Tutorial: Semantik und Interoperabilität

Dr.-Ing. Christoph Stahl
christoph.stahl@dfki.de
DFKI - FB Cyber Physical Systems, Bremen, Germany
Workshop Interoperabilität, AAL Kongress 2012

Inhalt

- Syntax, Semantik und Interoperabilität
- Was ist eine Ontologie
  - Wofür, Klassifikation
- Beispiele Informeller Ontologien
- Logik-basierte Ontologien
  - Description Logics, A-Box und T-Box
  - Reasoning
- Semantisches Web
  - RDF, OWL
  - Protégé Demo mit DogOnt Beispielen

Semantik und Interoperabilität

- Kommunikation basiert auf dem Austausch von Information zwischen Sender und Empfänger, diese muss in einer bestimmten Sprache formuliert sein.
- Eine Sprache besteht aus Symbolen, die nach bestimmten Regeln angeordnet sind (**Syntax**).
- Es muss sich auf eine **Interpretationsfunktion** geeinigt werden, die den Symbolen eine Bedeutung (**Semantik**) in der realen Welt zuordnet.

Interpretation des Terms „Ziege“

Konzept (Semantik)

Ziege

Interpretationsfunktion

Symbol (Syntax)

Referent (Reale Welt)

Was ist eine Ontologie?

- Aristoteles (4. Jahrh. v. Chr)
  - Philosophie als Wissenschaft des Seienden; Metaphysik untersucht was Dinge, Eigenschaften oder Prozesse ihrem Wesen nach sind und in welchem Verhältnis sie zueinander stehen.
- Gruber, 1993
  - **explizite Spezifikation einer Konzeptualisierung**
- Uschold und Grüninger, 1996
  - ein **gemeinsames Verständnis einer bestimmten Domäne von Interesse**
- Guarino, 1998
  - Engineering artifact
  - Ein spezifisches Vokabular, um eine bestimmte Domäne zu beschreiben
  - Explizite Annahmen bezüglich der Bedeutung der Vokabeln: Konzepte und Relationen

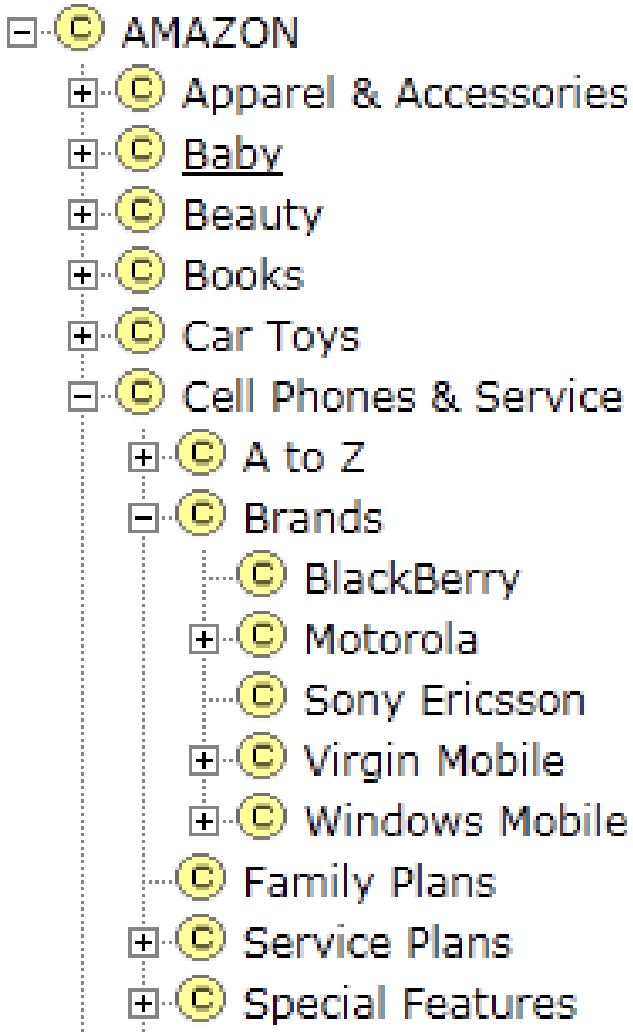
Wofür Ontologien?

- Ontologien verbessern die Kommunikation zwischen Menschen bzw. Maschinen durch die Spezifikation der Semantik der verwendeten Symbole
- Drei Hauptanwendungen für Ontologien (Jasper und Uschold, 1999):
  I. Kommunikation zwischen Menschen
  II. Interoperabilität von Softwaresystemen
  III. Qualitätsverbesserung bei der Systementwicklung bezüglich Spezifikation, Zuverlässigkeit und Wiederverwendbarkeit

Klassifikation von Ontologien

- Wir können unterscheiden zwischen
- Informellen Ontologien
  - Kataloge: Aufzählung von Objekten, implizite Semantik basierend auf Allgemeinwissen
  - Glossare, Lexika, Normen: Explizite Semantik gegeben durch Begriffsdefinitionen in natürlicher Sprache
- Logik-basierten Ontologien
  - Taxonomien: formale Spezifikation durch *is-a* Relation
  - Konzepte und Klassen: Axiomatisierung
  - Wissensrepräsentation: T-Box und A-Box mit Fakten

Amazon Produktkategorien



Informelle Ontologien

- Beispielsweise Standards und Industrienormen, W3C, ISO, DIN etc.
- Die Bedeutung von Fachbegriffen wird dem Leser mittels einer **Beschreibung in natürlicher Sprache** erklärt.
- Ambiguität von Begriffen und Symbolen wird von vorneherein vermieden (aber nur innerhalb der Domäne).
- Eine formale (logikbasierte) Spezifikation der Semantik wird in der Praxis für Softwaresysteme oft als nicht notwendig erachtet; die **Bedeutung von Symbolen** wird im Programm **hartkodiert**.
- **Mehrdeutigkeit** lässt unterschiedliche Interpretationen durch den Programmierer zu, diese führen zu unterschiedlichem Systemverhalten; **Inkonsistenzen** sind möglich.

Siehe auch M. Uschold: Where are the semantics in the Semantic Web

Beispiel Programmiersprachen

- Objektorientierte Sprachen definieren Klassen mit Attributen und Methoden
- Bedeutung der Methoden geht für Programmierer nur aus der Dokumentation hervor (und Hintergrundwissen)
- Portierung von Code problematisch, nicht automatisierbar

JAVA API

- Concat(String str)
- Equals(Object obj)
- indexOf(String str)
- toUpperCase()

Converts all of the characters **in this** String to upper case using the rules of the default locale.

C# .NET Framework

- Static Concat(String s1, String s2)
- Equals(String s)
- IndexOf(String str)
- toUpper()

Returns a **copy** of this String in uppercase.

Identisches Verhalten?

Beispiel: W3C HTML 4.01 Specification

W3C Recommendation

11.2 Elements for constructing tables

11.2.1 The TABLE element

```
<!ELEMENT TABLE - -
     (CAPTION?, (COL*|COLGROUP*), THEAD?, TFOOT?, TBODY+)>
<!ATTLIST TABLE                          -- table element --
  %attrs;                                -- %coreattrs, %i18n, %events --
  summary     %Text;         #IMPLIED    -- purpose/structure for speech output--
  width       %Length;       #IMPLIED    -- table width --
  border      %Pixels;       #IMPLIED    -- controls frame width around table --
  frame       %TFrame;       #IMPLIED    -- which parts of frame to render --
  rules       %TRules;       #IMPLIED    -- rulings between rows and cols --
  cellspacing %Length;       #IMPLIED    -- spacing between cells --
  cellpadding %Length;       #IMPLIED    -- spacing within cells --
  >
```

Start tag: ***required****, End tag:* ***required***

Attribute definitions

summary = *text* [CS]
This attribute provides a summary of the table's purpose and structure for user agents rendering to non-visual media such as speech and Braille.

align = left|center|right [CI]
Deprecated. This attribute specifies the position of the table with respect to the document. Permitted values:

- `left`: The table is to the left of the document.
- `center`: The table is to the center of the document.
- `right`: The table is to the right of the document.

width = *length* [CN]
This attribute specifies the desired width of the entire table and is intended for visual user agents. When the value is a percentage value, the value is relative to the user agent's available horizontal space. In the absence of any width specification, table width is determined by the user agent.

Attributes defined elsewhere

- id, class (document-wide identifiers)
- lang (language information), dir (text direction)
- title (element title)
- style (inline style information)

Fehlende Semantik von Inhalten


```
<table class="vertical-top">
 <tbody>
 <tr>
 <td>
<p>E-Mail: <a class="external-link" href =
"mailto:info@dfki.de">info@dfki.de</a></p>
<p>Der Standort des DFKI Bremen befindet sich auf dem Gelände der <a
href="http://www.uni-bremen.de/" target="_self">Universität
Bremen</a> </p>
<p><a href =
"http://maps.google.com/maps/ms?f=q&amp;hl=de&amp;geocode=&amp;ie=UTF8&amp;msa=0&amp;msid=113001974924518077858.0004436fc764cabd3fcc0&amp;ll=53.194516,8.762283&amp;spn=0.284255,0.620728&amp;z=11&amp;om=1&amp;iwloc=000443708b70c2604de56">Google Maps</a></p>
<p><strong><a class="noarrow" name="robotik"></a>Forschungsbereich
Robotics Innovation Center<br /></strong><br />
DFKI Bremen<br />
Forschungsbereich Robotics Innovation Center (RIC)<br />
Robert-Hooke-Straße 5<br />
D-28359 Bremen<br />
Tel.: +49 (0)421 / 178 45-4100<br />
Fax: +49 (0)421 / 178 45-4150<br />
E-Mail: <a href="mailto:robotik@dfki.de"
target="_self">robotik@dfki.de</a><br />
Web: <a href="../../robotik"
target="_self">http://www.dfki.de/robotik</a></p>
<p>Anreise mit dem <strong>Auto</strong>:</p>
```

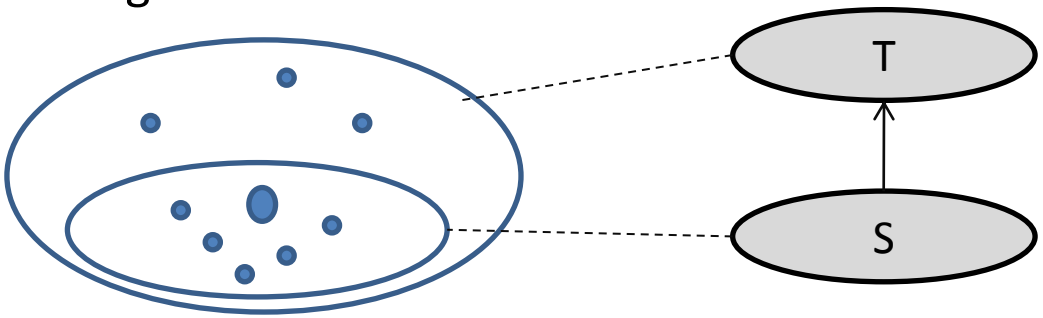
HTML spezifiziert lediglich Layout und Hyperlinks, nicht die Bedeutung von Inhalten

Formale Ontologien

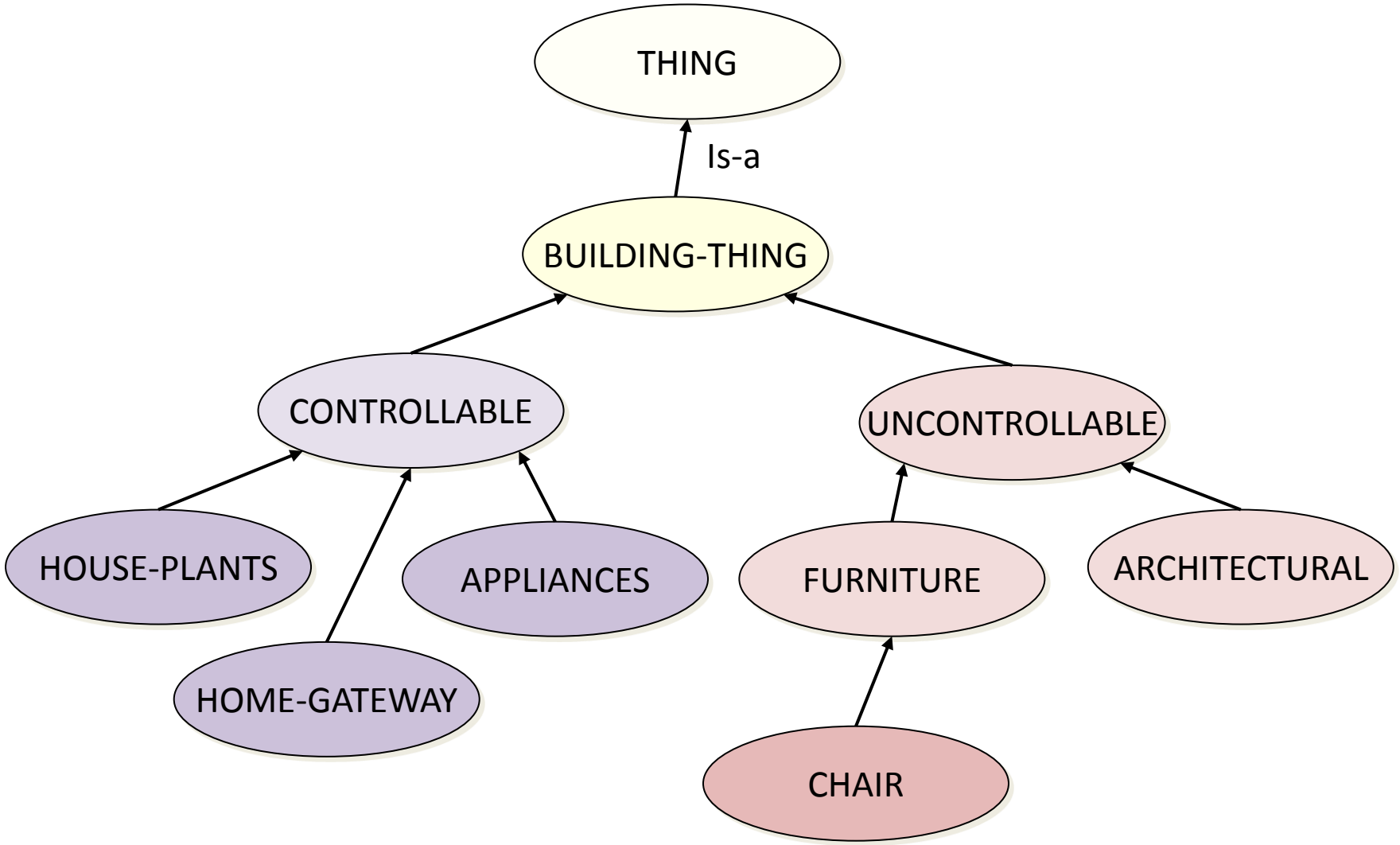
- Formale Sprache, bestehend aus Konzepten (Begriffen), Relationen und Axiomen
- Relationen beschreiben die Beziehungen zwischen Konzepten
- Eine **Taxonomie** benötigt mindestens eine ***is-a* Relation** zur hierarchischen Strukturierung von Konzepten
- Syntax kann als Formelsammlung oder Graph dargestellt werden.
- Semantik wird z.B. in OWL-DL durch **Description Logics (DL)** spezifiziert
- Vollständige Axiomatisierung ist möglich, z.B. in OWL-FULL
- Die **Zulässigkeit bestimmte Interpretationen** der Ontologie lässt sich anhand der Axiome und Semantik **maschinell prüfen**.

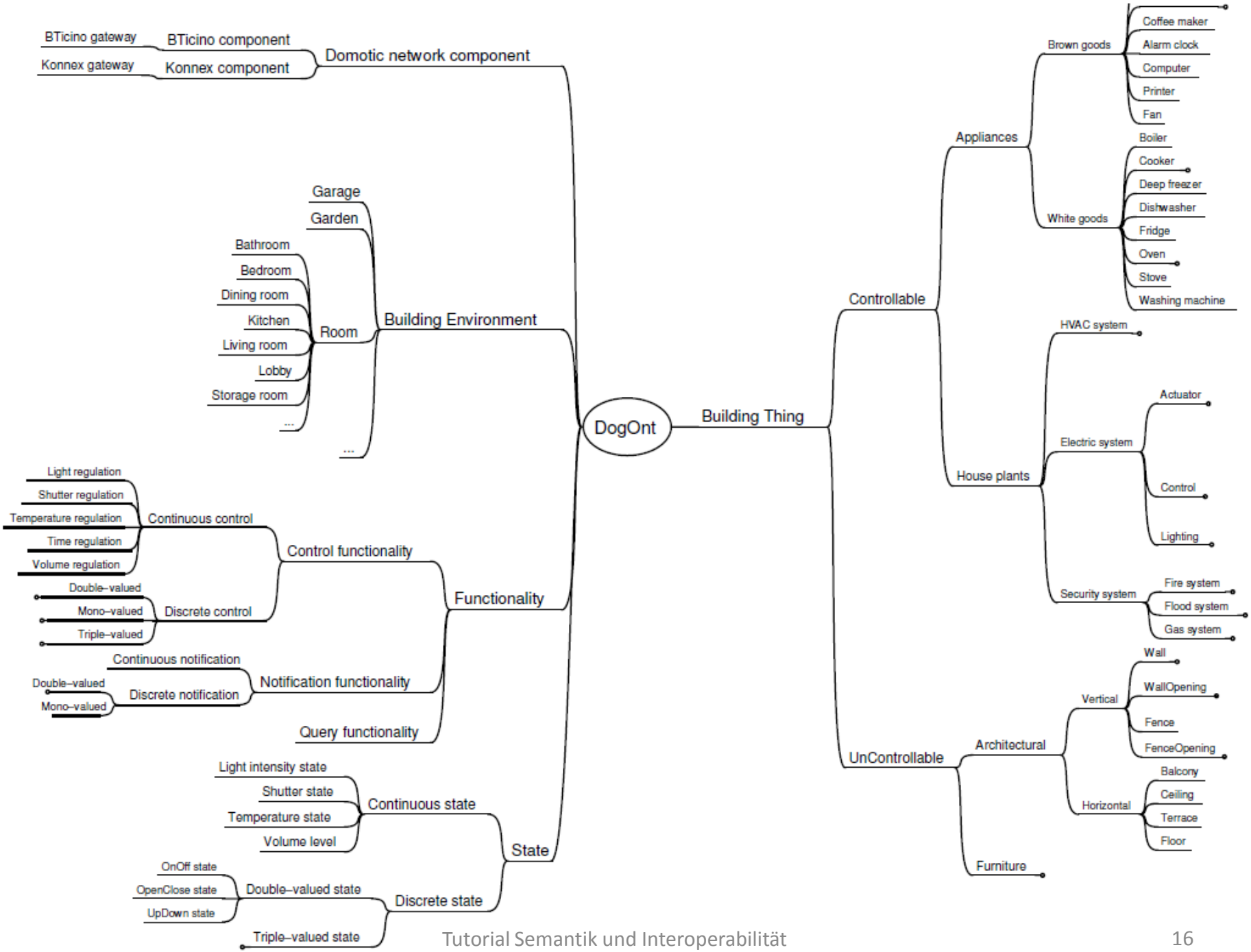
Konzepte und Taxonomien

- Das **Konzept** als elementare Wissenseinheit repräsentiert eine Klasse von Dingen innerhalb einer Domäne
  - Benannt nach den Elementen der Klasse
  - Graphisch dargestellt als Ellipse mit Bezeichner, im Stile von *NIKL*
- Semantik kann über Mengen definiert werden
  - Ein Konzept S repräsentiert eine Menge von Elementen, gegeben durch eine Interpretationsfunktion I
- **Subsumption**
  - Sei Konzept T eine andere Menge von Elementen
  - Falls S $\subset$ T, so sagt man
    - S spezialisiert T (is-a)
    - T subsumiert S



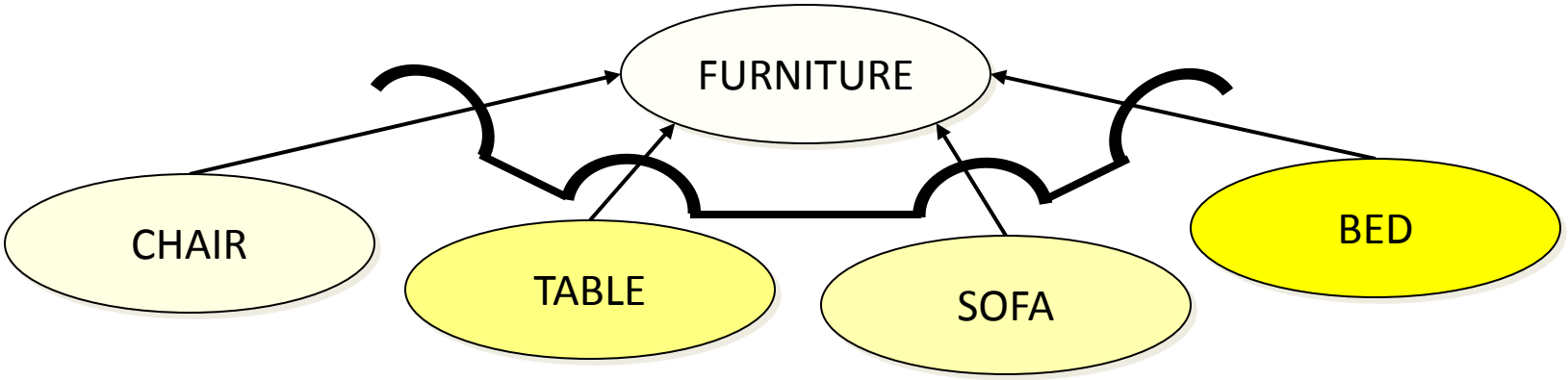
is-a Relation zwischen Konzepten




DogOnt
Building Thing
Domotic network component
BTicino component
BTicino gateway
Konnex component
Konnex gateway
Building Environment
Room
Garage
Garden
Bathroom
Bedroom
Dining room
Kitchen
Living room
Lobby
Storage room
...
...
Functionality
Control functionality
Continuous control
Light regulation
Shutter regulation
Temperature regulation
Time regulation
Volume regulation
Discrete control
Double–valued
Mono–valued
Triple–valued
Notification functionality
Continuous notification
Discrete notification
Double–valued
Mono–valued
Query functionality
State
Continuous state
Light intensity state
Shutter state
Temperature state
Volume level
Discrete state
Double–valued state
OnOff state
OpenClose state
UpDown state
Triple–valued state
Controllable
Appliances
Brown goods
Coffee maker
Alarm clock
Computer
Printer
Fan
White goods
Boiler
Cooker
Deep freezer
Dishwasher
Fridge
Oven
Stove
Washing machine
House plants
HVAC system
Electric system
Actuator
Control
Lighting
Security system
Fire system
Flood system
Gas system
UnControllable
Architectural
Vertical
Wall
WallOpening
Fence
FenceOpening
Horizontal
Balcony
Ceiling
Terrace
Floor
Furniture

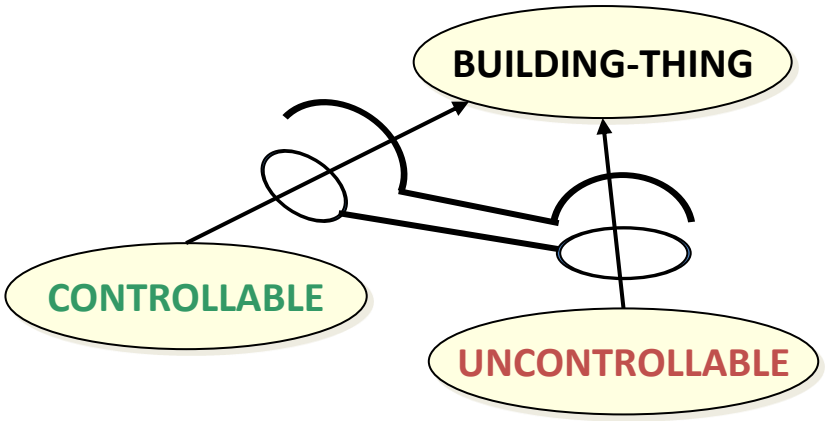
Disjunkte Klassen

- Keine Überschneidung der Konzepte
- Elemente können nur einer Unterklasse angehören, nicht mehreren



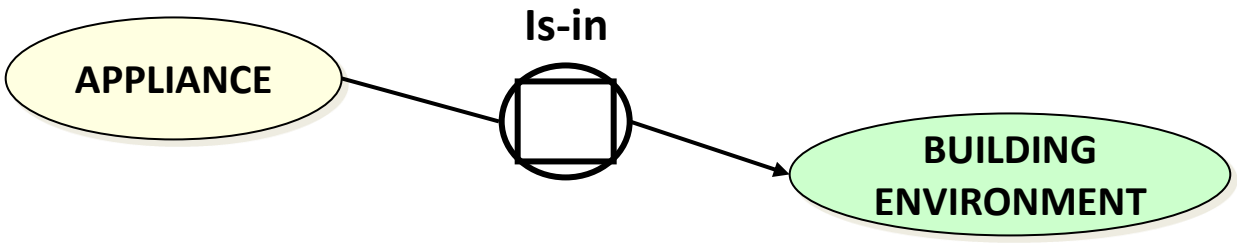
Partitionierung

- Die (disjunkten) Unterklassen decken die Oberklasse vollständig ab.
- Elemente müssen genau einer Unterklasse angehören



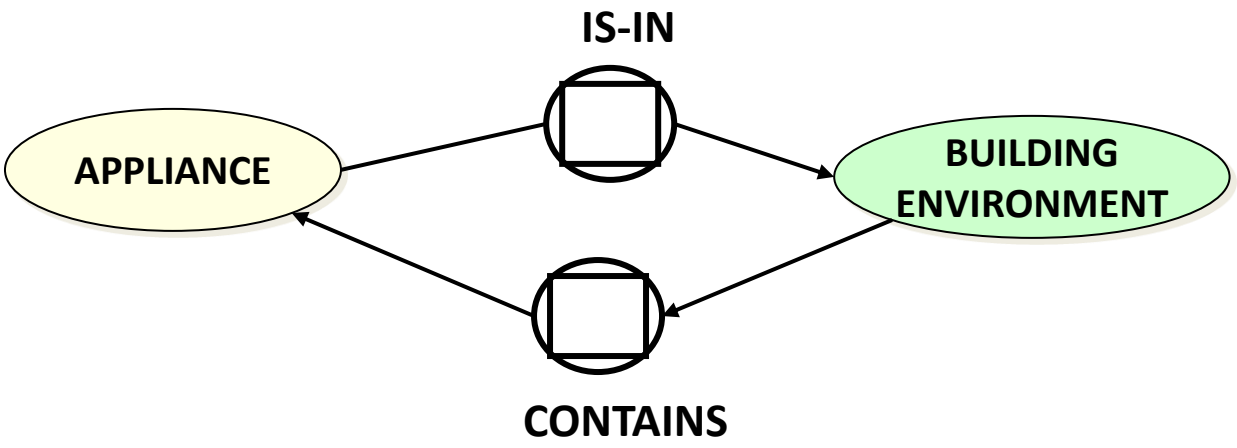
Rollen und Rollenfüller

- Eine Rolle |R|A definiert eine binäre Relation zwischen zwei Konzepten.
- Restriktion für Domäne und Wertebereich möglich, um Verwendung auf „sinnvolle“ Konzepte zu beschränken.
- DL Syntax: ∀IS-IN.BUILDING-ENVIRONMENT
- „Lamp_1 is-in LivingRoom“



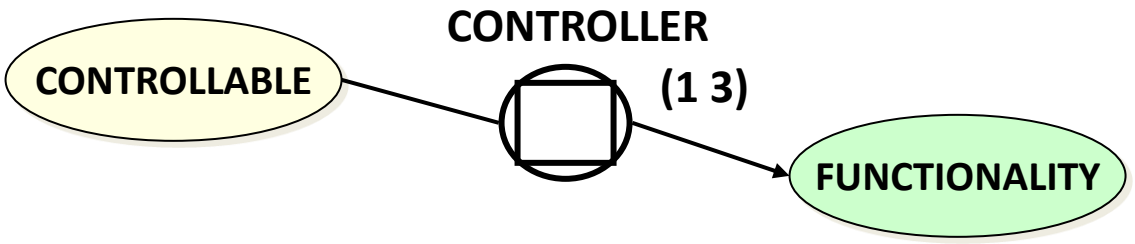
Inverse Rolle

- Eine Rolle |R|A^{-} definiert eine inverse Rolle zu |R|A
- Umgekehrte Leserichtung



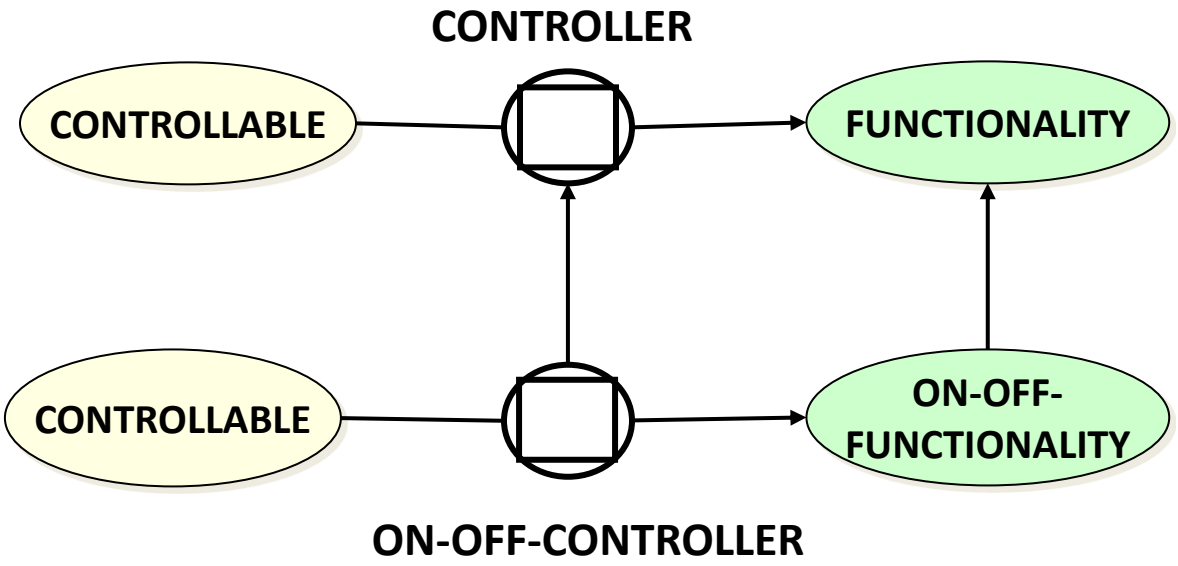
Rollen und Kardinalitäten

- Festlegen der minimalen und maximalen Anzahl von Elementen
- (u o) u:=untere Grenze, o:=obere Grenze, $0 <= n <= \infty$
- (n n) exakt n Elemente
- Bsp: Ein Controllable hat min. 1, max. 3 Funktionen

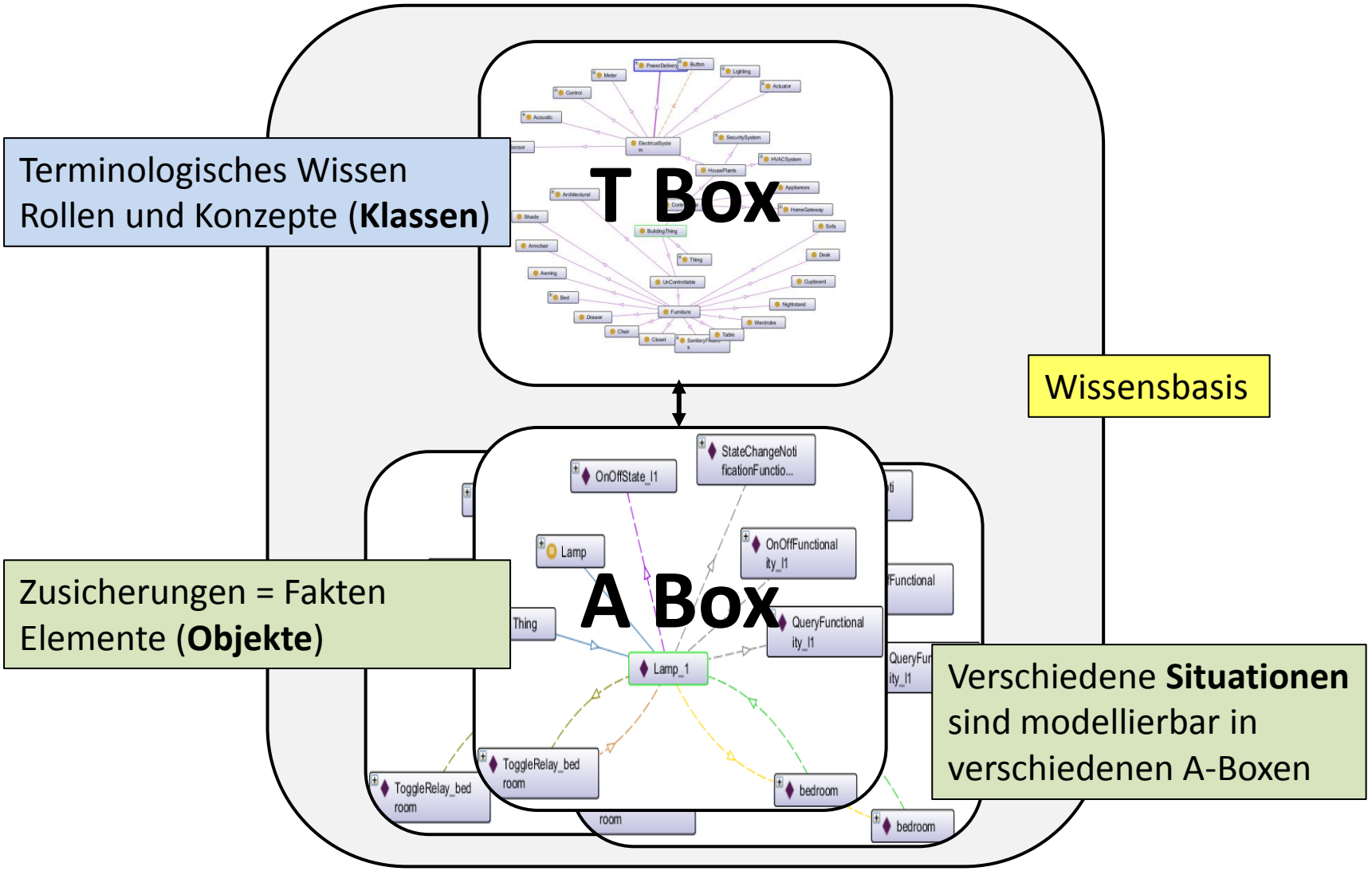


Rollentaxonomie

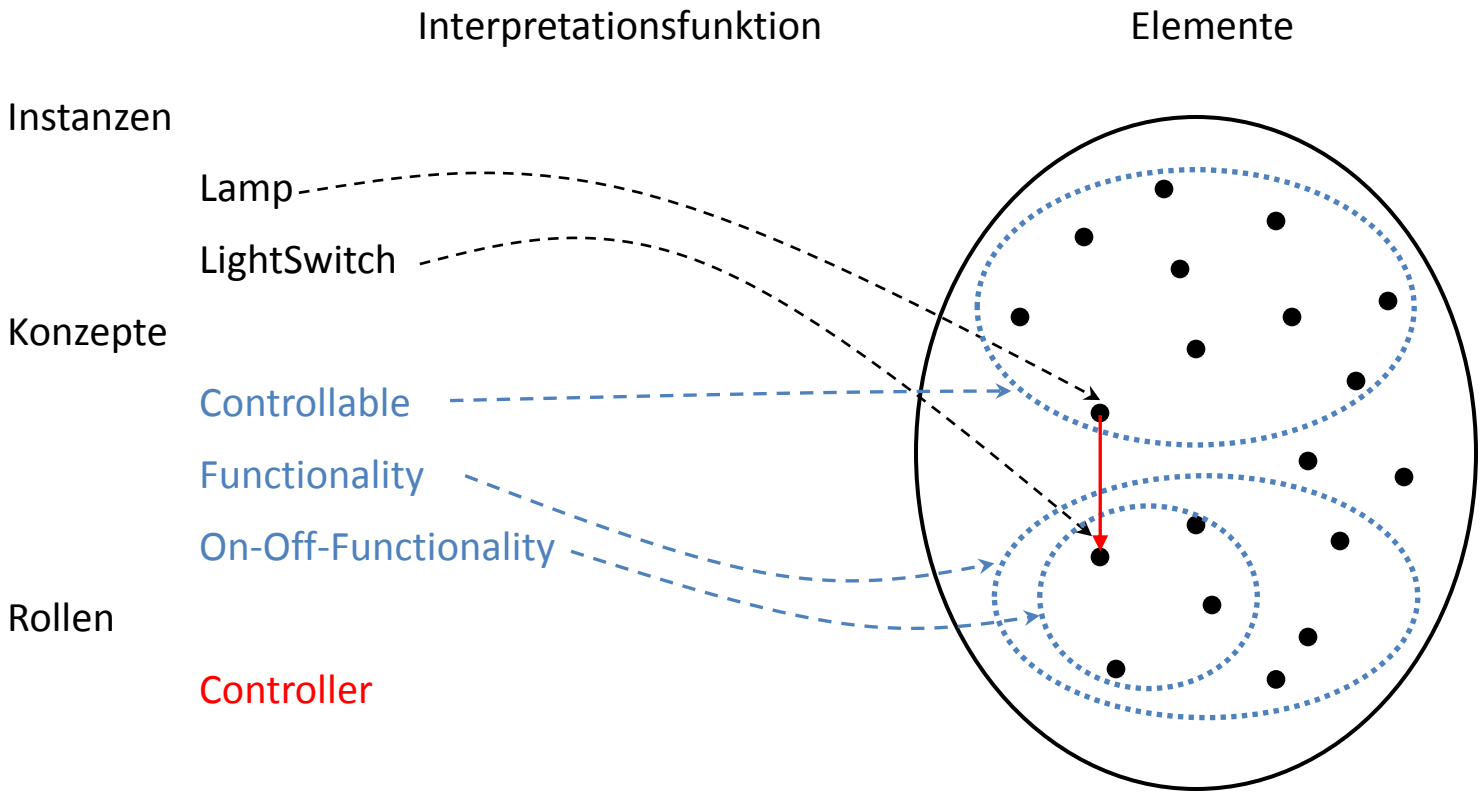
- Ist Rolle |R|B **spezifischer** als |R|A, so sagen wir
  - |R|A **differenziert**|R|B
  - |R|B **generalisiert**|R|A



Wissensrepräsentation in der KI



Modelltheoretische Semantik



Methodologie

- Der Entwurf von Ontologien birgt Fehlerquellen
  - Beim Entwurf von umfangreichen Ontologien können in der Praxis leicht Fehler passieren, vor allem wenn viele Personen daran beteiligt sind.
    - Es kann z.B. vorkommen, dass Konzepte mehrfach unter verschiedenem Namen definiert werden.
    - Es können widersprüchliche Definitionen gemacht werden, die keine Instanzen zulassen.
    - Lösungsansatz: Automatisches Reasoning mit Tools
  - Allgemeine Konzepte (Raum, Zeit etc.) werden in jeder Ontologie immer wieder aufs neue Definiert
    - Lösungsansatz: Modularisierung, Hyperontologien

Inferenz (Reasoning)

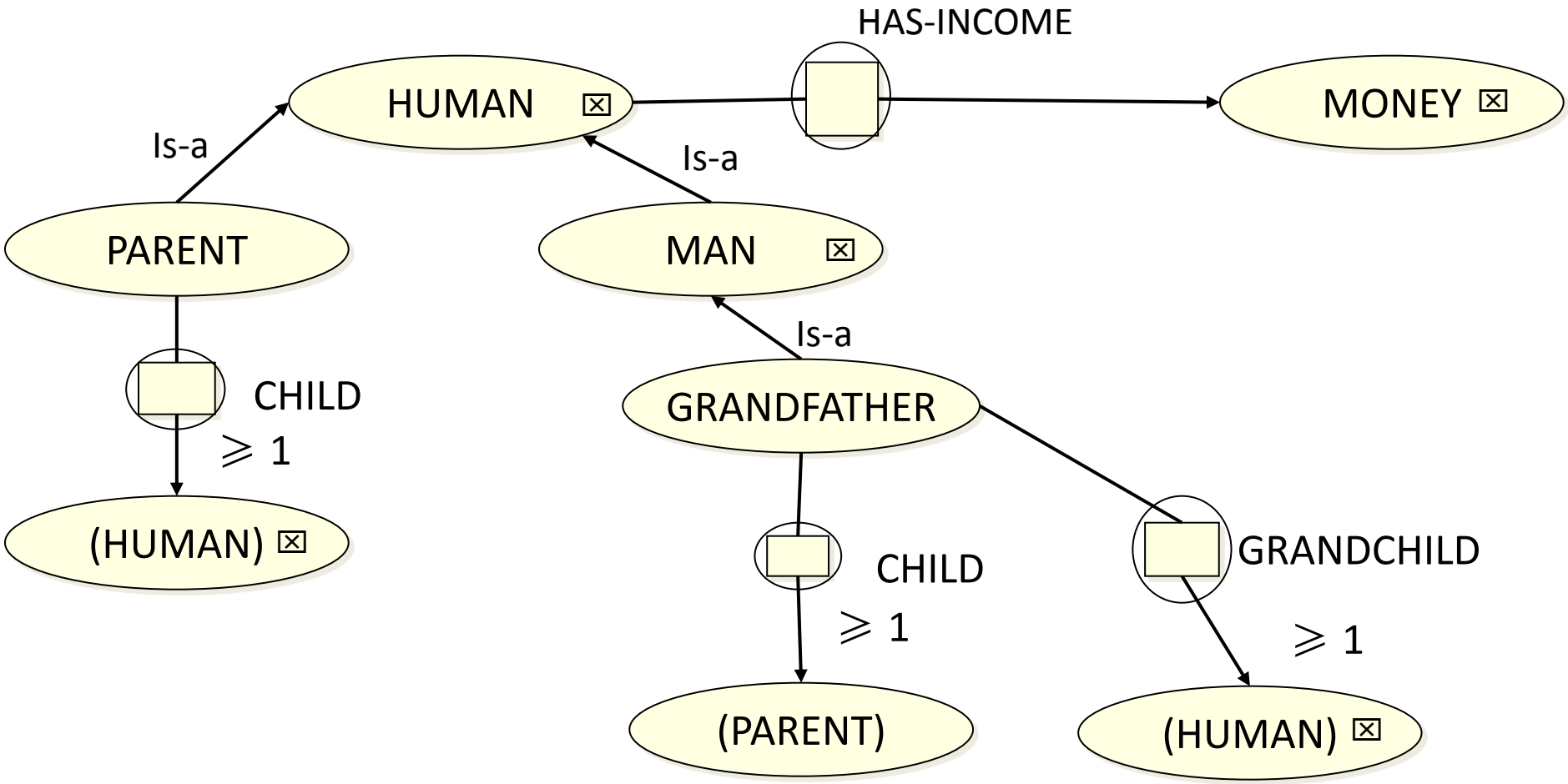
Auf Klassenebene (T-Box):

Welche **Subsumptionsbeziehungen** bestehen zwischen Konzepten?	(Klassifikation)
Welche **Rollenbeziehungen erben** diese Konzepte?	(Vererbung)
Sind die definierten Klassen **konsistent oder widersprüchlich** (leer)?	(Konsistenz)
Sind alle Konzepte **unterschiedlich** oder **gibt es Äquivalenzen**?	(Äquivalenz)
Haben zwei Klassen gemeinsame Objekte oder nicht?	(Disjunktheit)

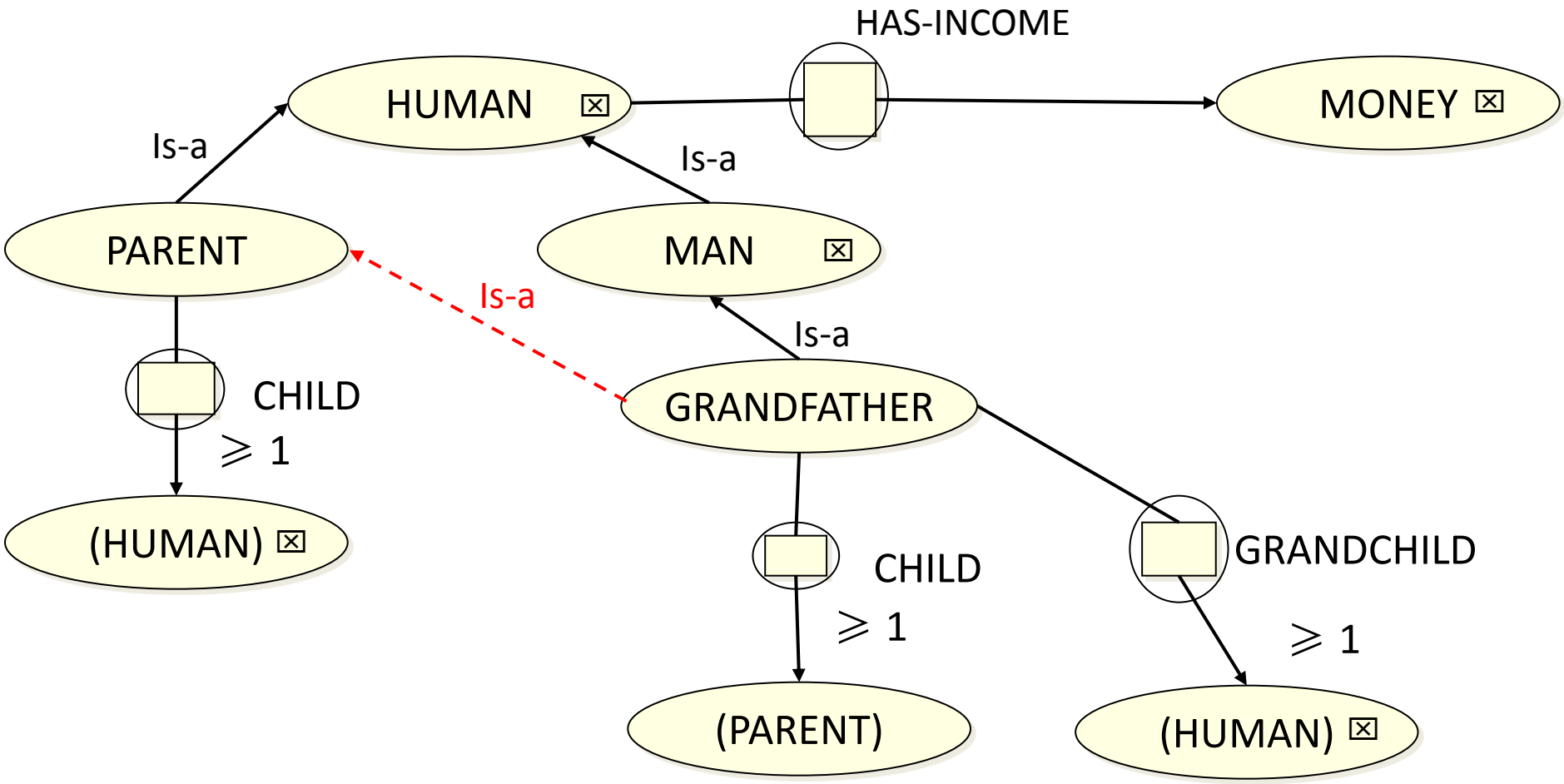
Auf Instanzebene (A-Box):

Welchen **Klassen** gehört ein bestimmtes **Objekt** an?	(Instanziierung)
Welche **Elemente** gehören zu einer **bestimmten Klasse**?	(Retrieval)
Welche **Rollenbeziehungen** bestehen zwischen Objekten?	(Vererbung)
Welche **Elemente** gleichen sich gemäß ihrer Definition?	(Äquivalenz)

Inferenz auf Klassenebene



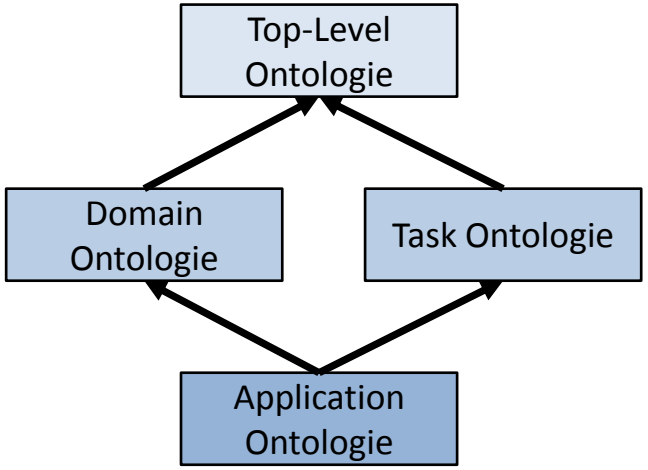
Inferenz auf Klassenebene



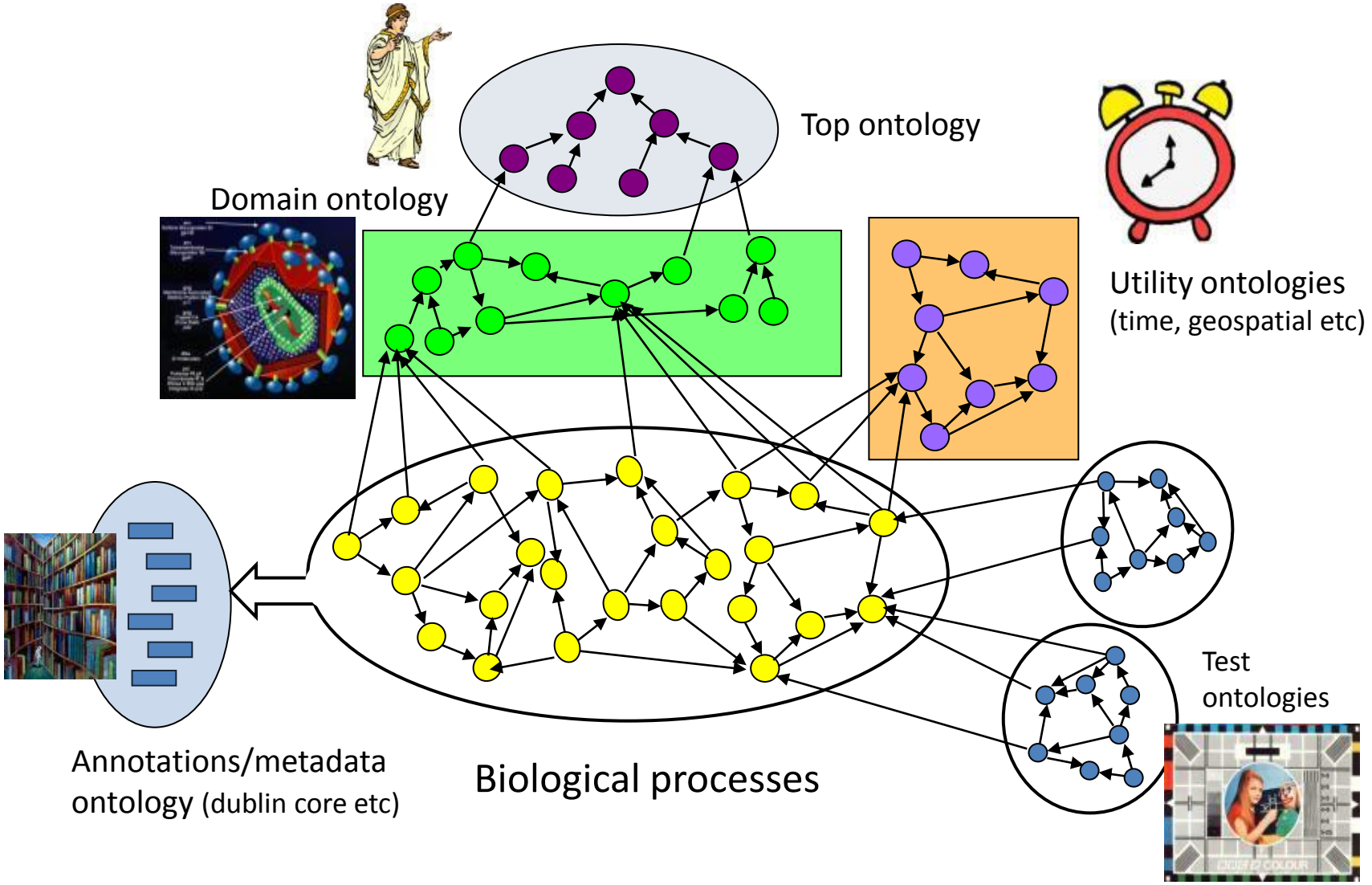
Top-Level und Domain Ontologien

[Guarino 1997] unterscheidet verschiedene Ebenen von Ontologien:

- **Top-Level Ontologien** beschreiben sehr **generelle Konzepte**: *Raum, Zeit, Objekt, Event, Aktion,* .. sowie **Relationen**: *equal, is-part-of, connected-to, dependent-on, caused-by, instance-of, subclass-of*, ..
- Domain und Task Ontologien beschreiben **allgemeines Vokabular** für bestimmte **Anwendungsdomänen** (z.B. Medizin oder AAL), um Interoperabilität zwischen Systemen zu erreichen
- Application Ontologien beschreiben Konzepte, die von einer bestimmten Domäne und Aktivität abhängen bzw. diese spezialisieren.



Hierarchies of Ontologies



Quelle: W. Wahlster

Beispiel: Suggested Upper Merged Ontology (SUMO)

- The Suggested Upper Merged Ontology (SUMO) and its domain ontologies form the **largest formal public ontology** in existence today. They are being used for research and applications in search, linguistics and reasoning.

http://www.ontologyportal.org/

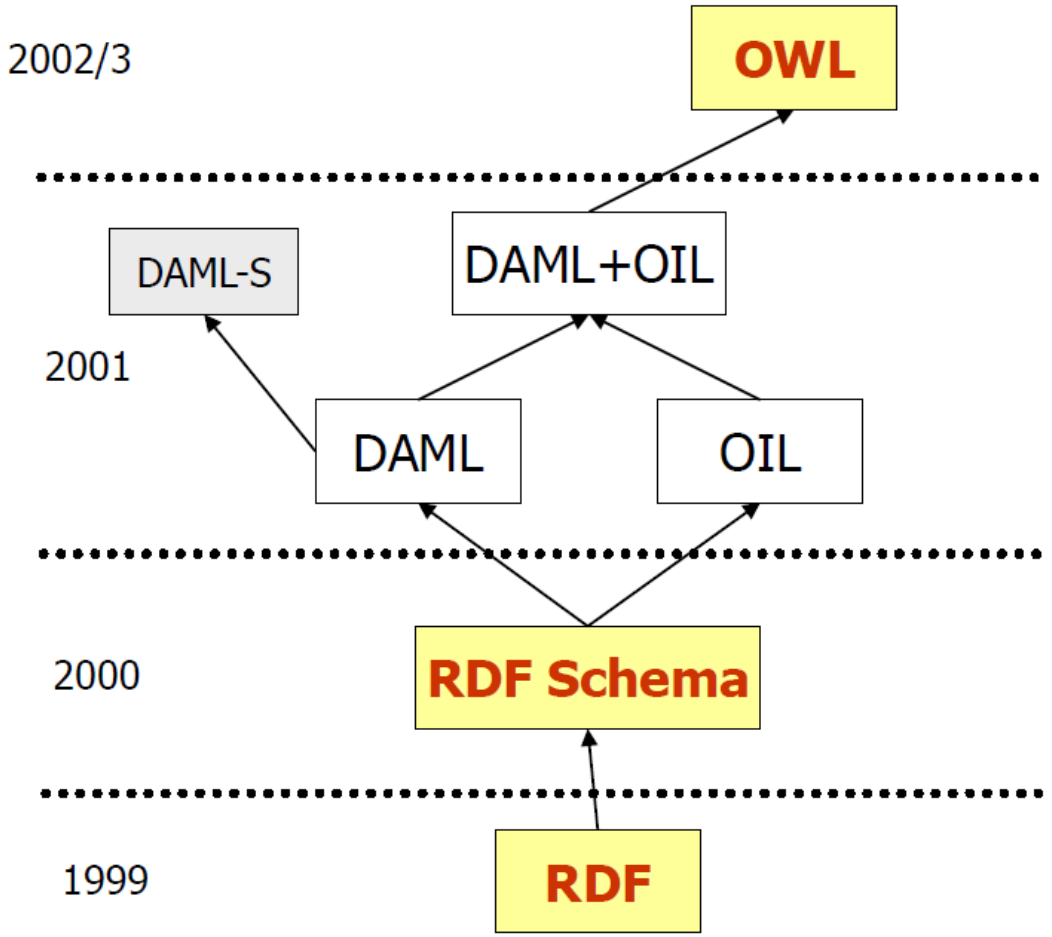
DogOnt Domain Ontologie

- **Domäne**: **Domotik** / **Gebäudeautomatisierung**
  - Beispielsweise Lichtsteuerung mit KNX
- Entworfen um die Fähigkeiten heutiger Systeme abzubilden und die **Interoperabilität mit zukünftigen Lösungen** zu unterstützen
- Gegliedert in fünf Teilbäume
  - Building things (controllable and furniture)
  - Building environment
  - State of controllable devices
  - Functionality of controllable devices
  - Networking
- Publikationen und OWL Dateien zum Download:
  - http://elite.polito.it/dogont-tools-80

Semantisches Web

- *Berners-Lee, 2001*
  - Das Semantische Web ist eine Erweiterung des herkömmlichen Webs, in der Informationen mit eindeutigen Bedeutungen versehen werden, um die Arbeit zwischen Mensch und Maschine zu erleichtern
- Grundlage bildet RDF, um Ressourcen im Web mittels Metadaten in maschinenlesbarer Form semantisch zu annotieren.
- Ontologien können in der Sprache OWL modelliert werden, die syntaktisch auf XML/RDF und semantisch auf Description Logics (DL) basiert.

Sprachen des Semantischen Webs



Das RDF Datenmodell

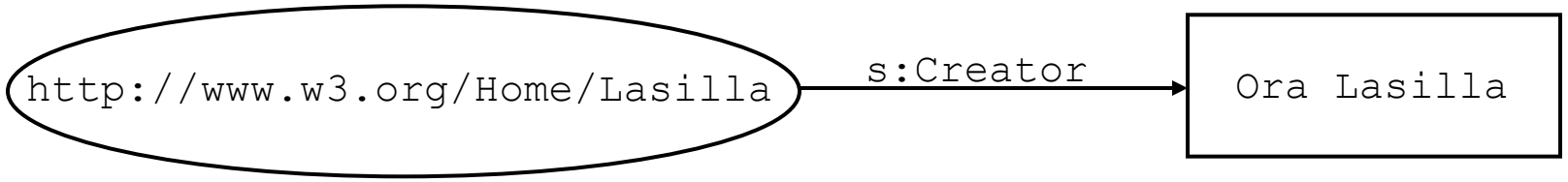
- Ressourcen
  - Eine Ressource ist jedes Objekt im Web, das durch eine URI **eindeutig** identifiziert werden kann
    - z.B. *http://www.dfki.de/my-rdf.html#resource1*
- Properties
  - Relationen zu anderen Ressourcen oder Literalen
- Statements
  - **Tripel** der Form (Subjekt, Prädikat, Objekt)

Ein einfaches RDF Beispiel

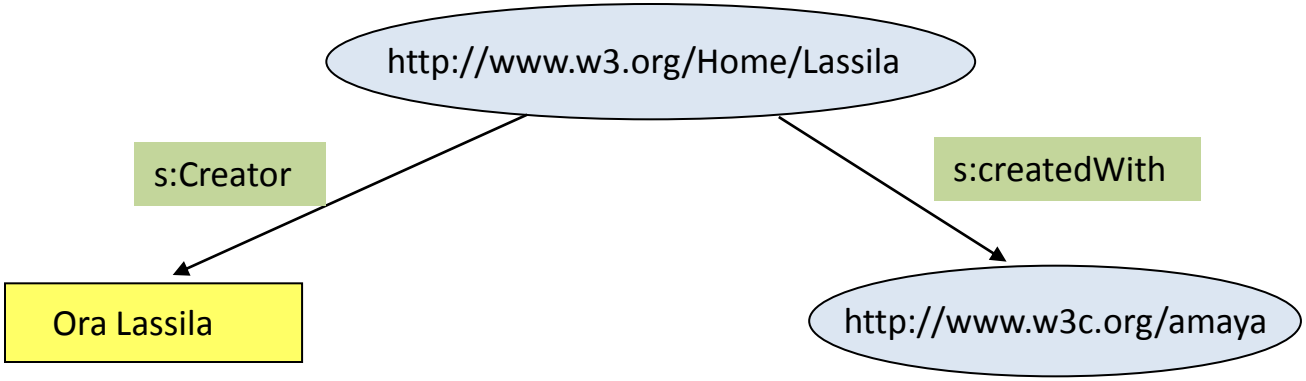
- **Statement**
  - „`http://www.w3.org/Home/Lassila`
    has the creator Ora Lassila“

- **Struktur**
  - Resource (Subject) `http://www.w3.org/Home/Lassila`
  - Property (Predicate) `Creator`
  - Value (Object) `“Ora Lassila“`

- **Darstellung als Graph**



RDF Syntax in XML




```
<rdf:rdf>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
     <s:Creator>Ora Lassila</s:Creator>
     <s:createdWith rdf:resource="http://www.w3c.org/amaya"/>
  </rdf:Description>
</rdf:rdf>
```

The Web Ontology Language
WOL ≡ OWL

- Entwickelt von der W3C Web-Ontology Working Group 2001-2004
- Weiterentwicklung der Sprachen DAML, OIL, RDF-S
- Basiert auf Description Logics zur Beschreibung von Konzepten durch Axiome
- Je ausdrucksmächtiger die Logik, desto schwieriger das automatische Schließen (Inferenz / Reasoning)
- Daher drei Varianten:
  - OWL-LITE, **OWL-DL** (Entscheidbar)
  - OWL-FULL erlaubt vollständige Axiomatisierung

OWL-DL/LITE Fähigkeiten

- Klassen
  - Mehrfachvererbung, Äquivalenz, Schnitt, Vereinigung*, Komplement*, Disjunktheit*
- Instanzen
- Attribute
  - Datentypen
- Relationen
  - Vererbung
  - Spezifikation von Definitionsbereich und Wertebereich, Kardinalitäten, Funktionen, Wert*
- Open World Assumption
  - Wahr ist, was nicht explizit verneint/ausgeschlossen ist

*nicht in OWL-LITE

OWL-DL und Description Logic

OWL Abstract Syntax	DL Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$
complementOf	$\neg C$
oneOf	$\{a_1\} \sqcup \ldots \sqcup \{a_n\}$
allValuesFrom	$\forall P.C$
someValuesFrom	$\exists P.C$
maxCardinality	$\leqslant n P$
subClassOf	$C_1 \sqsubseteq C_2$
equivalentClass	$C_1 \equiv C_2$
disjointWith	$C_1 \sqsubseteq \neg C_2$
sameIndividualAs	$\{a_1\} \equiv \{a_2\}$

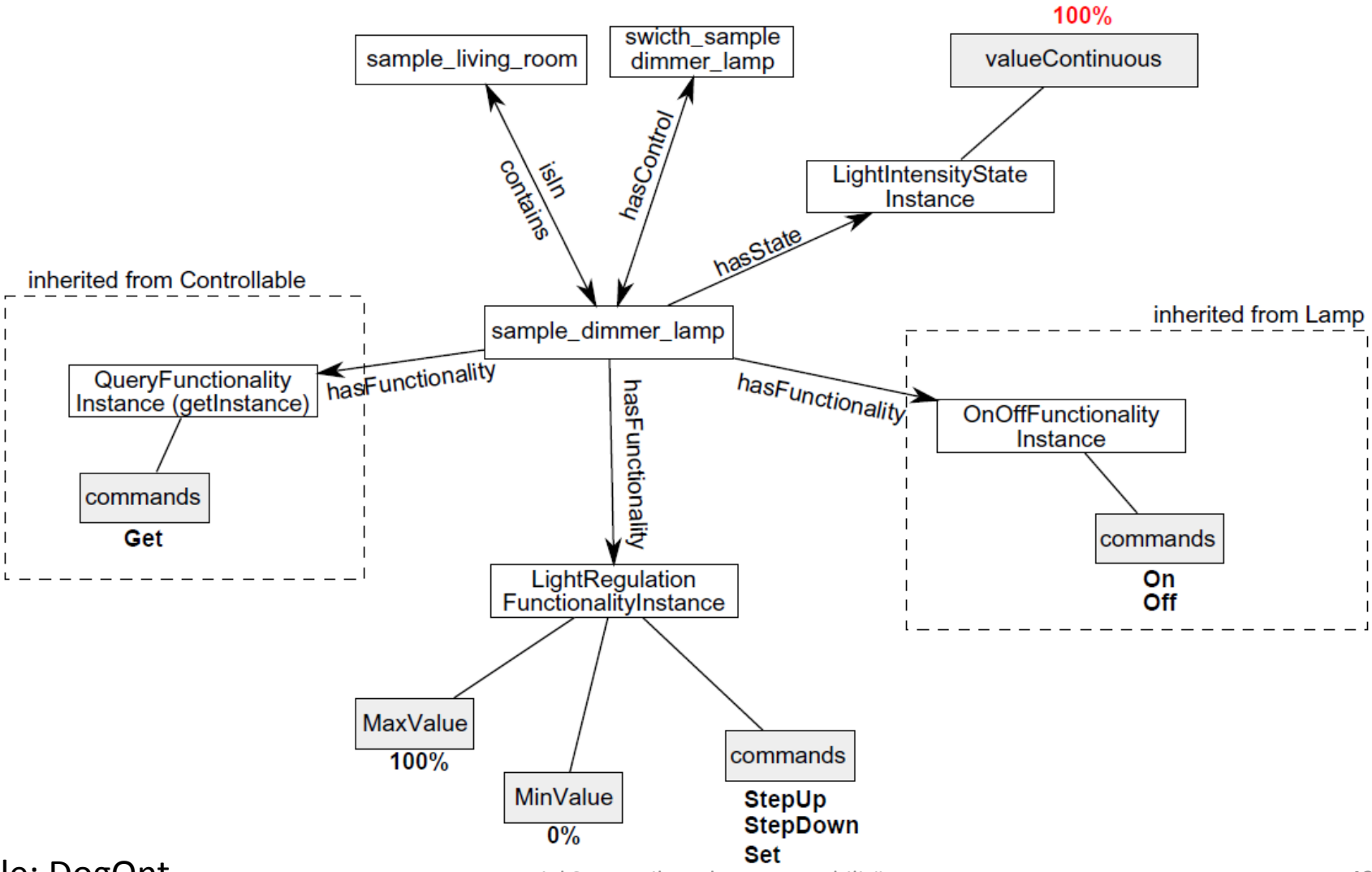
Klassen in OWL Syntax

- Klasse ***Lamp*** in DogOnt.owl
- <**owl:Class** rdf:about="http://elite.polito.it/ontologies/**dogont.owl#Lamp**">
  <**rdfs:label** rdf:datatype="&xsd;string">Lamp</rdfs:label>
  <**rdfs:subClassOf** rdf:resource="http://elite.polito.it/ontologies/dogont.owl#Lighting"/>
  <**rdfs:subClassOf**>
  <**owl:Restriction**>
  <**owl:onProperty** rdf:resource="http://elite.polito.it/ontologies/dogont.owl#hasState"/>
  <**owl:someValuesFrom** rdf:resource="http://elite.polito.it/ontologies/dogont.owl#OnOffState"/>
  </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
  <owl:Restriction>
  <**owl:onProperty** rdf:resource="http://elite.polito.it/ontologies/dogont.owl#hasFunctionality"/>
  <**owl:someValuesFrom** rdf:resource="dogont.owl#OnOffFunctionality"/>
  </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="&xsd;string">An artificial source of visible illumination</rdfs:comment>
  </owl:Class>

Objekte in RDF Syntax

- Element ***Lamp_1*** in simpleHome.owl
- <**owl:Thing** rdf:about="http://elite.polito.it/ontologies/**simpleHome.owl#Lamp_1**">
  <**rdf:type** rdf:resource="&dogont;Lamp"/>
  <**rdf:type** rdf:resource="&owl;NamedIndividual"/>
  <**dogont:hasFunctionality** rdf:resource="http://[...]/simpleHome.owl#OnOffFunctionality_l1"/>
  <**dogont:hasState** rdf:resource="http://elite.polito.it/ontologies/simpleHome.owl#OnOffState_l1"/>
  <**dogont:hasFunctionality** rdf:resource="http://elite[...]/simpleHome.owl#QueryFunctionality_l1"/>
  <**dogont:hasFunctionality** rdf:resource="http://[...]/#StateChangeNotificationFunctionality_l1"/>
  <**dogont:hasControl** rdf:resource="http://elite.polito.it/ontologies/simpleHome.owl#ToggleRelay_bedroom"/>
  <**dogont:isIn** rdf:resource="http://elite.polito.it/ontologies/simpleHome.owl#bedroom"/>
  </owl:Thing>

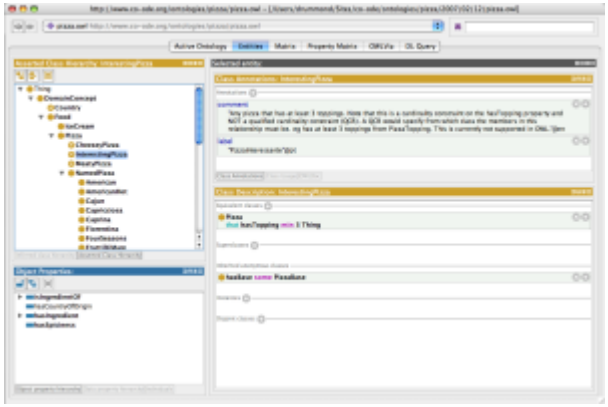
Modellierung einer Lampe in DogOnt mittels OWL und Protege



Quelle: DogOnt

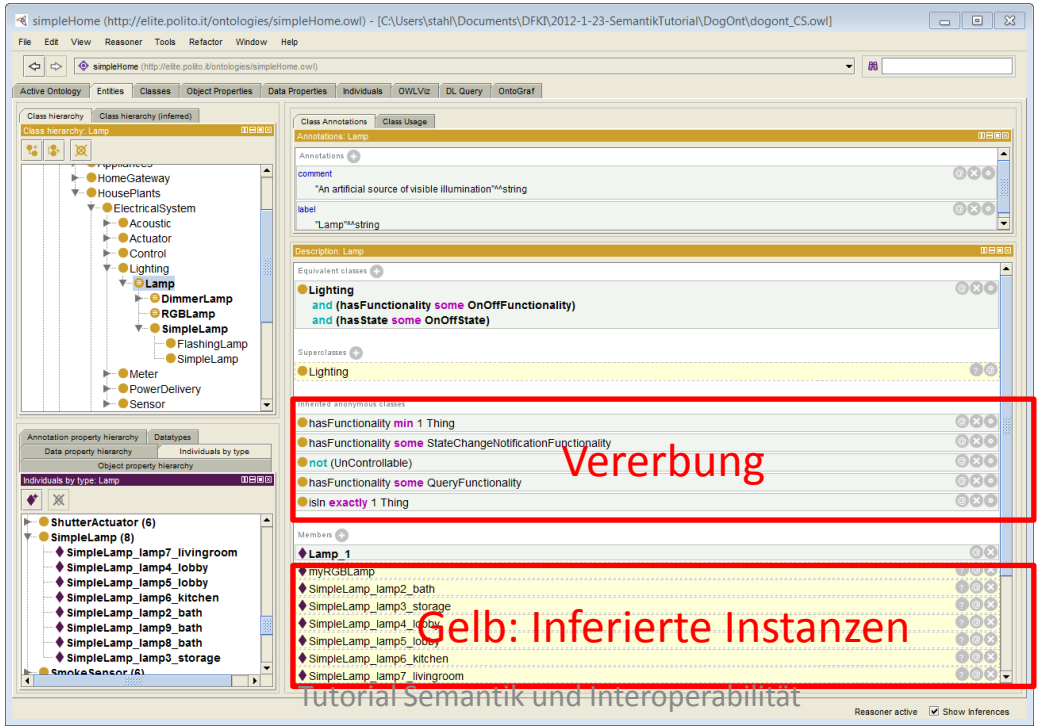
Protégé

- **Open-Source Editor** für Ontologien mit Unterstützung für **RDF und OWL**
- Entwickelt in Stanford
  - http://protege.stanford.edu/
- Versionsunterschiede
  - Protégé 3.x: OWL 1.0, OWL Full, RDF(S), **SPARQL** Query Language unterstützt
  - Protégé 4: OWL 2.0, nur LITE/DL, **FaCT++ Reasoner** direkt integriert
- Tutorial der Universität Manchester zum Umgang mit OWL in Protégé
  - http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_1.pdf



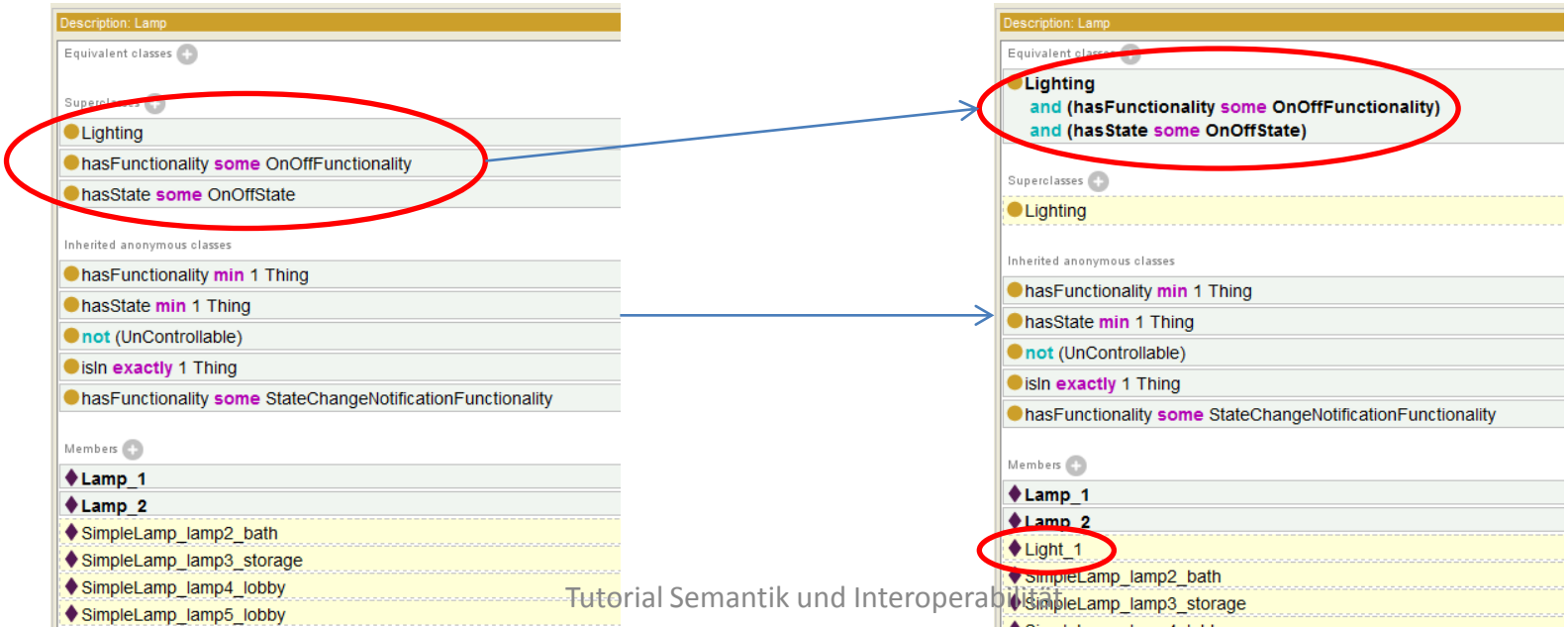
Vererbung und Instantiierung

- Klasse *SimpleLamp* ist Spezialisierung von Klasse *Lamp*
- *SimpleLamp_lamp1* etc.. sind aber noch nicht als Instanzen von *Lamp* zu sehen
- Lösung: Wir starten den **Reasoner** Hermit (Reasoner->Start Reasoner). Dieser erkennt nun die Zugehörigkeit der SimpleLamps zur Oberklasse Lamp.



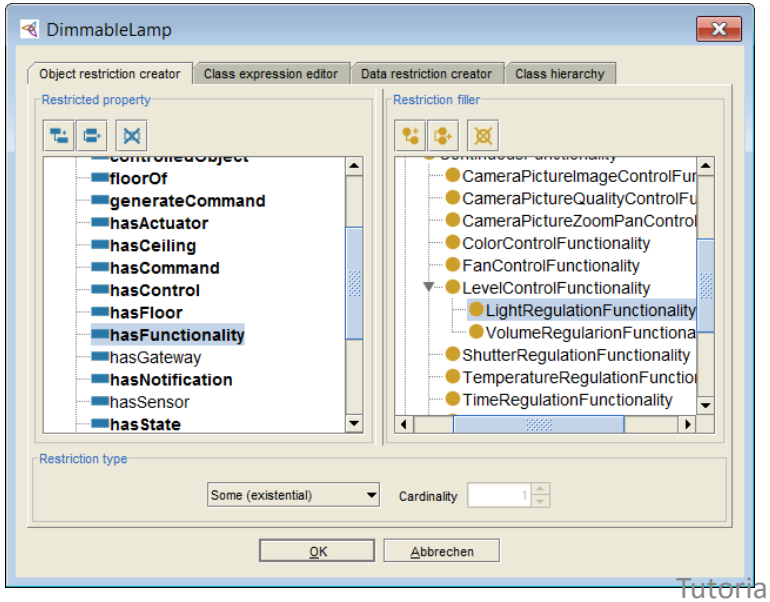
Definierte Klassen

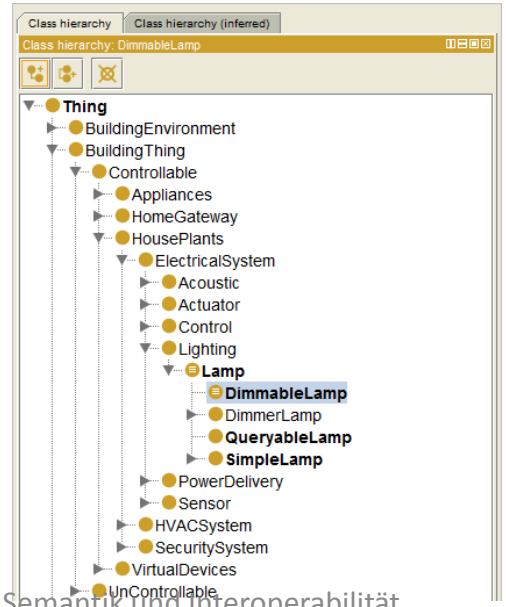
- Wir machen ein neues *Lighting*-Objekt mit *OnOffFunctionality* und *OnOffState*
- Light_1 hat nun **notwendige** Eigenschaften einer Lampe. Dennoch erkennt der Reasoner Light_1 nicht als Lampe an, weil Lamp keine **hinreichenden** Axiome hat.
- Lösung: wir markieren die drei Axiome bei *„Superclasses“* und wählen im Kontextmenü *„convert selected rows to defined class“.*
- Axiome erscheinen nun bei *„Equivalent classes“* und Icon ändert sich
- Wir synchronisieren den Reasoner, dieser klassifiziert nun Light_1 als Lamp.

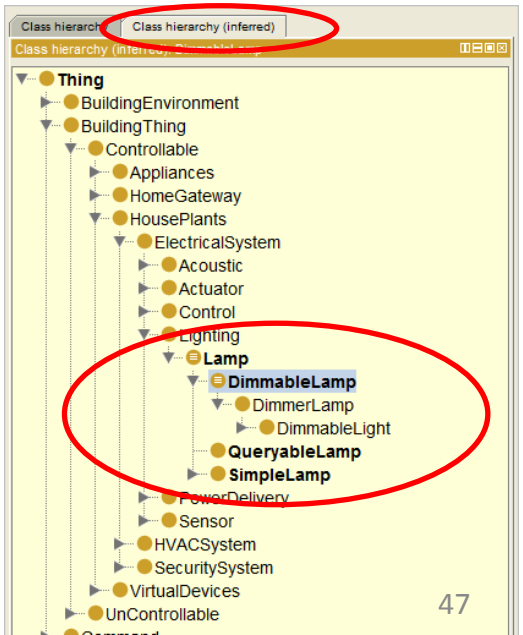


Klassifikation

- Wir machen eine neue Unterklasse von Lamp namens DimmableLamp mit hasFunctionality LightRegulationFunctionality
- Damit ist DimmableLamp spezieller als Lamp, aber allgemeiner als DimmerLamp
- Der Reasoner stellt die korrekte Subsumption in der inferierten Klassenhierarchie an:







Referenzen

- [Gruber 1993] – T. Gruber, „A Translation Approach to Portable Ontology Specifications.” Knowledge Acquisition, 5(2):199-220, 1993. http://tomgruber.org/writing/ontolingua-kaj-1993.htm
- [Uschold and Gruninger 1996] – M. Uschold and M. Gruninger, “Ontologies: Principles, Methods and Applications”, AIAI TR 191, February 1996. http://www.aiai.ed.ac.uk/oplan/documents/1996/96-ker-intro-ontologies.pdf
- [Guarino 1997]. Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. In M. T. Pazienza (ed.) *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology. Springer Verlag: 139-170.*
- [Guarino 1998] – Formal Ontology in Information Systems, Proceedings of the International Conference on Formal Ontology and Information Systems (FOIS), IOS Press, 1998. http://www.loa.istc.cnr.it/Papers/FOIS98.pdf