# INFO216:
# **Advanced Modelling**

## Theme, spring 2017:
## **Modelling and Programming the Web of Data**

Andreas L. Opdahl
<Andreas.Opdahl@uib.no>

# Session S13: OWL DL

- Themes:
  - description logic
  - decision problems
  - OWL DL
  - Manchester OWL-syntax
- Practical stuff:
  - perhaps Jena's OntModel class
  - we skip Protege-OWL 3 programming

# Readings

- Forum links (cursory):
  - http://www.w3.org/TR/owl2-primer/
    - show: Turtle and Manchester syntax
    - hide: other syntaxes
  - Description Logic Handbook:
    - Chapter 1: Nardi & Brachman:
      Introduction to Description Logics
    - Chapter 2: Baader & Nutt:
      Formal Description Logics *(gets hard)*

# Description Logic
# (DL)

# Description logics *(perhaps from INFO100?)*

- Description Logic (DL)
  - a simple *fragment* of predicate logic
    - ...or, rather, a *family of such fragments*
  - not very *expressive* ("uttrykkskraftig")
  - but (can have) *good decision problems*, i.e.,
    - it answers *decision problems* (rather) quickly
- Suitable for describing *concepts* ("begreper")
  - formal basis for *OWL DL*
  - can be used to:
    - describe *concepts* and their *roles* ("Tbox")
    - describe *individuals* and their *roles* ("ABox")

# Relationship to other logics

- *Proposition logics* are about *statements* (*propositions*):
  **"Martha is a Woman" ⇐**
  **"Martha is Human" ∧ "Martha is Female"**
- (First order) *predicate logics* are about *predicates* and *objects*:
  - **∀x. (Woman(x) ⇔ Human(x) □ Female(x))**
- *Description logics* are about *concepts*:
  - **Woman ≐ Human ⊓ Female**
  - ...and also *roles* and *individuals*
- There are many other logic systems:
  - *modal logics*: necessarily □, possibly ◊
  - *temporal logics*: always □, sometimes ◊, next time ○

# Definition of concepts ("begreper")

- **Woman ≐ Human ⊓ Female**
- **Man ≐ Human ⊓ ¬Woman**
- **Parent ≐ Mother ⊔ Father**
  - concepts: **Male, Human, Father, Mother...**
  - definition: **≐**
  - conjuction (and): **⊓**
  - disjunction (or): **⊔**
  - negation (not): **¬**
  - nested expressions: **(   )**
- **Childless ≐** ??

# Definition of concepts ("begreper")

- **Woman ≐ Human ⊓ Female**
- **Man ≐ Human ⊓ ¬Woman**
- **Parent ≐ Mother ⊔ Father**
  - concepts: **Male, Human, Father, Mother...**
  - definition: **≐**
  - conjuction (and): **⊓**
  - disjunction (or): **⊔**
  - negation (not): **¬**
  - nested expressions: **( )**
- **Childless ≐ Human ⊓ ¬Parent**

# Atomic and defined concepts and roles

- *Atomic concepts* are given
  - *corresponds to OWL-NamedClasses that are not composed from other classes*
- *Defined concepts*
  - *corresponds to OWL-NamedClasses that are composed from other classes*
  - defined by *concept expressions*
  - appear on the left side of $\doteq$ axioms
- Similar distinction between *atomic and defined roles*

# Roles

- **Mother ≐ Female ⊓ ∃hasChild.⊤**
- **Bachelor ≐ Male ⊓ ¬∃hasSpouse.⊤**
- **Uncle ≐ Male ⊓ ∃hasSibling.Parent**
  - roles: **hasChild, hasSibling...**
  - universal concept ("top"): ⊤
  - existential restriction: ∃
- **Grandparent ≐** ??
- **Grandparent ≐** ..((w/o Mother & Father))..
- **Uncle ≐** ..((without Parent))..

# Roles

- **Mother ≐ Female ⊓ ∃hasChild.⊤**
- **Bachelor ≐ Male ⊓ ¬∃hasSpouse.⊤**
- **Uncle ≐ Male ⊓ ∃hasSibling.Parent**
  - roles: **hasChild, hasSibling...**
  - universal concept ("top"): ⊤
  - existential restriction: ∃
- **Grandparent ≐ Human ⊓ ∃hasChild.Parent**
- **Grandparent ≐** ..((w/o Mother & Father))..
- **Uncle ≐** ..((without Parent))..

# Roles

- **Mother ≐ Female ⊓ ∃hasChild.⊤**
- **Bachelor ≐ Male ⊓ ¬∃hasSpouse.⊤**
- **Uncle ≐ Male ⊓ ∃hasSibling.Parent**
  - roles: **hasChild, hasSibling...**
  - universal concept ("top"): ⊤
  - existential restriction: ∃
- **Grandparent ≐ Human ⊓ ∃hasChild.Parent**
- **Grandparent ≐ Human ⊓ ∃hasChild.∃hasChild.⊤**
- **Uncle ≐** ..((without Parent))..

# Roles

- $Mother \doteq Female \sqcap \exists hasChild.\top$
- $Bachelor \doteq Male \sqcap \neg\exists hasSpouse.\top$
- $Uncle \doteq Male \sqcap \exists hasSibling.Parent$
  - roles: **hasChild, hasSibling...**
  - universal concept ("top"): $\top$
  - existential restriction: $\exists$
- $Grandparent \doteq Human \sqcap \exists hasChild.Parent$
- $Grandparent \doteq Human \sqcap \exists hasChild.\exists hasChild.\top$
- $Uncle \doteq Male \sqcap \exists hasSibling.\exists hasChild.\top$

# Null concept

- **Male ⊓ Female ⊑ ⊥**
  - null concept ("bottom"): ⊥
  - subsumption (sub concept): ⊑
  - equivalence: ≡
- ≐ is used for *definitions* (or just ≡)
- ≡ are used for *equivalence axioms*
- ⊑ are used for *specialisation axioms*
- *This was our first axiom!*
  - so far we have just defined *concepts*
  - we have not used them in *axioms*
- Note the use of ... ⊑ ⊥ ("subsumption of bottom")
  - to say that something is not the case

# More about roles

- **HappyFather ≐ Father ⊓ ∀hasChild.HappyPerson**
  - universal value restriction: ∀
- **MotherOfOne ≐ Mother ⊓ (=1 hasChild.⊤)**
- **Polygamist ≐ (≥3 hasSpouse.⊤)**
  - number restrictions: =, ≥, ≤
- **Narsissist ≐ ∃hasLoveFor.Self**
  - **self references**: Self
- **MassMurderer ≐** ??
- **SelfHater ≐** ??

# More about roles

- **HappyFather ≐ Father ⊓ ∀hasChild.HappyPerson**
  - universal value restriction: ∀
- **MotherOfOne ≐ Mother ⊓ (=1 hasChild.⊤)**
- **Polygamist ≐ (≥3 hasSpouse.⊤)**
  - number restrictions: =, ≥, ≤
- **Narsissist ≐ ∃hasLoveFor.Self**
  - **self references**: Self
- **MassMurderer ≐ (≥4 hasKilled).Human**
- **SelfHater ≐** ??

# More about roles

- HappyFather ≐ Father ⊓ ∀hasChild.HappyPerson
  - universal value restriction: ∀
- MotherOfOne ≐ Mother ⊓ (=1 hasChild.⊤)
- Polygamist ≐ (≥3 hasSpouse.⊤)
  - number restrictions: =, ≥, ≤
- Narsissist ≐ ∃hasLoveFor.Self
  - **self references**: Self
- MassMurderer ≐ (≥4 hasKilled).Human
- SelfHater ≐ ∃haterOf.Self

# Inverse and transitive roles

- **Child $\doteq$ Human $\sqcap$ $\exists$hasChild$^-$.$\top$**
- **hasParent $\doteq$ hasChild$^-$**
- **hasSibling $\doteq$ hasSibling$^-$**
- **BlueBlood $\doteq$ $\forall$hasParent$^*$.BlueBlood**
  - inverse role: **hasChild$^-$**
  - symmetric role: **hasSibling$^-$**
  - transitive role: **hasParent$^*$**
- **Niece $\doteq$** ??

# Inverse and transitive roles

- **Child ≐ Human ⊓ ∃hasChild⁻.⊤**
- **hasParent ≐ hasChild⁻**
- **hasSibling ≐ hasSibling⁻**
- **BlueBlood ≐ ∀hasParent*.BlueBlood**
    - inverse role: **hasChild⁻**
    - symmetric role: **hasSibling⁻**
    - transitive role: **hasParent***
- **Niece ≐ Human ⊓ ∃hasChild⁻.hasSibling.⊤**
- *We are starting to define roles*
    - so far, we have only defined *concepts*

# Composite roles

- Similar to composite concepts, e.g.:
  - **hasUncle ≐ hasParent o hasBrother**
  - **hasLovedChild ≐ hasChild ⊓ hasLoveFor**
  - **hasBrother ≐ (hasSibling | Male)**
- Mostly *not* supported by reasoning engines
  - they have “bad decision problems”
    - meaning that they compute slowly or intractably
  - ...with some exceptions
- **hasDaughter ≐** ??
- **halfSibling ≐** ??

# Composite roles

- Similar to composite concepts, e.g.:
  - **hasUncle ≐ hasParent o hasBrother**
  - **hasLovedChild ≐ hasChild ⊓ hasLoveFor**
  - **hasBrother ≐ (hasSibling | Male)**
- Mostly *not* supported by reasoning engines
  - they have "bad decision problems"
    - meaning that they compute slowly or intractably
  - ...with some exceptions
- **hasDaughter ≐ (hasChild | Female)**
- **halfSibling ≐** ??

# TBox

- *Terminology box* (TBox):
  - a collection of axioms about concepts and properties
  - axioms are definitions, equivalences or subsumptions
  - *definitions* ($\doteq$):
    - atomic concept on the left hand side *(l.h.s.)*
  - *equivalence* ($\equiv$):
    - concept expressions on both sides
  - *subsumption* ($\sqsubseteq$):
    - concept expressions on both sides

# TBox

- *Acyclic TBoxes:*
  - contains only definitions
  - subsumption axioms can (sometimes) be removed:
    - $T \sqsubseteq C$ is transformed into $T \doteq \overline{T} \sqcap C$
    - *when only a single l.h.s. term*
  - every defined concept (or role) can be *expanded* into an expression of only atomic concepts (or roles)
- *Expanded concepts* (or *roles*)
  - defined only in terms of *atomic concepts* (and *roles*)
  - the TBox can (sometimes) be emptied

# Statements about individuals

- So far we have defined concepts and roles (*TBox*)
- We have two types of axioms about individuals (*ABox*):
  - *class assertion* (using a *concept*):

    **`Märtha : Female ⊓ Royal`**
  - *role assertion* (using a *role*):

    **`<Märtha, EmmaTallulah> : hasChild`**

    **`<Märtha, HaakonMagnus> : hasBrother`**
- *Axioms* about concepts/roles and *assertions* about individuals/roles are used to create knowledge bases:
  - concepts, roles in the *TBox* (“the tags”)
  - individuals, roles in the *ABox* (“the tagged data”)

# Syntaxes differ a bit...

- So far we have defined concepts and roles (*TBox*)
- We have two types of axioms about individuals (*ABox*):
  - *class assertion* (using a *concept*):
    **`Female(Märtha),(Female ⊓ Royal)(Märtha)`**
  - *role assertion* (using a *role*):
    **`hasChild(Märtha, EmmaTallulah)`**
    **`hasBrother(Märtha, HaakonMagnus)`**
- *Axioms* about concepts/roles and *assertions* about individuals/roles are used to create knowledge bases:
  - concepts, roles in the *TBox* (“the tags”)
  - individuals, roles in the *ABox* (“the tagged data”)

# Types of axioms

- Terminology axioms (in the TBox):
  - subsumptions: $C \sqsubseteq D$
  - equivalences: $C \equiv D$

    corresponds to: $C \sqsubseteq D$, $D \sqsubseteq C$
  - definitions: $A \doteq C$

C and D are *classes*, A is an *atomic class*!

- Individual assertions (in the ABox):
  - class assertions: a:C
  - role assertions: <a,b>:R

a and b are *individuals*. R is a *role*!

- A knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of
  - TBox: $\mathcal{T}$ and ABox: $\mathcal{A}$

# Decision Problems

# Reasoning over knowledge bases

- *What more can we do with ontologies?*
- For example:
  - a *security ontology* that describes an organisation and its computer systems as concepts, roles and individuals
  - can answer *competency questions*, e.g.:
    - are all the *security levels* subclasses of one another?
    - what is the highest security level of a *temporary*?
    - what is the necessary security level of a *component*?
    - which employees have access to *critical data*?
    - for which *security roles* is an employee qualified?
    - which individuals are *suspicious persons*?
  - *DL offers a clear and compact way or representing and reasoning about questions such as these!*

# Decision problems

- A computational problem with a yes/no answer, e.g.
  - is C *subsumed* by D ($\mathcal{K} \vDash C \sqsubseteq D$)?
  - are C and D *consistent* ($\mathcal{K} \vDash a:(C \sqcap D)$)?
  - does *a* *belong* to C ($\mathcal{K} \vDash a:C$)?
  - is *a* *R-related* to *b* ($\mathcal{K} \vDash$ <a,b>:R)?

C and D are classes,
a and b are individuals.
R is a role!

- *Decidability ("bestembarhet")*:
  - we can always calculate the yes/no answer in finite time
- *Semi-decidability ("semibestembarhet")*:
  - we can always calculate a yes-answer in finite time,
    ...but not always a no-answer
- *Undecidability ("ubestembarhet"):*
  - we cannot always calculate the answer in finite time

# Decision problems for concepts

- There are four basic decision problems for concepts:
  - consistency: whether there is an individual **a** so that

    $\mathcal{T} \models a{:}C$,

    $\mathcal{T} \not\models C \sqsubseteq \bot$
  - subsumption: $\mathcal{T} \models C \sqsubseteq D$,

    $\mathcal{T} \models C \sqcap \neg D \sqsubseteq \bot$
  - equivalence: $\mathcal{T} \models C \equiv D$ *or* $C \equiv_{\mathcal{T}} D$,

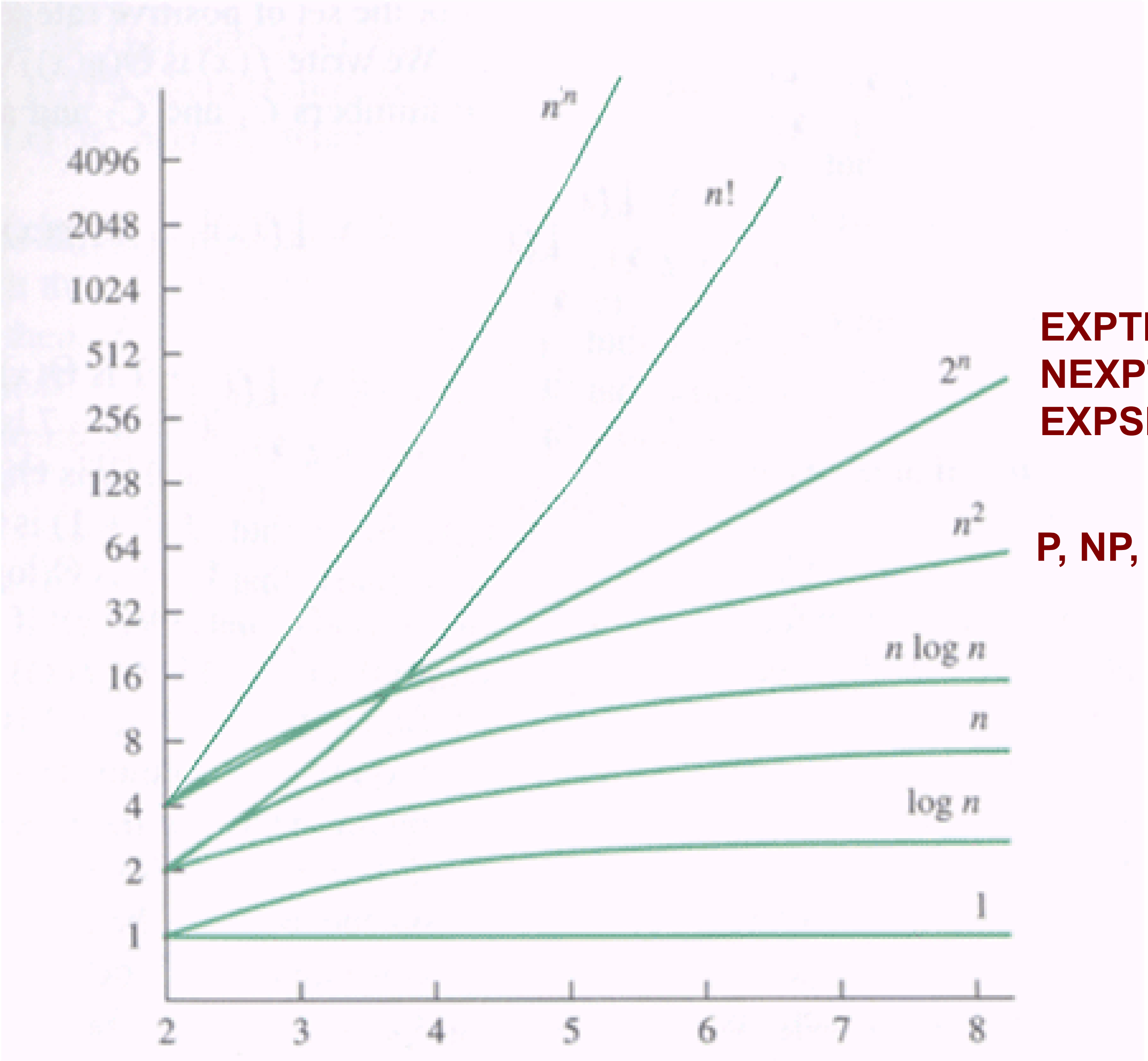
    $\mathcal{T} \models C \sqsubseteq D,\ D \sqsubseteq C$
  - disjunction: $\mathcal{T} \models C \sqcap D \sqsubseteq \bot$
- *All four can be reduced to subsumption or consistency!*
- $\mathcal{T}$ can be *emptied*, by expanding all its concepts

# Decision problems for individuals

- Decision problems for individuals and roles:
  - instance checking: $\mathcal{A} \models$ **a:C**,
    $\not\models \mathcal{A} \sqcap \neg$**(a:C)**
    *is individual **a** member of class **C**?*
  - role checking: $\mathcal{A} \models$ **<a,b>:R**,
    $\not\models \mathcal{A} \sqcap \neg$**(<a,b>:R)**
    *is individual **a** **R**-related to individual **b**?*
  - classifications (not yes/no):
    to which classes does **a** belong?
    all individuals of class **C**?
- *All boil down to consistency checking for ABoxes*
  - ...under certain (rather weak) conditions

# Complexity

- Decidability is often necessary
  - but not enough
  - we also want a decision “in reasonable time”
  - different DL-variants have different *complexity*
  - many different *complexity classes*
    - polynomial (**P**), exponential (**EXP**)...
    - ...in time and space
- *Tractable* (or *feasible*) complexity
  - acceptable complexity for large knowledge bases
  - typically *polynomial* complexity (**P**)
  - complexity grows $O(n^c)$ of problem size $n$




**EXPTIME, NEXPTIME, EXPSPACE**

**P, NP, PSPACE**

# DL-complexity

- We have presented many DL-notations
  - *do not* use all at the same time!
  - that gives high complexity
  - which is why we have different OWL Profiles
- Complexity calculator on the net:
  - *Complexity of reasoning in Description Logics*
    http://www.cs.man.ac.uk/~ezolin/dl/

# OWL DL

# Relation to OWL

- OWL DL and description logic are closely matched
  - everything in OWL DL has a DL-counterpart
  - most everything in DL can be expressed in OWL DL
- DL is a family of logic systems:
  - some of them correspond to particular OWL profiles
  - OWL1 DL: $\mathcal{SHOIN}^{(\mathcal{D})}$
  - OWL2 DL: $\mathcal{SROIQ}^{(\mathcal{D})}$

# OWL profiles revisited

- OWL "1" (2002):
  - *OWL Full – "anything goes"*
  - OWL DL – fragment of OWL Full,
    - formal semantics through *description logic*
  - OWL Lite – simple fragment of OWL DL, not much used
- OWL 2 (2008):
  - *OWL2 Full – "anything goes"*
  - OWL2 DL – fragment of OWL2 full, extension of OWL DL
  - OWL2 DL – has three further fragments:
    - OWL2 EL – quick reasoning, fragment of OWL2 DL
    - OWL2 RL – rule language, fragment of OWL2 DL
    - OWL2 QL – query language, fragment of OWL2 DL

# And there is more...

- A few other constructions
- Formal definitions of
    - syntax (rules for valid expressions, reasoning)
    - semantics (rules for interpreting expressions)
- Tools and techniques
- Lots of applications

# Protege-OWL

# Protege-OWL

- Extension of Protegé
    - ordinary Protegé supports *frames*
    - Protegé-OWL
        - reuses much of the Protege-Frames GUI

# Protege-OWL 3.x

- Supports OWL 1.1:
  - uses *Jena* internally
  - wraps Jena's API with a *Protege-OWL API*
    - stays with Jena's graph metaphor
    - you “create the ontology as a graph”
  - many plug-ins:
    - SWRL, Jess, reasoning...
  - still actively developed

# Protege-OWL 4.x, 5.x, 6 beta

- Supports OWL 2:
  - complete reimplementation of internals
  - *not* based on Jena
  - offers a dedicated *OWL API* (in Java)
    - description-logic metaphor
    - your "build the ontology from axioms"
  - more and more plug-ins
  - still actively developed

# Manchester OWL syntax

# Manchester OWL-syntax

- A simple DL notation without special symbols
  - used by Protege-OWL to construct classes
  - similar to DL syntax
- ```
  Class: Woman
      EquivalentTo: Human and Female
  ```
- ```
  Class: Man
      EquivalentTo: Human and not Female
  ```
- ```
  Class: Parent
      EquivalentTo: Mother or Father
  ```
- Can be used to *serialise* complete ontologies
  - ...we will look mostly at Tbox expressions
- http://www.w3.org/TR/owl2-manchester-syntax/
  ```

# Comparison

- DL:

  **Male ≐ Human ⊓ ¬Female**

- Machester OWL:

```
Class: Man
    EquivalentTo: Human and not Female
```

- TURTLE:

```
family:Man owl:equivalentClass
    owl:intersectionOf (
        family:Human
        [   a owl:Class ;
            owl:complementOf family:Woman
        ]
    ) .
```

# Roles in Manchester OWL syntax

- Class: Mother
  EquivalentTo:
  Female and hasChild some owl:Thing
- Class: Bachelor
  EquivalentTo:
  Male and not hasSpouse some owl:Thing
- Class: Uncle
  EquivalentTo:
  Male and hasSibling some Parent
  - universal concept (top): owl:Thing
  - existential restriction: some

# Null concept in Manchester OWL syntax

- **Class:** **<class-name>**

  **EquivalentTo: Male and Female**

  **SubClassOf: owl:Nothing**

  - null concept (bottom): **owl:Nothing**
  - subsumption (subconcept): **SubClassOf:**
  - equivalence: **EquivalentTo:**
    - ...used both for *definitions* and for *axioms*

# More roles in Manchester OWL syntax

- **Class: HappyFather**
  **EquivalentTo:**
  **Father and hasChild only Happy**
  - value restriction: **only**
- **Class: MotherOfOne**
  **EquivalentTo: Mother and**
  **hasChild exactly 1**
- **Class: Bigamist**
  **EquivalentTo: hasSpouse min 2**
  - number restriction: **exactly**, **min**, **max**
- **Class: Narcissist**
  **EquivalentTo: loves some Self**

# Inverse, symmetric and transitive roles

- **Class: Child**
  **EquivalentTo:**
  **Human and inverse hasChild some owl:Thing**
- **Class: hasParent**
  **EquivalentTo: inverse hasChild**
- **ObjectProperty: hasSibling**
  **Characteristic: Symmetric**
- **ObjectProperty: hasAncestor**
  **Characteristic: Transitive**
- inverse role: **inverse**
  - symmetric role:
    **Characteristic: SymmetricProperty**
  - transitive role:
    **Characteristic: TransitiveProperty**