

# Semantic Web

## Logic Languages

F. Abel, N. Henze, D. Krause

IVS Semantic Web Group

18.12.2008

- 1** Introduction
  
- 2** Propositional Logic
  - BNF-grammar
  
- 3** First Order Logic
  - BNF-grammar
  - Expressing Ontologies
  
- 4** Description Logics

# Introduction - Types of Logic

Language	Ontological Commitment (What exists in the world)	Epistemological Commitment (What an agent believes about facts)
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief 0..1
Fuzzy logic	degree of truth	degree of belief 0..1

From Enrico Franconi: Foundations of Propositional Logic

## Advantages of logic languages

- high-level language in which knowledge can be expressed
- well-understood formal semantics, unambiguous meaning to logical statements
- precise notion of *logical consequence*
- proof systems. Proof systems should be
  - sound (*korrekt: jede ableitbare Aussage gilt ("alles, was beweisbar ist, ist wahr")*)
  - complete (*vollständig: jede wahre Aussage ist formal ableitbar*)
- proof systems allow to trace the proof
- **semantics** of a logical language: the truth of a sentence
- **grounding** the connection, if any, between the logical reasoning process and the real environment
- **First Order Logic:** sound and complete proof systems.

# Propositional Logic - BNF grammar

*Sentence*  $\longrightarrow$  *AtomicSentence* | *ComplexSentence*  
*AtomicSentence*  $\longrightarrow$  *True* | *False* | *Symbol*  
*Symbol*  $\longrightarrow$  *P* | *Q* | *R* | ...  
*ComplexSentence*  $\longrightarrow$   $\neg$ *Sentence*  
| (*Sentence*  $\wedge$  *Sentence*)  
| (*Sentence*  $\vee$  *Sentence*)  
| (*Sentence*  $\implies$  *Sentence*)  
| (*Sentence*  $\iff$  *Sentence*)

# Propositional Logic - Truth table

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \implies Q$	$P \iff Q$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

# First Order Logic - BNF grammar

First Order Logic (FOL) / Predicate Logic / Prädikatenlogik 1. Stufe

<i>Sentence</i>	→	<i>AtomicSentence</i>   ( <i>Sentence</i> <i>Connective</i> <i>Sentence</i> )   <i>Quantifier</i> <i>Variable</i> ,... <i>Sentence</i>   $\neg$ <i>Sentence</i>
<i>AtomicSentence</i>	→	<i>Predicate</i> ( <i>Term</i> , ...)   <i>Term</i> = <i>Term</i>
<i>Term</i>	→	<i>Function</i> ( <i>Term</i> , ..)   <i>Constant</i>   <i>Variable</i>
<i>Connective</i>	→	$\implies$   $\vee$   $\wedge$   $\iff$
<i>Quantifier</i>	→	$\forall$   $\exists$
<i>Constant</i>	→	<i>y</i>   <i>x</i>   <i>john</i>   ...
<i>Variable</i>	→	<i>Y</i>   <i>X</i>   <i>S</i>   ...
<i>Predicate</i>	→	<i>Before</i>   <i>HasColor</i>   <i>Raining</i>   ...
<i>Function</i>	→	<i>Mother</i>   <i>LeftLeg</i>   ...

## Classes

Classes can be expressed by Predicates.

### Example

Mother, Father, ...

Expressing subclasses, disjointness, union, restriction classes, ... is possible:

### Example

$\forall x.(Mother(x) \rightarrow Human(x))$

## Properties

Properties can be expressed by Predicates and Functions.

### Example

has\_husband, has\_child, ...

Domain and range restrictions:

### Example

$$\forall x.\forall y.(has\_husband(x, y) \rightarrow (Female(x) \wedge Male(y)))$$

What about symmetric, functional, transitive, and inverse functions?

## Instances

Instances are expressed as facts.

### Example

Mother(Mary)

has\_husband(Mary,Peter)

Slides from P. Hitzler, S. Rudolph, S. Volz: Semantic Web – Grundlagen

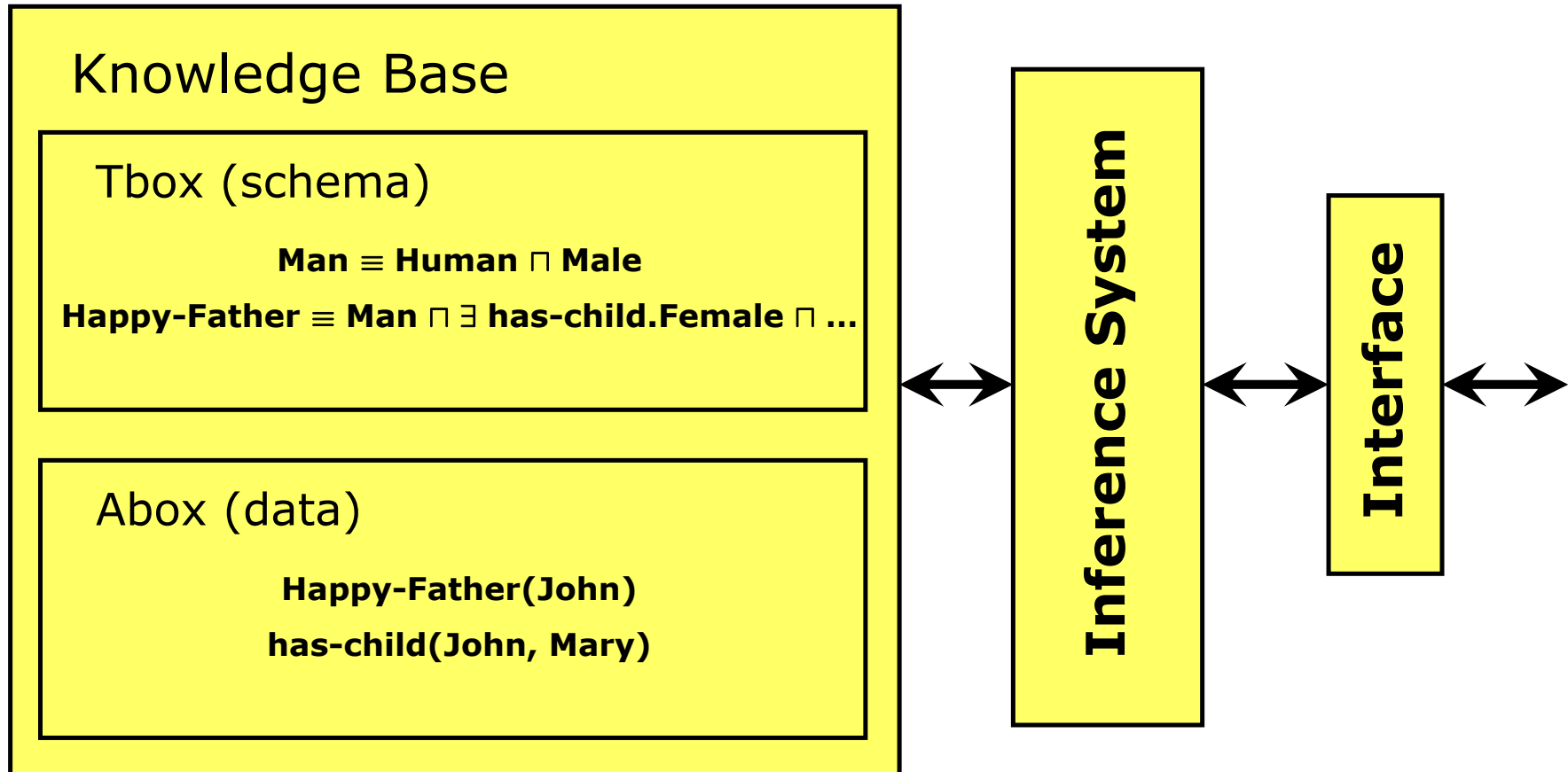
## FOL als Ontologiesprache

- Warum nicht einfach FOL für Ontologien nehmen?
  - FOL kann alles
    - Assembler auch!**
  - FOL ist
    - sehr ausdrucksstark
    - deshalb unhandlich bei der Modellierung
    - schlecht geeignet um Konsens bei der Modellierung zu finden
    - Beweistheoretisch sehr komplex
- ⇒ Suche geeignetes Fragment von FOL

## Beschreibungslogiken (Description Logics, DLs)

- Fragmente von FOL
  - meist entscheidbar
  - ausdrucksstark
  - entwickelt aus semantischen Netzwerken
  - enge Beziehungen zu Modallogiken
- 
- W3C Standard OWL DL ist die Beschreibungslogik SHOIN(D)
  - Wir besprechen zunächst die einfachere ALC

# Allgemeine DL Architektur



## DLs – allgemeiner Aufbau

- DLs sind eine **Familie** logikbasierter Formalismen zur Wissensrepräsentation
- Spezielle Sprachen v.a. charakterisiert durch:
  - Konstruktoren für komplexe Konzepte und Rollen aus einfacheren.
  - Menge von Axiomen um Fakten über Konzepte, Rollen und Individuen auszudrücken.
- ALC ist die kleinste DL, die aussagenlogisch abgeschlossen ist
  - Konjunktion, Disjunktion, Negation sind Konstruktoren, geschrieben  $\sqcap$ ,  $\sqcup$ ,  $\neg$ .
  - Quantoren schränken Rollenbereiche ein:

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

## Weitere DL Konzept- und Rollenkonstruktoren

- Andere Konstruktoren sind z.B.
  - Number restrictions (cardinality constraints) für Rollen:
    - $\geq 3$  hasChild,  $\leq 1$  hasMother
  - 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: hasChild<sup>-</sup>  $\equiv$  hasParent
  - Transitive roles: hasAncestor  $\sqsubseteq^+$  hasAncestor
  - Role composition: hasParent.hasBrother(uncle)

## ALC: Grundbausteine

- Grundbausteine:
  - Klassen
  - Rollen
  - Individuen
- Professor(RudiStuder)
  - Individuum RudiStuder ist in Klasse Professor
- Zugehoerigkeit(RudiStuder,AIFB)
  - RudiStuder ist dem AIFB zugehörig

## ALC: Subklassenbeziehungen

- Professor  $\sqsubseteq$  Fakultaetsmitglied
  - entspricht  $(\forall x)(\text{Professor}(x) \rightarrow \text{Fakultaetsmitglied}(x))$
  - entspricht owl:subClassOf
- Professor  $\equiv$  Fakultaetsmitglied
  - entspricht  $(\forall x)(\text{Professor}(x) \leftrightarrow \text{Fakultaetsmitglied}(x))$
  - entspricht owl:equivalentClass

## ALC: komplexe Klassenbeziehungen

- Konjunktion  $\sqcap$  entspricht owl:intersectionOf
- Disjunktion  $\sqcup$  entspricht owl:unionOf
- Negation  $\neg$  entspricht owl:complementOf
- Professor  $\sqsubseteq$  (Person  $\sqcap$  Unversitaetsangehoeriger)  
 $\sqcup$  (Person  $\sqcap$   $\neg$ Doktorand)

$$\begin{aligned} &(\forall x)(\text{Professor}(x) \rightarrow \\ &\quad ((\text{Person}(x) \wedge \text{Unversitaetsangehoeriger}(x)) \\ &\quad \vee (\text{Person}(x) \wedge \neg \text{Doktorand}(x)))) \end{aligned}$$

## ALC: Quantoren

- $\text{Pruefung} \sqsubseteq \forall \text{hatPruefer. Professor}$   
 $(\forall x)(\text{Pruefung}(x) \rightarrow (\forall y)(\text{hatPruefer}(x,y) \rightarrow \text{Professor}(y)))$   
– entspricht owl:allValuesFrom
- $\text{Pruefung} \sqsubseteq \exists \text{hatPruefer. Person}$   
 $(\forall x)(\text{Pruefung}(x) \rightarrow (\exists y)(\text{hatPruefer}(x,y) \wedge \text{Person}(y)))$   
– entspricht owl:someValuesFrom

## Modellierung in ALC

- owl:nothing:  $\perp \equiv C \sqcap \neg C$
- owl:thing:  $\top \equiv C \sqcup \neg C$
- owl:disjointWith:  
oder gleichbedeutend:  $C \sqcap D \equiv \perp$   
 $C \sqsubseteq \neg D$
- rdfs:range:  $\top \sqsubseteq \forall R.C$
- rdfs:domain:  $\exists R.\top \sqsubseteq C$

## ALC: Syntax

- Folgende Syntaxregeln erzeugen Klassen in ALC. Dabei ist  $A$  eine atomare Klasse und  $R$  eine Rolle.

$$C, D \rightarrow A \mid \top \mid \perp \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$$

- Eine *ALC TBox* besteht aus Aussagen der Form  $C \sqsubseteq D$  und  $C \equiv D$ , wobei  $C, D$  Klassen sind.
- Eine *ALC ABox* besteht aus Aussagen der Form  $C(a)$  und  $R(a,b)$ , wobei  $C$  eine komplexe Klasse,  $R$  eine Rolle und  $a, b$  Individuen sind.
- Eine *ALC-Wissensbasis* besteht aus einer ABox und einer TBox.

## ALC: Semantik

Übersetzung von TBox-Aussagen in die Prädikatenlogik mittels der Abbildung  $\pi$  (rechts).

Dabei sind  $C, D$  komplexe Klassen,  $R$  eine Rolle und  $A$  eine atomare Klasse.

$$\pi(C \sqsubseteq D) = (\forall x)(\pi_x(C) \rightarrow \pi_x(D))$$

$$\pi(C \equiv D) = (\forall x)(\pi_x(C) \leftrightarrow \pi_x(D))$$

$$\pi_x(A) = A(x)$$

$$\pi_x(\neg C) = \neg \pi_x(C)$$

$$\pi_x(C \sqcap D) = \pi_x(C) \wedge \pi_x(D)$$

$$\pi_x(C \sqcup D) = \pi_x(C) \vee \pi_x(D)$$

$$\pi_x(\forall R.C) = (\forall y)(R(x, y) \rightarrow \pi_y(C))$$

$$\pi_x(\exists R.C) = (\exists y)(R(x, y) \wedge \pi_y(C))$$

$$\pi_y(A) = \mathbf{A(y)}$$

$$\pi_y(\neg C) = \neg \pi_y(C)$$

$$\pi_y(C \sqcap D) = \pi_y(C) \wedge \pi_y(D)$$

$$\pi_y(C \sqcup D) = \pi_y(C) \vee \pi_y(D)$$

$$\pi_y(\forall R.C) = (\forall x)(R(y, x) \rightarrow \pi_x(C))$$

$$\pi_y(\exists R.C) = (\exists x)(R(y, x) \wedge \pi_x(C))$$

## DL Wissensbasen

- DL Wissensbasen bestehen aus 2 Teilen:
  - TBox: Axiome, die die Struktur der zu modellierenden Domäne beschreiben (konzeptionelles Schema):
    - **HappyFather**  $\equiv$  **Man**  $\sqcap$   $\exists$ **hasChild.Female**  $\sqcap$  ...
    - **Elephant**  $\sqsubseteq$  **Animal**  $\sqcap$  **Large**  $\sqcap$  **Grey**
    - **transitive(hasAncestor)**
  - ABox: Axiome, die konkrete Situationen (Daten) beschreiben:
    - **HappyFather(John)**
    - **hasChild(John, Mary)**
- Unterscheidung TBox/ABox hat keine logische Bedeutung ... ist aber konzeptionell einfacher.

## Einfaches Beispiel

Terminologisches Wissen (*TBox*):

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

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

Wissen um Individuen (*ABox*):

Orphan(harrypotter)

hasParent(harrypotter,jamespotter)

Semantik und logische Konsequenzen klar, da  
übersetzbar nach FOL.

## OWL und ALC

Folgende OWL DL Sprachelemente sind in ALC repräsentierbar:

- Klassen, Rollen, Individuen
- Klassenzugehörigkeit, Rolleninstanzen
- owl:Thing und owl:Nothing
- Klasseninklusion, -äquivalenz, -disjunktheit
- owl:intersectionOf, owl:unionOf
- owl:complementOf
- Rollenrestriktionen
- rdfs:range und rdfs:domain

## OWL als SHOIN(D): Individuen

- owl:sameAs
  - DL:  $a=b$
  - FOL: Erweiterung durch Gleichheitsprädikat
  
- owl:differentFrom
  - DL:  $a \neq b$
  - FOL:  $\neg(a=b)$

## OWL als SHOIN(D): abgeschlossene Klassen

### Abgeschlossene Klassen

- owl:oneOf
  - DL:  $C \equiv \{a, b, c\}$
  - FOL:  $(\forall x) (C(x) \leftrightarrow (x=a \vee x=b \vee x=c))$
- owl:hasValue
  - Darstellbar mittels owl:someValuesFrom und owl:oneOf

## OWL als SHOIN(D): Zahlenrestriktionen

### Zahlenrestriktionen mittels Gleichheitsprädikat

```

<owl:Class rdf:about="#Pruefung">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatPruefer"/>
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:maxcardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

```

Eine Prüfung kann *höchstens zwei* Prüfer haben.

- DL:  $\text{Pruefung} \sqsubseteq \leq 2 \text{ hatPruefer}$
- In FOL:  $(P \dots \text{Prüfung}, h \dots \text{hatPruefer})$   
 $(\forall x)(P(x) \rightarrow \neg(\exists x_1)(\exists x_2)(\exists x_3)(x_1 \neq x_2 \wedge x_2 \neq x_3 \wedge x_1 \neq x_3 \wedge h(x, x_1) \wedge h(x, x_2) \wedge h(x, x_3)))$

Entsprechend für die anderen Zahlenrestriktionen

## OWL als SHOIN(D): Rollenkonstruktoren

- Rdfs:subPropertyOf
  - DL:  $R \sqsubseteq S$
  - FOL:  $(\forall x)(\forall y)(R(x,y) \rightarrow S(x,y))$
- Entsprechend Rollenäquivalenz
- Inverse Rollen:  $R \equiv S^{-}$ 
  - FOL:  $(\forall x)(\forall y)(R(x,y) \leftrightarrow S(y,x))$
- Transitive Rollen:  $R^+ \sqsubseteq R$ 
  - FOL:  $(\forall x)(\forall y)(\forall z)(R(x,y) \wedge R(y,z) \rightarrow R(x,z))$
- Symmetrie:  $R \equiv R^{-}$
- Funktionalität:  $\top \sqsubseteq \leq 1 R$
- Inverse Funktionalität:  $\top \sqsubseteq \leq 1 R^{-}$

## Datentypen

- Erlaubt ist die Verwendung von Datentypen im zweiten Argument konkreter Rollen in der ABox.
- Ausserdem kann eine Menge konkreter Daten eine abgeschlossene Klasse bilden
- Datentypen lassen sich nicht ohne Weiteres in FOL ausdrücken. Man kann die FOL Semantik aber entsprechend erweitern.

## OWL DL als SHOIN(D): Überblick

Erlaubt sind:

- ALC
- Gleichheit und Ungleichheit zwischen Individuen
- Abgeschlossene Klassen
- Zahlenrestriktionen
- Subrollen und Rollenäquivalenz
- Inverse und transitive Rollen
- Datentypen

## Bezeichner für Beschreibungslogiken

- ALC: Attribute Language with Complement
- S: ALC + Rollentransitivität
- H: Subrollenbeziehung
- O: abgeschlossene Klassen
- I: inverse Rollen
- N: Zahlenrestriktionen  $\leq n$  R etc.
  - Q: Qualifizierende Zahlenrestriktionen  $\leq n$  R.C etc.
- (D): Datentypen
- F: Funktionale Rollen
  
- OWL DL ist SHOIN(D)
- OWL Lite ist SHIF(D)

# Übersicht Syntax für DLs (ohne Datentypen)

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	$\{i_1, \dots, i_n\}$

Roles		
I	Atomic	$R$
	Inverse	$R^-$

Ontology (=Knowledge Base)		
Concept Axioms (TBox)		
	Subclass	$C \sqsubseteq D$
	Equivalent	$C \equiv D$
Role Axioms (RBox)		
I	Subrole	$R \sqsubseteq S$
S	Transitivity	$\text{Trans}(S)$
Assertional Axioms (ABox)		
	Instance	$C(a)$
	Role	$R(a, b)$
	Same	$a = b$
	Different	$a \neq b$

$S = ALC + \text{Transitivity}$

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

# OWL DL als DL: Übersicht Klassenkonstruktoren

Constructor	DL Syntax	Example	FOL Syntax
intersectionOf	$C_1 \sqcap \dots \sqcap C_n$	Human $\sqcap$ Male	$C_1(x) \wedge \dots \wedge C_n(x)$
unionOf	$C_1 \sqcup \dots \sqcup C_n$	Doctor $\sqcup$ Lawyer	$C_1(x) \vee \dots \vee C_n(x)$
complementOf	$\neg C$	$\neg$ Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \dots \sqcup \{x_n\}$	{john} $\sqcup$ {mary}	$x = x_1 \vee \dots \vee x = x_n$
allValuesFrom	$\forall P.C$	$\forall$ hasChild.Doctor	$\forall y.P(x, y) \rightarrow C(y)$
someValuesFrom	$\exists P.C$	$\exists$ hasChild.Lawyer	$\exists y.P(x, y) \wedge C(y)$
maxCardinality	$\leq nP$	$\leq 1$ hasChild	$\exists \leq n y.P(x, y)$
minCardinality	$\geq nP$	$\geq 2$ hasChild	$\exists \geq n y.P(x, y)$

Beliebig komplexes Schachteln von Konstruktoren erlaubt:

Person  $\sqcap \forall$ hasChild.(Doctor  $\sqcup \exists$ hasChild.Doctor)

## OWL DL als DL: Übersicht Axiome

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 <sup>-</sup>
transitiveProperty	$P^+ \sqsubseteq P$	ancestor <sup>+</sup> $\sqsubseteq$ ancestor
functionalProperty	$\top \sqsubseteq \leq 1P$	$\top \sqsubseteq \leq 1$ hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leq 1P^-$	$\top \sqsubseteq \leq 1$ hasSSN <sup>-</sup>

- General Class Inclusion ( $\sqsubseteq$ ) genügt:  
 $C \equiv D$  gdw (  $C \sqsubseteq D$  und  $D \sqsubseteq C$  )
- Offensichtliche FOL-Äquivalenzen  
 $C \equiv D \Leftrightarrow (\forall x) ( C(x) \leftrightarrow D(x) )$   
 $C \sqsubseteq D \Leftrightarrow (\forall x) ( C(x) \rightarrow D(x) )$

## Komplexitäten (worst-case)

OWL Variante	Datenkomplexität	Kombinierte Komplexität
OWL Full	unentscheidbar	unentscheidbar
OWL DL	unbekannt	NExptime
OWL DL ohne Nominals	NP	Exptime
OWL Lite	NP	Exptime

Datenkomplexität: nur bezüglich ABox

Kombinierte Komplexität: bezüglich ABox und TBox