (immediate, for unchanging entities)
  - Between a quality and its region (temporary, for changing entities)
- Participation
- Representation

DOLCE's basic relations

- Parthood
  - Between quality regions (immediate)
  - Between arbitrary objects (temporary)
- Dependence: Specific/generic constant dependence
- Constitution
- Inherence (between a quality and its host)
- Quale
  - Between a quality and its region (immediate, for unchanging entities)
  - Between a quality and its region (temporary, for changing entities)
- Participation
- Representation

DOLCE's basic relations

- Parthood
  - Between quality regions (immediate)
  - Between arbitrary objects (temporary)
- Dependence: Specific/generic constant dependence
- Constitution
- Inherence (between a quality and its host)
- Quale
  - Between a quality and its region (immediate, for unchanging entities)
  - Between a quality and its region (temporary, for changing entities)
- Participation
- Representation

DOLCE's basic relations

- Parthood
  - Between quality regions (immediate)
  - Between arbitrary objects (temporary)
- Dependence: Specific/generic constant dependence
- Constitution
- Inherence (between a quality and its host)
- Quale
  - Between a quality and its region (immediate, for unchanging entities)
  - Between a quality and its region (temporary, for changing entities)
- Participation
- Representation

DOLCE's basic relations

- Parthood
  - Between quality regions (immediate)
  - Between arbitrary objects (temporary)
- Dependence: Specific/generic constant dependence
- Constitution
- Inherence (between a quality and its host)
- Quale
  - Between a quality and its region (immediate, for unchanging entities)
  - Between a quality and its region (temporary, for changing entities)
- Participation
- Representation

DOLCE's basic relations

- Parthood
  - Between quality regions (immediate)
  - Between arbitrary objects (temporary)
- Dependence: Specific/generic constant dependence
- Constitution
- Inherence (between a quality and its host)
- Quale
  - Between a quality and its region (immediate, for unchanging entities)
  - Between a quality and its region (temporary, for changing entities)
- Participation
- Representation

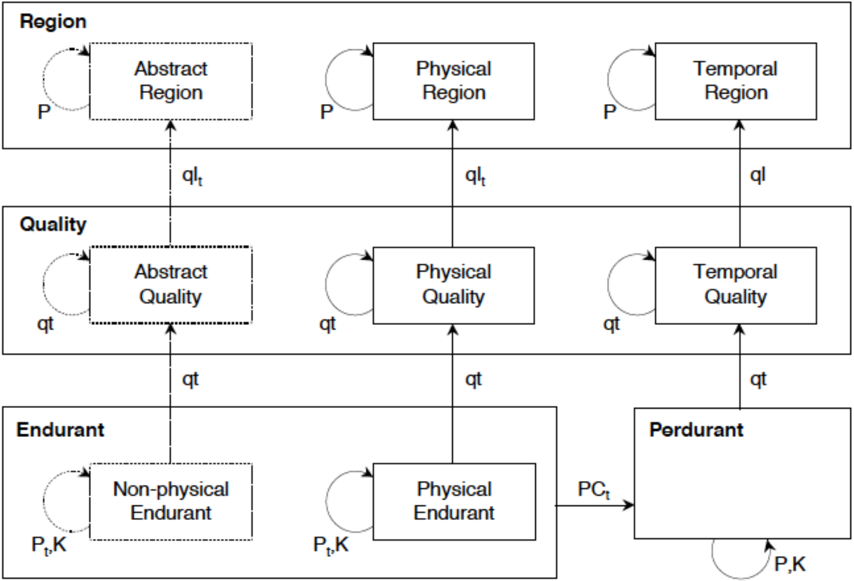
DOLCE's basic relations

- Parthood
  - Between quality regions (immediate)
  - Between arbitrary objects (temporary)
- Dependence: Specific/generic constant dependence
- Constitution
- Inherence (between a quality and its host)
- Quale
  - Between a quality and its region (immediate, for unchanging entities)
  - Between a quality and its region (temporary, for changing entities)
- Participation
- Representation

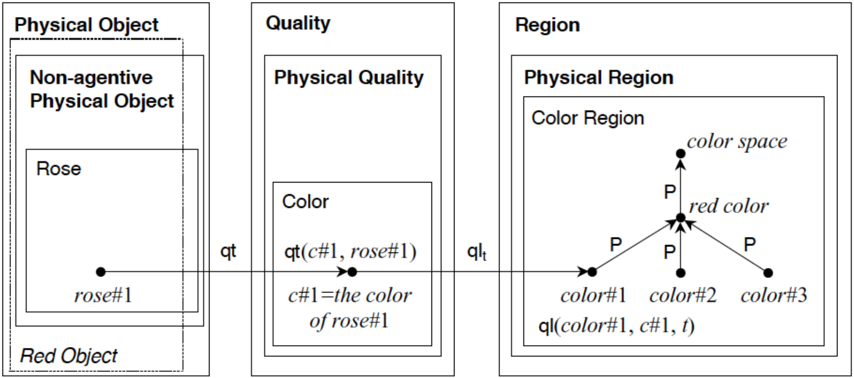
DOLCE's basic relations

- Parthood
  - Between quality regions (immediate)
  - Between arbitrary objects (temporary)
- Dependence: Specific/generic constant dependence
- Constitution
- Inherence (between a quality and its host)
- Quale
  - Between a quality and its region (immediate, for unchanging entities)
  - Between a quality and its region (temporary, for changing entities)
- Participation
- Representation

DOLCE's primitive relations between basic categories



DOLCE's basic relations (w.r.t. qualities)



DOLCE's basics on universals

(Dd1) $\mathsf{RG}(\phi) \triangleq \Box\forall x(\phi(x) \to \Box\phi(x))$ (ϕ *is Rigid*)

(Dd2) $\mathsf{NEP}(\phi) \triangleq \Box\exists x(\phi(x))$ (ϕ *is Non-Empty*)

(Dd3) $\mathsf{DJ}(\phi,\psi) \triangleq \Box\neg\exists x(\phi(x) \wedge \psi(x))$ (ϕ *and* ψ *are Disjoint*)

(Dd4) $\mathsf{SB}(\phi,\psi) \triangleq \Box\forall x(\psi(x) \to \phi(x))$ (ϕ *Subsumes* ψ)

(Dd5) $\mathsf{EQ}(\phi,\psi) \triangleq \mathsf{SB}(\phi,\psi) \wedge \mathsf{SB}(\psi,\phi)$ (ϕ *and* ψ *are Equal*)

(Dd6) $\mathsf{PSB}(\phi,\psi) \triangleq \mathsf{SB}(\phi,\psi) \wedge \neg\mathsf{SB}(\phi,\psi)$ (ϕ *Properly Subsumes* ψ)

(Dd7) $\mathsf{L}(\phi) \triangleq \Box\forall\psi(\mathsf{SB}(\phi,\psi) \to \mathsf{EQ}(\phi,\psi))$ (ϕ *is a Leaf*)

(Dd8) $\mathsf{SBL}(\phi,\psi) \triangleq \mathsf{SB}(\phi,\psi) \wedge \mathsf{L}(\psi)$ (ψ *is a Leaf Subsumed by* ϕ)

(Dd9) $\mathsf{PSBL}(\phi,\psi) \triangleq \mathsf{PSB}(\phi,\psi) \wedge \mathsf{L}(\psi)$ (ψ *is a Leaf Properly Subsumed by* ϕ)

....

DOLCE's characterisation of categories

Physical Object

(Ad32)* $\mathsf{GK}(SC, SAG)$

(Ad30)* $\mathsf{GK}(NAPO, M)$

(Ad70)* $\mathsf{OGD}(F, NAPO)$

(Ad71)* $\mathsf{OSD}(MOB, APO)$

(Ad72)* $\mathsf{OGD}(SAG, APO)$

Feature

(Ad70)* $\mathsf{OGD}(F, NAPO)$

Non-physical Endurant

(Ad12)* $\mathsf{P}(x,y,t) \rightarrow (NPED(x) \leftrightarrow NPED(y))$

(Ad22)* $\mathsf{K}(x,y,t) \rightarrow (NPED(x) \leftrightarrow NPED(y))$

(Ad41)* $\mathsf{qt}(x,y) \rightarrow (AQ(x) \leftrightarrow (AQ(y) \vee NPED(y)))$

(Ad48)* $AQ(x) \rightarrow \exists! y(\mathsf{qt}(x,y) \wedge NPED(y))$

(Ad51)* $NPED(x) \rightarrow \exists \phi, y(\mathsf{SBL}(AQ, \phi) \wedge \mathsf{qt}(\phi, y, x))$

(Ad74)* $\mathsf{OD}(NPED, PED)$

... etc...

Can all that be used?

- DOLCE in KIF
- DOLCE in OWL:
  - DOLCE-Lite: simplified translations of Dolce2.0
  - Does *not* consider: modality, temporal indexing, relation composition
  - Different names are adopted for relations that have the same name but different arities in the FOL version
  - Some commonsense concepts have been added as examples
- DOLCE-2.1-Lite-Plus version includes some modules for Plans, Information Objects, Semiotics, Temporal relations, Social notions (collectives, organizations, etc.), a Reification vocabulary, etc.

Can all that be used?

- DOLCE in KIF
- DOLCE in OWL:
  - DOLCE-Lite: simplified translations of Dolce2.0
  - Does *not* consider: modality, temporal indexing, relation composition
  - Different names are adopted for relations that have the same name but different arities in the FOL version
  - Some commonsense concepts have been added as examples
- DOLCE-2.1-Lite-Plus version includes some modules for Plans, Information Objects, Semiotics, Temporal relations, Social notions (collectives, organizations, etc.), a Reification vocabulary, etc.

Can all that be used?

- DOLCE in KIF
- DOLCE in OWL:
    - DOLCE-Lite: simplified translations of Dolce2.0
    - Does *not* consider: modality, temporal indexing, relation composition
    - Different names are adopted for relations that have the same name but different arities in the FOL version
    - Some commonsense concepts have been added as examples
- DOLCE-2.1-Lite-Plus version includes some modules for Plans, Information Objects, Semiotics, Temporal relations, Social notions (collectives, organizations, etc.), a Reification vocabulary, etc.

DLP3971

- Several Modules for (re)use: DOLCE-Lite, SocialUnits, SpatialRelations, ExtendedDnS, and others
- Still rather complex to understand (aside from using OWL-DL): Full DOLCE-Lite-Plus with 208 classes, 313 object properties, etc (check the "Active ontology" tab in Protégé) and basic DOLCE-Lite 37 classes, 70 object properties etc (in $\mathcal{SHI}$)
- Time for a DOLCE-Lite ultra-"ultralight"? e.g. for use with OWL 2 QL or OWL 2 EL
  - Current DOLCE Ultra Lite—DUL—uses friendly names and comments for classes and properties, has simple restrictions for classes, and includes into a unique file the main parts of DOLCE, D&S and other modules of DOLCE Lite+
  - BUT... is still in OWL-DL (OWL-Lite+Disjointness)
- `http://wiki.loa-cnr.it/index.php/LoaWiki:Ontologies`

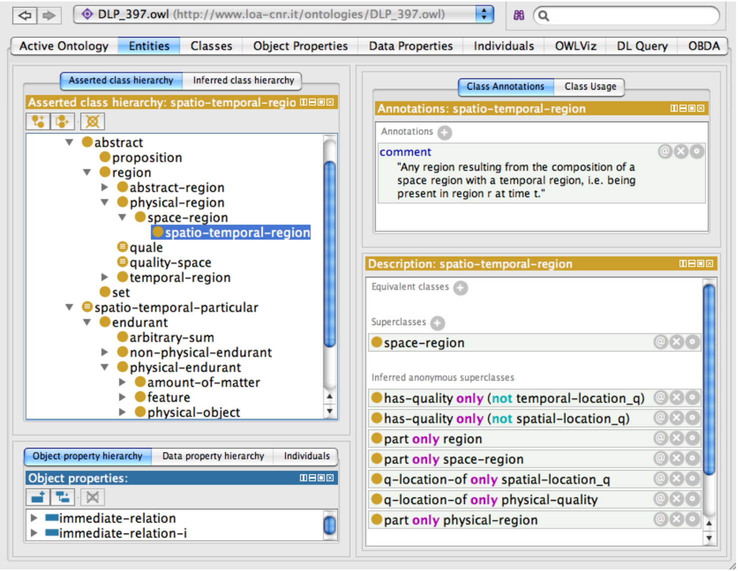
DLP3971

- Several Modules for (re)use: DOLCE-Lite, SocialUnits, SpatialRelations, ExtendedDnS, and others
- Still rather complex to understand (aside from using OWL-DL): Full DOLCE-Lite-Plus with 208 classes, 313 object properties, etc (check the "Active ontology" tab in Protégé) and basic DOLCE-Lite 37 classes, 70 object properties etc (in $\mathcal{SHI}$)
- Time for a DOLCE-Lite ultra-"ultralight"? e.g. for use with OWL 2 QL or OWL 2 EL
  - Current DOLCE Ultra Lite—DUL—uses friendly names and comments for classes and properties, has simple restrictions for classes, and includes into a unique file the main parts of DOLCE, D&S and other modules of DOLCE Lite+
  - BUT... is still in OWL-DL (OWL-Lite+Disjointness)
- http://wiki.loa-cnr.it/index.php/LoaWiki:Ontologies

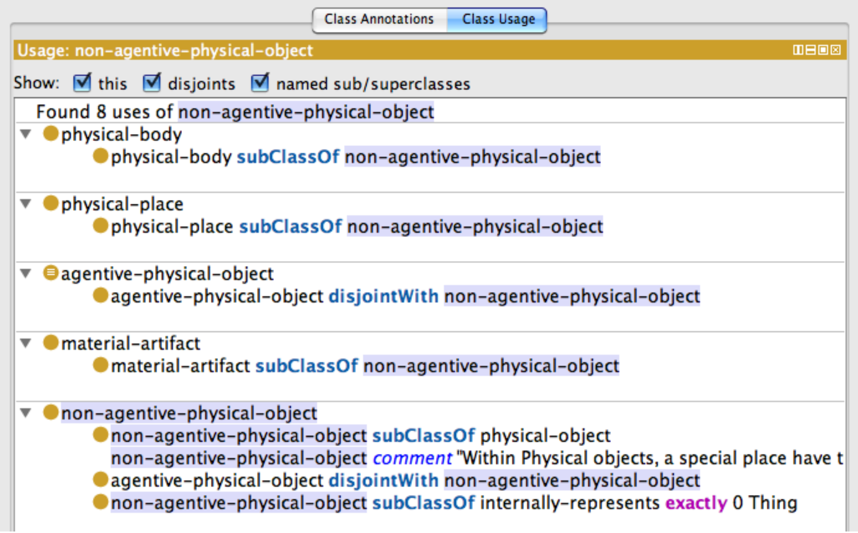
DLP3971

- Several Modules for (re)use: DOLCE-Lite, SocialUnits, SpatialRelations, ExtendedDnS, and others
- Still rather complex to understand (aside from using OWL-DL): Full DOLCE-Lite-Plus with 208 classes, 313 object properties, etc (check the "Active ontology" tab in Protégé) and basic DOLCE-Lite 37 classes, 70 object properties etc (in $\mathcal{SHI}$)
- Time for a DOLCE-Lite ultra-"ultralight"? e.g. for use with OWL 2 QL or OWL 2 EL
  - Current DOLCE Ultra Lite—DUL—uses friendly names and comments for classes and properties, has simple restrictions for classes, and includes into a unique file the main parts of DOLCE, D&S and other modules of DOLCE Lite+
  - BUT... is still in OWL-DL (OWL-Lite+Disjointness)
- http://wiki.loa-cnr.it/index.php/LoaWiki:Ontologies

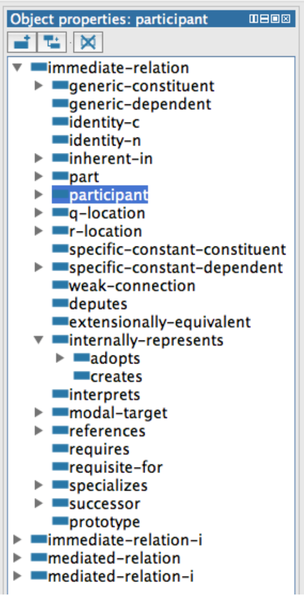
Examples


DLP_397.owl (http://www.loa-cnr.it/ontologies/DLP_397.owl)
Active Ontology | Entities | Classes | Object Properties | Data Properties | Individuals | OWLViz | DL Query | OBDA
Asserted class hierarchy | Inferred class hierarchy
Asserted class hierarchy: spatio-temporal-regio
abstract
proposition
region
abstract-region
physical-region
space-region
spatio-temporal-region
quale
quality-space
temporal-region
set
spatio-temporal-particular
endurant
arbitrary-sum
non-physical-endurant
physical-endurant
amount-of-matter
feature
physical-object
Object property hierarchy | Data property hierarchy | Individuals
Object properties:
immediate-relation
immediate-relation-i
Class Annotations | Class Usage
Annotations: spatio-temporal-region
Annotations
comment
"Any region resulting from the composition of a space region with a temporal region, i.e. being present in region r at time t."
Description: spatio-temporal-region
Equivalent classes
Superclasses
space-region
Inferred anonymous superclasses
has-quality only (not temporal-location_q)
has-quality only (not spatial-location_q)
part only region
part only space-region
q-location-of only spatial-location_q
q-location-of only physical-quality
part only physical-region

Examples


Class Annotations
Class Usage
Usage: non-agentive-physical-object
Show: this disjoints named sub/superclasses
Found 8 uses of non-agentive-physical-object
physical-body
physical-body subClassOf non-agentive-physical-object
physical-place
physical-place subClassOf non-agentive-physical-object
agentive-physical-object
agentive-physical-object disjointWith non-agentive-physical-object
material-artifact
material-artifact subClassOf non-agentive-physical-object
non-agentive-physical-object
non-agentive-physical-object subClassOf physical-object
non-agentive-physical-object comment "Within Physical objects, a special place have t
agentive-physical-object disjointWith non-agentive-physical-object
non-agentive-physical-object subClassOf internally-represents exactly 0 Thing

Comment: "The immediate relation holding between endurants and perdurants (e.g. in 'the car is running').Participation can be constant (in all parts of the perdurant, e.g. in 'the car is running'), or temporary (in only some parts, e.g. in 'I'm electing the president').A 'functional' participant is specialized for those forms of participation that depend on the nature of participants, processes, or on the intentionality of agentive participants. Traditional 'thematic role' should be mapped to functional participation.For relations holding between participants in a same perdurant, see the co-participates relation."



Outline

BFO Overview

- Ontology as reality representation
- Aims at reconciling the so-called three-dimensionalist and four-dimensionalist views
  - A Snap ontology of endurants which is reproduced at each moment of time and is used to characterize static views of the world
  - Span ontology of happenings and occurrents and, more generally, of entities which persist in time by perduring
  - Endurants (Snap) or perdurants (Span)
- Limited granularity
- Heavily influenced by parthood relations, boundaries, dependence

BFO Overview

- Ontology as reality representation
- Aims at reconciling the so-called three-dimensionalist and four-dimensionalist views
  - A Snap ontology of endurants which is reproduced at each moment of time and is used to characterize static views of the world
  - Span ontology of happenings and occurrents and, more generally, of entities which persist in time by perduring
  - Endurants (Snap) or perdurants (Span)
- Limited granularity
- Heavily influenced by parthood relations, boundaries, dependence

BFO Overview

- Ontology as reality representation
- Aims at reconciling the so-called three-dimensionalist and four-dimensionalist views
  - A Snap ontology of endurants which is reproduced at each moment of time and is used to characterize static views of the world
  - Span ontology of happenings and occurrents and, more generally, of entities which persist in time by perduring
  - Endurants (Snap) or perdurants (Span)
- Limited granularity
- Heavily influenced by parthood relations, boundaries, dependence

BFO Overview

- Ontology as reality representation
- Aims at reconciling the so-called three-dimensionalist and four-dimensionalist views
  - A Snap ontology of endurants which is reproduced at each moment of time and is used to characterize static views of the world
  - Span ontology of happenings and occurrents and, more generally, of entities which persist in time by perduring
  - Endurants (Snap) or perdurants (Span)
- Limited granularity
- Heavily influenced by parthood relations, boundaries, dependence

BFO Overview

- Ontology as reality representation
- Aims at reconciling the so-called three-dimensionalist and four-dimensionalist views
  - A Snap ontology of endurants which is reproduced at each moment of time and is used to characterize static views of the world
  - Span ontology of happenings and occurrents and, more generally, of entities which persist in time by perduring
  - Endurants (Snap) or perdurants (Span)
- Limited granularity
- Heavily influenced by parthood relations, boundaries, dependence

BFO Overview

- Ontology as reality representation
- Aims at reconciling the so-called three-dimensionalist and four-dimensionalist views
    - A Snap ontology of endurants which is reproduced at each moment of time and is used to characterize static views of the world
    - Span ontology of happenings and occurrents and, more generally, of entities which persist in time by perduring
    - Endurants (Snap) or perdurants (Span)
- Limited granularity
- Heavily influenced by parthood relations, boundaries, dependence

BFO Overview

- Ontology as reality representation
- Aims at reconciling the so-called three-dimensionalist and four-dimensionalist views
  - A Snap ontology of endurants which is reproduced at each moment of time and is used to characterize static views of the world
  - Span ontology of happenings and occurrents and, more generally, of entities which persist in time by perduring
  - Endurants (Snap) or perdurants (Span)
- Limited granularity
- Heavily influenced by parthood relations, boundaries, dependence

Overview

- BFO 1.1 in OWL with 39 classes, no object or data properties, in $\mathcal{ALC}$.
- There is a `bfo-ro.owl` to integration relations of the Relation Ontology with BFO
- Version in Isabelle (mainly part-wholes, but not all categories)
- Version in OBO (the original Gene Ontology format, with limited, but expanding, types of relationships)
- Version in Prover9 (first order logic model checker and theorem prover)

Overview

- BFO 1.1 in OWL with 39 classes, no object or data properties, in $\mathcal{ALC}$.
- There is a `bfo-ro.owl` to integration relations of the Relation Ontology with BFO
- Version in Isabelle (mainly part-wholes, but not all categories)
- Version in OBO (the original Gene Ontology format, with limited, but expanding, types of relationships)
- Version in Prover9 (first order logic model checker and theorem prover)

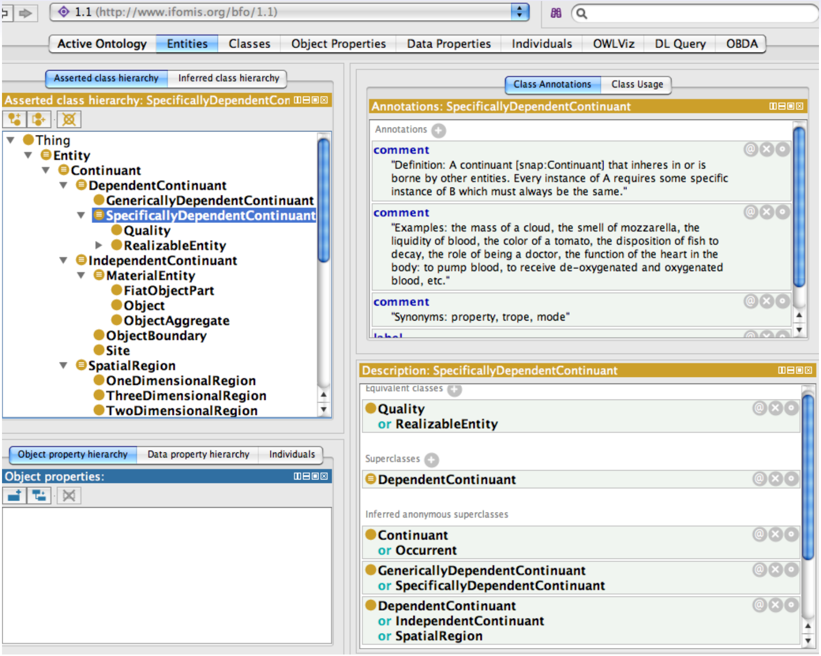
Overview

- BFO 1.1 in OWL with 39 classes, no object or data properties, in $\mathcal{ALC}$.
- There is a `bfo-ro.owl` to integration relations of the Relation Ontology with BFO
- Version in Isabelle (mainly part-wholes, but not all categories)
- Version in OBO (the original Gene Ontology format, with limited, but expanding, types of relationships)
- Version in Prover9 (first order logic model checker and theorem prover)

BFO Taxonomy

```
bfo:Entity
  snap:Continuant
    snap:DependentContinuant
      snap:GenericalyDependentContinuant
      snap:SpecificalyDependentContinuant
        snap:Quality
        snap:RealizableEntity
          snap:Disposition
          snap:Function
          snap:Role
    snap:IndependentContinuant
      snap:MaterialEntity
          snap:Object
          snap:FiatObjectPart
          snap:ObjectAggregate
      snap:ObjectBoundary
      snap:Site
    snap:SpatialRegion
      snap:ZeroDimensionalRegion
      snap:OneDimensionalRegion
      snap:TwoDimensionalRegion
      snap:ThreeDimensionalRegion
```

```
span:Occurrent
  span:ProcessualEntity
    span:Process
    span:ProcessBoundary
    span:FiatProcessPart
    span:ProcessAggregate
    span:ProcessualContext
  span:SpatiotemporalRegion
    span:ConnectedTemporalRegion
      span:SpatiotemporalInstant
      span:SpatiotemporalInterval
    span:ScatteredSpatiotemporalRegion
  span:TemporalRegion
    span:ConnectedSpatiotemporalRegion
      span:TemporalInstant
      span:TemporalInterval
    span:ScatteredTemporalRegion
```


1.1 (http://www.ifomis.org/bfo/1.1)
Active Ontology
Entities
Classes
Object Properties
Data Properties
Individuals
OWLViz
DL Query
OBDA
Asserted class hierarchy
Inferred class hierarchy
Asserted class hierarchy: SpecificallyDependentCon
Thing
Entity
Continuant
DependentContinuant
GenericallyDependentContinuant
SpecificallyDependentContinuant
Quality
RealizableEntity
IndependentContinuant
MaterialEntity
FiatObjectPart
Object
ObjectAggregate
ObjectBoundary
Site
SpatialRegion
OneDimensionalRegion
ThreeDimensionalRegion
TwoDimensionalRegion
Object property hierarchy
Data property hierarchy
Individuals
Object properties:
Class Annotations
Class Usage
Annotations: SpecificallyDependentContinuant
Annotations
comment
"Definition: A continuant [snap:Continuant] that inheres in or is borne by other entities. Every instance of A requires some specific instance of B which must always be the same."
comment
"Examples: the mass of a cloud, the smell of mozzarella, the liquidity of blood, the color of a tomato, the disposition of fish to decay, the role of being a doctor, the function of the heart in the body: to pump blood, to receive de-oxygenated and oxygenated blood, etc."
comment
"Synonyms: property, trope, mode"
Description: SpecificallyDependentContinuant
Equivalent classes
Quality
or RealizableEntity
Superclasses
DependentContinuant
Inferred anonymous superclasses
Continuant
or Occurrent
GenericallyDependentContinuant
or SpecificallyDependentContinuant
DependentContinuant
or IndependentContinuant
or SpatialRegion

BFO Core

- A non-extensional temporal mereology with collections, sums, and universals
- BFO as a collection of smaller theories
  - EMR, QSizeR, RBG, QDiaSizeR, ..., Adjacency, Collections, SumsPartitions, Universals, Instantiation, ExtensionsOfUniversals, PartonomicInclusion, UniversalParthood
- Reference material `http://www.ifomis.org/bfo/fol` and `http://www.acsu.buffalo.edu/∼bittner3/Theories/BFO/`

Section of one of the sub-theories in BFO Core

```
theory UniversalParthood

imports ExtensionsOfUniversals PartonomicInclusion

begin

consts


UPt1 :: Un => Un => Ti => o
UPt2 :: Un => Un => Ti => o
UPt12 :: Un => Un => Ti => o

UP1 :: Un => Un => o
UP2 :: Un => Un  => o
UP12 :: Un => Un  => o

defs

UPt1-def: UPt1(c,d,t) == (ALL x. (Inst(x,c,t) --> (EX y. (Inst(y,d,t) & P(x,y,t)))))
UPt2-def: UPt2(c,d,t) == (ALL y. (Inst(y,d,t) --> (EX x. (Inst(x,c,t) & P(x,y,t)))))
UPt12-def: UPt12(c,d,t) == UPt1(c,d,t) & UPt2(c,d,t)

UP1-def: UP1(c,d) == (ALL t. UPt1(c,d,t))
UP2-def: UP2(c,d) == (ALL t. UPt2(c,d,t))
UP12-def: UP12(c,d) == (ALL t. UPt12(c,d,t))
```

Outline

Ontology and ontologies

Foundational Ontologies
- DOLCE
- BFO

Ontology Design Patterns
- Overview
- Patterns

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

content of slides based on Presutti et al, 2008

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

content of slides based on Presutti et al, 2008

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

content of slides based on Presutti et al, 2008

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

content of slides based on Presutti et al, 2008

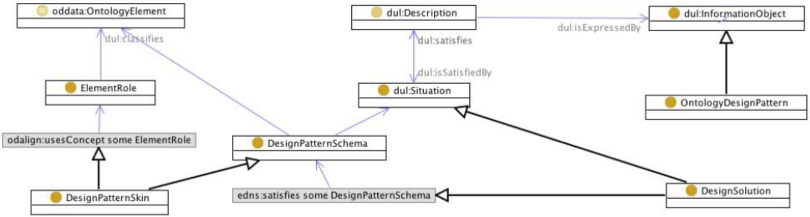
Rationale

- 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

content of slides based on Presutti et al, 2008

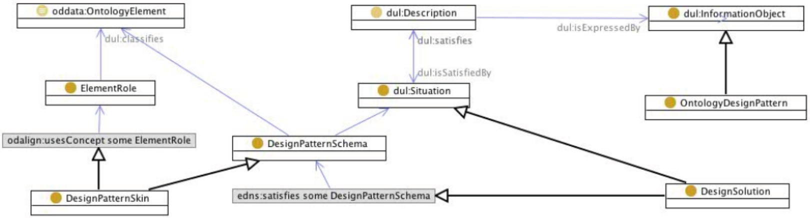
ODP definition

- An ODP is an information object
- A design pattern schema is the description of an ODP, including the roles, tasks, and parameters needed in order to solve an ontology design issue
- *An ODP is a modeling solution to solve a recurrent ontology design problem. It is an Information Object that expresses a Design Pattern Schema (or skin) that can only be satisfied by DesignSolutions. Design solutions provide the setting for Ontology Elements that play some ElementRole(s) from the schema.* (Presutti et al, 2008)



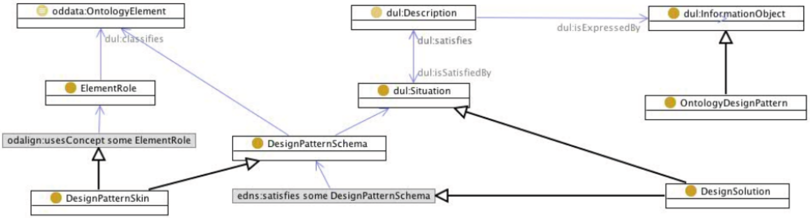
ODP definition

- An ODP is an information object
- A design pattern schema is the description of an ODP, including the roles, tasks, and parameters needed in order to solve an ontology design issue
- *An ODP is a modeling solution to solve a recurrent ontology design problem. It is an Information Object that expresses a Design Pattern Schema (or skin) that can only be satisfied by DesignSolutions. Design solutions provide the setting for Ontology Elements that play some ElementRole(s) from the schema.* (Presutti et al, 2008)



ODP definition

- An ODP is an information object
- A design pattern schema is the description of an ODP, including the roles, tasks, and parameters needed in order to solve an ontology design issue
- *An ODP is a modeling solution to solve a recurrent ontology design problem. It is an Information Object that expresses a Design Pattern Schema (or skin) that can only be satisfied by DesignSolutions. Design solutions provide the setting for Ontology Elements that play some ElementRole(s) from the schema.* (Presutti et al, 2008)



Outline

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—next lecture)
- 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

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—next lecture)
- 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

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—next lecture)
- 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

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—next lecture)
- 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

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—next lecture)
- 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

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—next lecture)
- 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

Structural OPs

- Logical OPs:
  - Are compositions of logical constructs that solve a problem of expressivity in OWL-DL (and, in cases, also in OWL 2 DL)
  - Only expressed in terms of a logical vocabulary, because their signature (the set of predicate names, e.g. the set of classes and properties in an OWL ontology) is empty
  - Independent from a specific domain of interest
  - **Logical macros** compose OWL DL constructs; e.g. the universal+existential OWL macro
  - **Transformation patterns** translate a logical expression from a logical language into another; e.g. n-aries

Structural OPs

- Logical OPs:
  - Are compositions of logical constructs that solve a problem of expressivity in OWL-DL (and, in cases, also in OWL 2 DL)
  - Only expressed in terms of a logical vocabulary, because their signature (the set of predicate names, e.g. the set of classes and properties in an OWL ontology) is empty
  - Independent from a specific domain of interest
  - **Logical macros** compose OWL DL constructs; e.g. the universal+existential OWL macro
  - **Transformation patterns** translate a logical expression from a logical language into another; e.g. n-aries

Structural OPs

- Logical OPs:
  - Are compositions of logical constructs that solve a problem of expressivity in OWL-DL (and, in cases, also in OWL 2 DL)
  - Only expressed in terms of a logical vocabulary, because their signature (the set of predicate names, e.g. the set of classes and properties in an OWL ontology) is empty
  - Independent from a specific domain of interest
  - **Logical macros** compose OWL DL constructs; e.g. the universal+existential OWL macro
  - **Transformation patterns** translate a logical expression from a logical language into another; e.g. n-aries

Structural OPs

- Logical OPs:
  - Are compositions of logical constructs that solve a problem of expressivity in OWL-DL (and, in cases, also in OWL 2 DL)
  - Only expressed in terms of a logical vocabulary, because their signature (the set of predicate names, e.g. the set of classes and properties in an OWL ontology) is empty
  - Independent from a specific domain of interest
  - **Logical macros** compose OWL DL constructs; e.g. the universal+existential OWL macro
  - **Transformation patterns** translate a logical expression from a logical language into another; e.g. n-aries

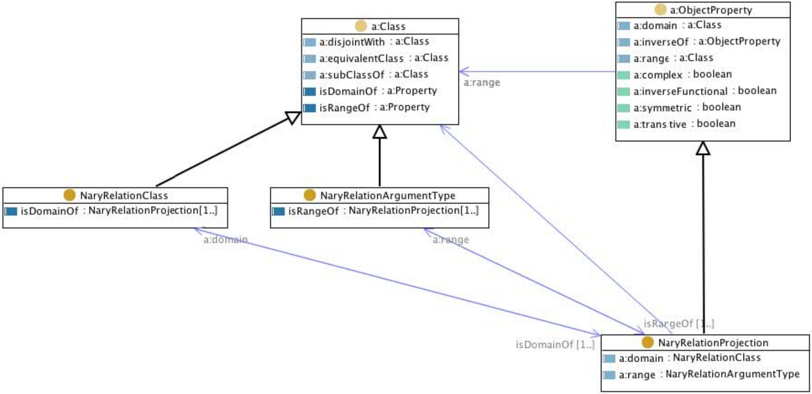
Structural OPs

- Logical OPs:
  - Are compositions of logical constructs that solve a problem of expressivity in OWL-DL (and, in cases, also in OWL 2 DL)
  - Only expressed in terms of a logical vocabulary, because their signature (the set of predicate names, e.g. the set of classes and properties in an OWL ontology) is empty
  - Independent from a specific domain of interest
  - **Logical macros** compose OWL DL constructs; e.g. the universal+existential OWL macro
  - **Transformation patterns** translate a logical expression from a logical language into another; e.g. n-aries

Structural OPs

- Logical OPs:
  - Are compositions of logical constructs that solve a problem of expressivity in OWL-DL (and, in cases, also in OWL 2 DL)
  - Only expressed in terms of a logical vocabulary, because their signature (the set of predicate names, e.g. the set of classes and properties in an OWL ontology) is empty
  - Independent from a specific domain of interest
  - **Logical macros** compose OWL DL constructs; e.g. the universal+existential OWL macro
  - **Transformation patterns** translate a logical expression from a logical language into another; e.g. n-aries

Example: n-ary relation Logical OP



Architectural OPs

- Architectural OPs are defined in terms of composition of Logical OPs that are used in order to affect the overall shape of the ontology; i.e., an Architectural OP identifies a composition of Logical OPs that are to be exclusively used in the design of an ontology
- Examples of Architectural OPs are: Taxonomy, Modular Architecture, and Lightweight Ontology
- E.g., **Modular Architecture** Architectural OP consists of an ontology network, where the involved ontologies play the role of modules, which are connected by the *owl:import* operation with one root ontology that imports all the modules

Architectural OPs

- Architectural OPs are defined in terms of composition of Logical OPs that are used in order to affect the overall shape of the ontology; i.e., an Architectural OP identifies a composition of Logical OPs that are to be exclusively used in the design of an ontology
- Examples of Architectural OPs are: Taxonomy, Modular Architecture, and Lightweight Ontology
- E.g., **Modular Architecture** Architectural OP consists of an ontology network, where the involved ontologies play the role of modules, which are connected by the *owl:import* operation with one root ontology that imports all the modules

Architectural OPs

- Architectural OPs are defined in terms of composition of Logical OPs that are used in order to affect the overall shape of the ontology; i.e., an Architectural OP identifies a composition of Logical OPs that are to be exclusively used in the design of an ontology
- Examples of Architectural OPs are: Taxonomy, Modular Architecture, and Lightweight Ontology
- E.g., **Modular Architecture** Architectural OP consists of an ontology network, where the involved ontologies play the role of modules, which are connected by the *owl:import* operation with one root ontology that imports all the modules

Reasoning OPs

- Applications of Logical OPs oriented to obtain certain reasoning results, based on the behavior implemented in a reasoning engine
- Examples of Reasoning OPs include: classification, subsumption, inheritance, materialization, and de-anonymizing
- Inform about the state of that ontology, and let a system decide what reasoning has to be performed on the ontology in order to carry out queries, evaluation, etc
- Name all relevant classes, so no anonymous complex class descriptions are left (restriction deanonymizing), Name anonymous individuals (skolem de-anonymizing), Materialize the subsumption hierarchy (automatic subsumption) and normalize names, Instantiate the deepest possible class or property, Normalize property instances (property value materialization)

Lexico-Syntactic OPs

- linguistic structures or schemas that consist of certain types of words following a specific order and that permit to generalize and extract some conclusions about the meaning they express; *verbalisation* patterns
- E.g., "subClassOf" relation, NP<subclass> be NP<superclass>, a Noun Phrase should appear before the verb—represented by its basic form or lemma, be in this example—and the verb should in its turn be followed by another Noun Phrase
- Other Lexical OPs provided for OWL's equivalence between classes, object property, subpropertyOf relation, datatype property, existential restriction, universal restriction, disjointness, union of classes
- For English language only, thus far
- Similar to idea of specification of ORM's verbalization templates

Lexico-Syntactic OPs

- linguistic structures or schemas that consist of certain types of words following a specific order and that permit to generalize and extract some conclusions about the meaning they express; *verbalisation* patterns
- E.g., "subClassOf" relation, NP<subclass> be NP<superclass>, a Noun Phrase should appear before the verb—represented by its basic form or lemma, be in this example—and the verb should in its turn be followed by another Noun Phrase
- Other Lexical OPs provided for OWL's equivalence between classes, object property, subpropertyOf relation, datatype property, existential restriction, universal restriction, disjointness, union of classes
- For English language only, thus far
- Similar to idea of specification of ORM's verbalization templates

Lexico-Syntactic OPs

- linguistic structures or schemas that consist of certain types of words following a specific order and that permit to generalize and extract some conclusions about the meaning they express; *verbalisation* patterns
- E.g., "subClassOf" relation, NP<subclass> be NP<superclass>, a Noun Phrase should appear before the verb—represented by its basic form or lemma, be in this example—and the verb should in its turn be followed by another Noun Phrase
- Other Lexical OPs provided for OWL's equivalence between classes, object property, subpropertyOf relation, datatype property, existential restriction, universal restriction, disjointness, union of classes
- For English language only, thus far
- Similar to idea of specification of ORM's verbalization templates

Lexico-Syntactic OPs

- linguistic structures or schemas that consist of certain types of words following a specific order and that permit to generalize and extract some conclusions about the meaning they express; *verbalisation* patterns
- E.g., "subClassOf" relation, NP<subclass> be NP<superclass>, a Noun Phrase should appear before the verb—represented by its basic form or lemma, be in this example—and the verb should in its turn be followed by another Noun Phrase
- Other Lexical OPs provided for OWL's equivalence between classes, object property, subpropertyOf relation, datatype property, existential restriction, universal restriction, disjointness, union of classes
- For English language only, thus far
- Similar to idea of specification of ORM's verbalization templates

Lexico-Syntactic OPs

- linguistic structures or schemas that consist of certain types of words following a specific order and that permit to generalize and extract some conclusions about the meaning they express; *verbalisation* patterns
- E.g., "subClassOf" relation, NP<subclass> be NP<superclass>, a Noun Phrase should appear before the verb—represented by its basic form or lemma, be in this example—and the verb should in its turn be followed by another Noun Phrase
- Other Lexical OPs provided for OWL's equivalence between classes, object property, subpropertyOf relation, datatype property, existential restriction, universal restriction, disjointness, union of classes
- For English language only, thus far
- Similar to idea of specification of ORM's verbalization templates

How to create an ODP

- See chapter 3 of (Presutti et al., 2008)
- From where do ODPs come from (section 3.4—in part: lagacy sources, which we deal with in the next lecture)
- Annotation schema
- How to use them
- Content Ontology Design Anti-pattern (AntiCP)

How to create an ODP

- See chapter 3 of (Presutti et al., 2008)
- From where do ODPs come from (section 3.4—in part: lagacy sources, which we deal with in the next lecture)
- Annotation schema
- How to use them
- Content Ontology Design Anti-pattern (AntiCP)

How to create an ODP

- See chapter 3 of (Presutti et al., 2008)
- From where do ODPs come from (section 3.4—in part: lagacy sources, which we deal with in the next lecture)
- Annotation schema
- How to use them
- Content Ontology Design Anti-pattern (AntiCP)

How to create an ODP

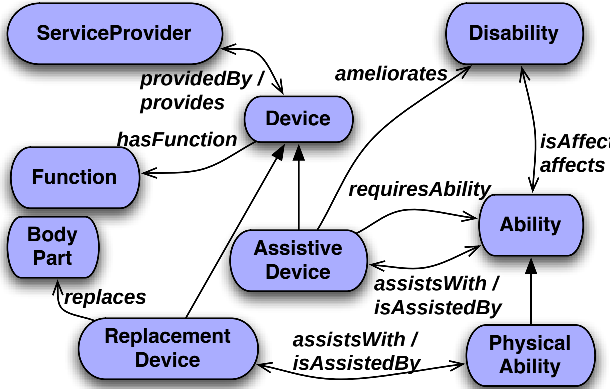
- See chapter 3 of (Presutti et al., 2008)
- From where do ODPs come from (section 3.4—in part: lagacy sources, which we deal with in the next lecture)
- Annotation schema
- How to use them
- Content Ontology Design Anti-pattern (AntiCP)

How to create an ODP

- See chapter 3 of (Presutti et al., 2008)
- From where do ODPs come from (section 3.4—in part: lagacy sources, which we deal with in the next lecture)
- Annotation schema
- How to use them
- Content Ontology Design Anti-pattern (AntiCP)

Sample exercise: an ODP for the ADOLENA ontology?

- Novel Abilities and Disabilities OntoLogy for ENhancing Accessibility: ADOLENA
- Can this be engineered into an ODP? If so, which type(s), how, what information is needed to document an ODP?



Summary