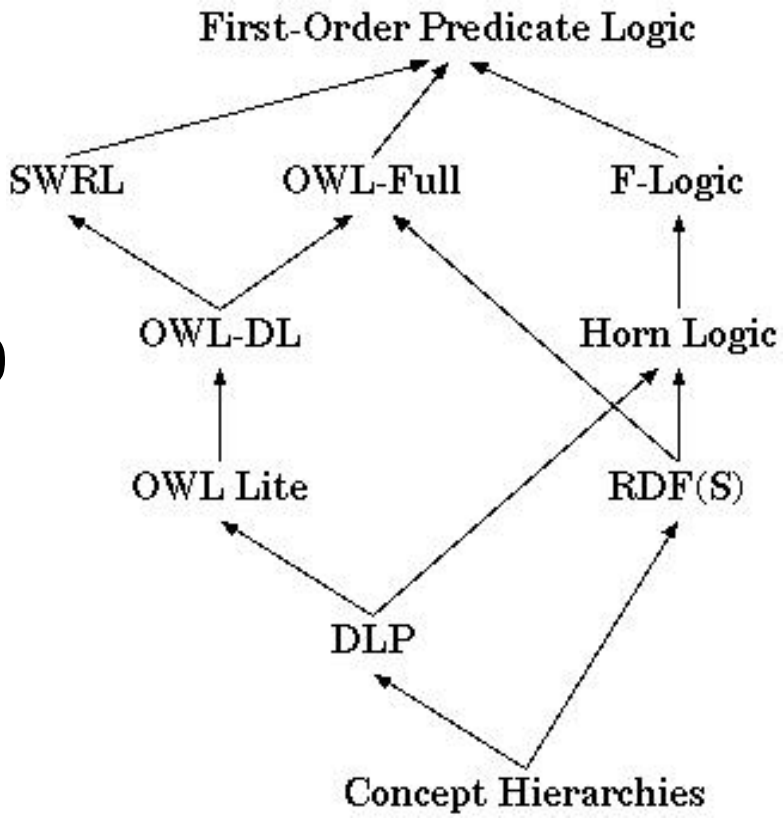
Semantic Web

OWL

Acknowledgements to Pascal Hitzler, York Sure

OWL – General



- W3C Recommendation since 200
- Semantic fragment of FOL
- Three variants:
  OWL Lite $\subseteq$ OWL DL $\subseteq$ OWL Full
- No reification in OWL DL
  RDFS is fragment of OWL Full
- OWL DL is decidable
- OWL DL = SHOIN(D) (description logics)
- W3C-Documents contain many more details that we cannot talk about here

Content

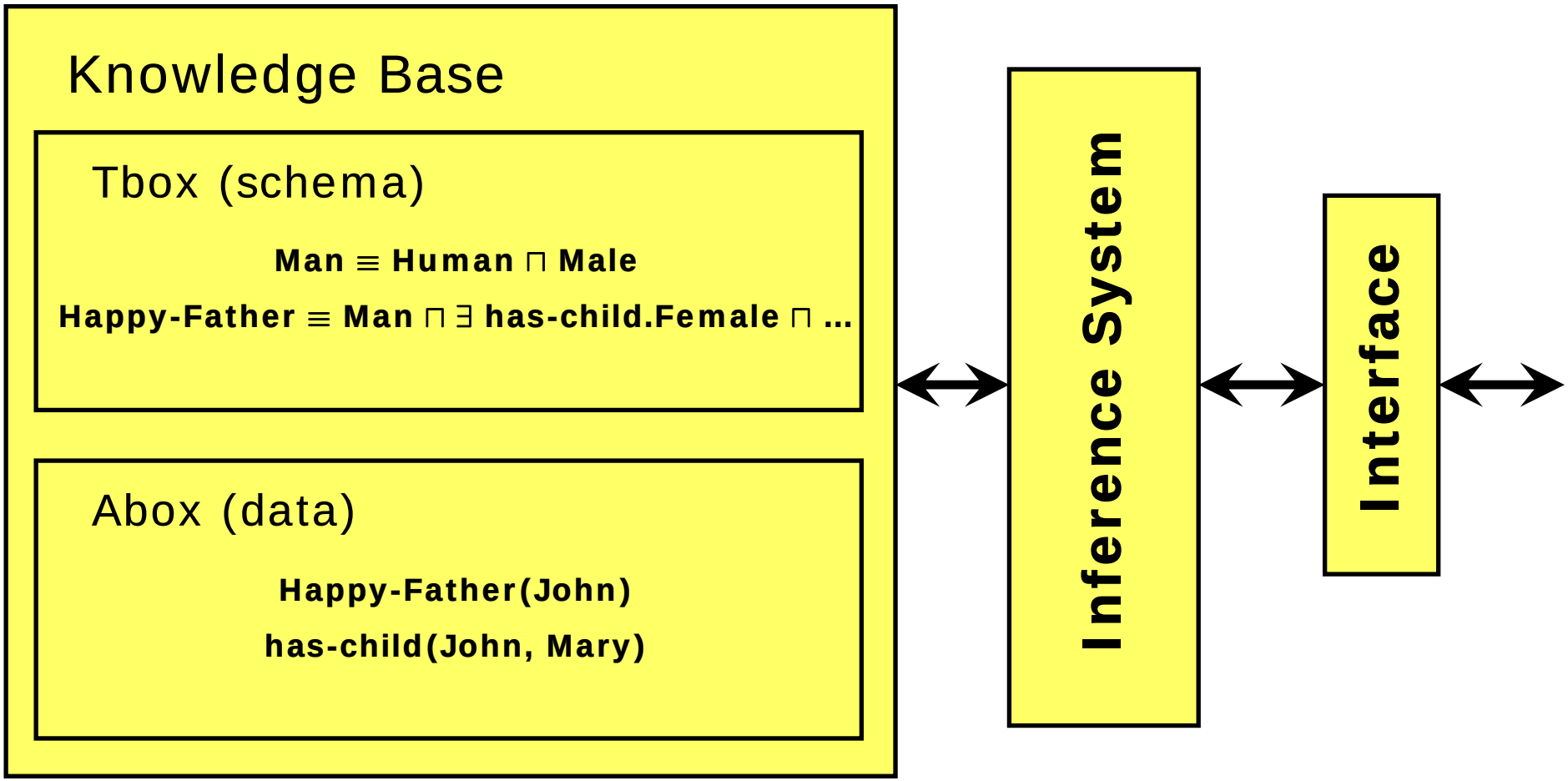
OWL – **Syntax and model theoretic semantics**

a. Description logics: SHOIN(D)
b. **OWL as SHOIN(D)**
c. **Serializations**
d. **Knowledge modelling in OWL**

Description Logics (Terminological Logics, DLs)

- Fragments of FOL
- Most often decidable
- Moderately expressive
- Stem from semantic networks
- Close relationship to propositional modal logics
- W3C Standard OWL DL corresponds to SHOIN(D)

General DL Architecture



DLs – general structure

- DLs are a Family of logic-based formalism for knowledge representation
- Special language characterized by:
  - Constructors to define complex concepts and roles based on simpler ones.
  - Set of axiom to express facts using concepts, roles and individuals.

- ALC is the smallest DL, which is propositionally closed:
  - $\wedge$, $\vee$, $\neg$ are constructors, noted by $\sqcap$, $\sqcup$, $\neg$.
  - Quantors define how roles are to be interpreted:

$$\text{Man} \sqcap \exists \text{hasChild.Female} \sqcap \exists \text{hasChild.Male} \sqcap \forall \text{hasChild.}(\text{Rich} \sqcup \text{Happy})$$

Further DL concepts and role constructors

- Number restrictions (cardinality constraints) for roles:
  $\geq$3 hasChild, $\leq$1hasMother
- Qualified number restrictions:
  $\geq$2 hasChild.Female, $\leq$1 hasParent.Male
- Nominals (definition by extension):
  {Italy, France, Spain}
- Concrete domains (datatypes): hasAge.($\geq$21)
- Inverse roles: $\text{hasChild}^{-} \equiv \text{hasParent}$
- Transitive roles: hasAncestor* (descendant)
- Role composition: hasParent.hasBrother (uncle)

DL Knowledge Bases

- DL Knowledge Bases consist of two parts (in general):
  - TBox: Axioms, describing the structure of a modelled domain (conceptual schema):
    - HappyFather $\equiv$ Man $\sqcap \exists$hasChild.Female $\sqcap$ …
    - Elephant $\sqsubseteq$ Animal $\sqcap$ Large $\sqcap$ Grey
    - transitive(hasAncestor)
  - Abox: Axiome describing concrete situations (data, facts):
    - HappyFather(John)
    - hasChild(John, Mary)
- The distinction between TBox/ABox does not have a deep logical distinction
  … but it is common useful modelling practice.

Syntax for DLs (ohne concrete domains)

	Concepts	
ALC	Atomic	A, B
	Not	$\neg C$
	And	$C \sqcap D$
	Or	$C \sqcup D$
	Exists	$\exists R.C$
	For all	$\forall R.C$
Q (N)	At least	$\geq n\ R.C$ $(\geq n\ R)$
	At most	$\leq n\ R.C$ $(\leq n\ R)$
O	Nominal	$\{i_1, \ldots, i_n\}$

	Roles	
I	Atomic	R
	Inverse	R^-

Ontology (=Knowledge Base)

	Concept Axioms (TBox)	
	Subclass	$C \sqsubseteq D$
	Equivalent	$C \equiv D$

	Role Axioms (RBox)	
H	Subrole	$R \sqsubseteq S$
S	Transitivity	Trans(S)

	Assertional Axioms (ABox)	
	Instance	C(a)
	Role	R(a,b)
	Same	$a = b$
	Different	$a \neq b$

S = ALC + Transitivity

OWL DL = SHOIN(D) (D: concrete domain)

Content

OWL – **Syntax and model theoretic semantics**

a. **Description logics: SHOIN(D)**
b. **OWL as SHOIN(D)**
c. **Serializations**
d. **Knowledge modelling in OWL**

OWL DL as DL: Class constructors

Constructor	DL Syntax	Example	FOL Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human ⊓ Male	$C_1(x) \wedge \ldots \wedge C_n(x)$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor ⊔ Lawyer	$C_1(x) \vee \ldots \vee C_n(x)$
complementOf	$\neg C$	¬Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} ⊔ {mary}	$x = x_1 \vee \ldots \vee x = x_n$
allValuesFrom	$\forall P.C$	∀hasChild.Doctor	$\forall y.P(x,y) \rightarrow C(y)$
someValuesFrom	$\exists P.C$	∃hasChild.Lawyer	$\exists y.P(x,y) \wedge C(y)$
maxCardinality	$\leqslant nP$	⩽1hasChild	$\exists^{\leqslant n} y.P(x,y)$
minCardinality	$\geqslant nP$	⩾2hasChild	$\exists^{\geqslant n} y.P(x,y)$

Nesting of expression is allowed at arbitrary depth:

Person ⊓ ∀hasChild.(Doctor ⊔ ∃hasChild.Doctor)

OWL DL as DL: Axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg${peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+ \sqsubseteq$ ancestor
functionalProperty	$\top \sqsubseteq \leqslant 1P$	$\top \sqsubseteq \leqslant$1hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leqslant 1P^-$	$\top \sqsubseteq \leqslant$1hasSSN$^-$

- General Class Inclusion ($\sqsubseteq$):
  C $\equiv$ D gdw (C $\sqsubseteq$ D und D $\sqsubseteq$ C)
- Obvious equivalances with FOL:
  C $\equiv$ D $\Leftrightarrow$ ($\forall$x) (C(x) $\leftrightarrow$ D(x))
  C $\sqsubseteq$ D $\Leftrightarrow$ ($\forall$x) (C(x) $\rightarrow$ D(x))

OWL/DL Example

Terminological Knowledge (*TBox*):

$\text{Human} \sqsubseteq \exists \text{parentOf.Human}$

$\text{Orphan} \equiv \text{Human} \sqcap \neg \exists \text{childOf.Alive}$

Knowledge about Individuals (*ABox*):

Orphan(harrypotter)

ParentOf(jamespotter,harrypotter)

Semantics and logical consequences may be derived by translation to FOL

Model theoretical Semantics – direct

Concept expressions	
A	Subset of Δ^I
$\neg C$	$\Delta^I \setminus C^I$
$C \sqcap D$	$\{x \mid x \in C^I \text{ and } x \in D^I\}$
$C \sqcup D$	$\{x \mid x \in C^I \text{ or } x \in D^I\}$
$\exists R.C$	$\{x \mid (x,y) \in R^I \text{ and } y \in C^I\}$
$\forall R.C$	$\{x \mid \text{if } (x,y) \in R^I \text{ then } y \in C^I\}$
$\geq n\, R.C$	$\{x \mid \#\{(x,y) \in R^I \text{ and } y \in C^I\} \geq n\}$
$\leq n\, R.C$	$\{x \mid \#\{(x,y) \in R^I \text{ and } y \in C^I\} \leq n\}$
$\{i_1, \ldots, i_n\}$	$\{i_1^I, \ldots, i_n^I\}$

Role expressions	
R	Subset of $\Delta \times \Delta$
R^-	$\{(y,x) \mid (x,y) \in R^I\}$

Ontology (=Knowledge Base)

Concept Axioms (TBox)	
$C \sqsubseteq D$	$C^I \subseteq D^I$
$C \equiv D$	$C^I \equiv D^I$

Role Axioms (rarely: RBox)	
$R \sqsubseteq S$	$R^I \subseteq S^I$

Assertional Axioms (ABox)	
C(a)	$a^I \in C^I$
R(a,b)	$(a^I, b^I) \in R^I$
a = b	$a^I = b^I$
$a \neq b$	$a^I \neq b^I$

Concrete Domains

- Strings and Integers (required by W3C OWL rec)
- Further datatypes may be supported.
- Restricted to decidable predicates over the concrete domain

- Each concrete domain must be implemented separately and then included into the reasoner (weak analogy: built-ins – but no procedural semantics!)

OWL Lite

- Simple fragment, comparatively low complexity
- Some constructors may not be used:
  - owl:unionOf
  - owl:complementOf
  - owl:oneOf
  - owl:hasValue
  - owl:disjointWith
- The applicability of some operators is restricted:
  - owl:intersectionOf
  - owl:minCardinality
  - owl:maxCardinality
  - owl:cardinality

OWL Full

- equals OWL DL union RDFS
- RDF is contained in OWL Full, not in OWL DL

- Intuition:
  - OWL Full allows reification.
  - OWL Full is no “nice” fragment of FOL
  - OWL Full ist not decidable

Reification – Example

Sometimes one might state a proposition about a class name:

Class: father.

(concrete) Rolls: germanClassName(father,“Vater“)
frenchClassName(father, „père“)
englishClassName(father, „father“)

→ Cannot be expressed in OWL DL!

Complexity (worst-case)

OWL variant	Datenkomplexität	Combined complexity
OWL Full	undecidable	undecidable
OWL DL	not known	NExptime
OWL DL without nominals	NP (neues Resultat IJCAI 2005!)	Exptime
OWL Lite	NP	Exptime

Data complexity: assume fixed T-Box, complexit wrt size of ABox

Combined complexity: wrt combined size of ABox and TBox

Content

OWL – **Syntax and model theoretic semantics**

a. **Description logics: SHOIN(D)**
b. **OWL as SHOIN(D)**
c. **Serializations**
d. **Knowledge modelling in OWL**

Serializations/different Syntaxex

- OWL RDF Syntax — W3C recommendation
- OWL Abstract Syntax — W3C recommendation
  See next section

- OWL XML Syntax — W3C document

- DL Schreibweise — widely used in scientific contexts
- FOL Schreibweise — uncommon

- For the implementation and the testing of KAON2 a functional syntax has been developed

Lisp-like

Example: RDF Syntax

Person ⊓ ∀hasChild.(Doctor ⊔ ∃hasChild.Doctor):

```
<owl:Class>
  <owl:intersectionOf rdf:parseType="collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild"/>
      <owl:allValuesFrom>
        <owl:unionOf rdf:parseType="collection">
          <owl:Class rdf:about="#Doctor"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasChild"/>
            <owl:someValuesFrom rdf:resource="#Doctor"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:allValuesFrom>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Content

OWL – **Syntax and model theoretic semantics**

a. **Description logics: SHOIN(D)**
b. **OWL as SHOIN(D)**
c. **Serializations**
d. **Knowledge modelling in OWL**

Knowledge modelling in OWL

Example ontology and conclusion from http://owl.man.ac.uk/2003/why/latest/#2

- Also an example for OWL Abstract Syntax.

```
Namespace(a = <http://cohse.semanticweb.org/ontologies/people#>)
Ontology(
    ObjectProperty(a:drives)
    ObjectProperty(a:eaten_by)
    ObjectProperty(a:eats inverseOf(a:eaten_by) domain(a:animal))
    …
    Class(a:adult partial annotation(rdfs:comment "Things that are adult.")
    Class(a:animal partial restriction(a:eats someValuesFrom (owl:Thing)))
    Class(a:animal_lover complete intersectionOf(restriction(a:has_pet
          minCardinality(3)) a:person))
…)
```

Knowledge modelling: examples

Class(a:bus_driver complete intersectionOf(a:person
restriction(a:drives someValuesFrom (a:bus))))

$$\textbf{bus\_driver} \equiv \textbf{person} \sqcap \exists \textbf{drives.bus}$$

Class(a:driver complete intersectionOf(a:person
restriction(a:drives someValuesFrom (a:vehicle))))

$$\textbf{driver} \equiv \textbf{person} \sqcap \exists \textbf{drives.vehicle}$$

Class(a:bus partial a:vehicle)

$$\textbf{bus} \sqsubseteq \textbf{vehicle}$$

- A bus driver is a person that drives a bus.
- A bus is a vehicle.
- A bus driver drives a vehicle, so must be a driver.

The subclass is inferred due to subclasses being used in existential quantification.

Knowledge modelling: examples

Class(a:driver complete intersectionOf(a:person restriction(a:drives someValuesFrom (a:vehicle)))) **$driver \equiv person \sqcap \exists drives.vehicle$**

Class(a:driver partial a:adult) **$driver \sqsubseteq adult$**

Class(a:grownup complete intersectionOf(a:adult a:person))

$grownup \equiv adult \sqcap person$

- Drivers are defined as persons that drive cars (complete definition)
- We also know that drivers are adults (partial definition)
- So all drivers must be adult persons (e.g. grownups)

An example of axioms being used to assert additional necessary information about a class. We do not need to know that a driver is an adult in order to recognize one, but once we have recognized a driver, we know that they must be adult.

$\exists$partof.animal $\sqcup$ animal $\not\equiv$ plant $\sqcup$ $\exists$partof.plant

Knowledge modelling: Examples

Class(a:cow partial a:vegetarian)

DisjointClasses(unionOf(restriction(a:part_of someValuesFrom (a:animal)) a:animal) unionOf(a:plant restriction(a:part_of someValuesFrom (a:plant))))

Class(a:vegetarian complete intersectionOf(restriction(a:eats allValuesFrom (complementOf(restriction(a:part_of someValuesFrom (a:animal))))) restriction(a:eats allValuesFrom (complementOf(a:animal))) a:animal))

Class(a:mad_cow complete intersectionOf(a:cow restriction(a:eats someValuesFrom (intersectionOf(restriction(a:part_of someValuesFrom (a:sheep)) a:brain)))))

Class(a:sheep partial a:animal restriction(a:eats allValuesFrom (a:grass)))

- Cows are naturally vegetarians
- A mad cow is one that has been eating sheeps brains
- Sheep are animals

Thus a mad cow has been eating part of an animal, which is inconsistent with the definition of a vegetarian

Knowledge modelling: Example

Individual(a:Walt type(a:person) value(a:has_pet a:Huey)
value(a:has_pet a:Louie) value(a:has_pet a:Dewey))
Individual(a:Huey type(a:duck))
Individual(a:Dewey type(a:duck))
Individual(a:Louie type(a:duck))
DifferentIndividuals(a:Huey a:Dewey a:Louie)
Class(a:animal_lover complete intersectionOf(a:person
restriction(a:has_pet minCardinality(3))))
ObjectProperty(a:has_pet domain(a:person) range(a:animal))

- Walt has pets Huey, Dewey and Louie.
- Huey, Dewey and Louie are all distinct individuals.
- Walt has at least three pets and is thus an animal lover.

Note that in this case, we don't actually need to include person in the definition of animal lover (as the domain restriction will allow us to draw this inference).

Knowledge modelling: OWA vs. CWA

OWA: Open World Assumption

The existence of further individuals is possible if it is not explicitly excluded.

OWL uses OWA!

CWA: Closed World Assumption

One assumes that the knowledge base contains all known individuals and all known facts.

		Are all children of Bill male?	*No idea, since we do not know all children of Bill.*	*If we assume that we know everything about Bill, then all of his children are male.*
			DL answers	**Prolog**
child(Bill,Bob)	**Man(Bob)**	**? ⊨ ∀child.Man(Bill)**	**don't know**	**yes**
≤ 1 child.T(Bill)		**? ⊨ ∀child.Man(Bill)**	**yes**	*Now we know everything about Bill's children.*

Knowledge modelling: Domain and Range

- ObjectProperty(xyz:has_topping
  domain(xyz:Pizza)
  range(xyz:Pizza_topping))

  $\top \sqsubseteq \forall has\_topping^{-}.Pizza$
  $\top \sqsubseteq \forall has\_topping.Pizza\_topping$

- Class(xyz:Ice_cream_cone partial
  restriction(xyz:has_topping someValuesFrom (xyz:Ice_cream)))

  $Ice\_cream\_cone \sqsubseteq \exists has\_topping.Ice\_cream$

- If Ice_cream_cone and Pizza *not* disjunct:
  - Ice_cream_cone is classified as Pizza
  - …but: Ice_cream is *not* classified as Pizza_topping
  - Consequences: *all* Ice_cream_cones are Pizzas,
    and *some* Ice_cream is a Pizza_topping

Knowledge modelling: Some Research Challenges

- Concluding with
  - uncertainty (fuzzy, probabilistic)
  - Inkonsistencies (paraconsistent)
  - Rules
  - Further AI-Paradigms (nonmonotonic reasoning, preferences …)
- Maintenance (updates, infrastructure, etc)
- Scalability of reasoning
- …