INFO216: **Knowledge Graphs**

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

Session S14: OWL DL

- Themes:
  - description logic
  - decision problems
  - OWL DL
  - Manchester OWL-syntax

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

- 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*:
  - $\forall x.\,(\mathrm{Woman}(x) \Leftrightarrow \mathrm{Human}(x) \wedge \mathrm{Female}(x))$
- *Description logics* are about *concepts*:
  - $\mathrm{Woman} \doteq \mathrm{Human} \sqcap \mathrm{Female}$
  - ...and also about *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: **Human, Female, Woman…**
    - definition: ≐
    - conjunction (and): ⊓
    - disjunction (or): ⊔
    - negation (not): ¬
    - nested expressions: **(   )**
- **Childless** ≐ ..using Human and Parent..

Definition of concepts (“begreper”)

- **Woman $\doteq$ Human $\sqcap$ Female**
- **Man $\doteq$ Human $\sqcap$ $\neg$ Woman**
- **Parent $\doteq$ Mother $\sqcup$ Father**
  - concepts: **Human, Female, Woman...**
  - definition: $\doteq$
  - conjuction (and): $\sqcap$
  - disjunction (or): $\sqcup$
  - negation (not): $\neg$
  - nested expressions: **(   )**
- **Childless $\doteq$ Human $\sqcap$ $\neg$ Parent**

Types of concepts ("begreper")

- $\mathbf{Woman} \doteq \mathbf{Human} \sqcap \mathbf{Female}$
- $\mathbf{Man} \doteq \mathbf{Human} \sqcap \neg \mathbf{Woman}$
- $\mathbf{Parent} \doteq \mathbf{Mother} \sqcup \mathbf{Father}$
    - named (or atomic) concepts:
      Human, Female, Woman...
    - complex concepts / concept expressions:
      $\neg$ **Woman, Human** $\sqcap$ **Female...**
    - (named) basic concepts: **Human, Female...**
        - only used on the r.h.s. of definitions
    - (named) defined concepts: **Woman, Man...**
  - acyclicity and
  - *unequivocality*, i.e., each named concept used on the l.h.s. of only *a single* definition

Basic and defined concepts and roles

- *Named (or atomic) basic concepts* are given
  - *correspond to OWL-NamedClasses that are not composed from other classes/properties/...*
- *Named (or atomic) defined concepts*
  - *correspond to OWL-NamedClasses that are composed from other classes*
  - defined by *concept expressions*
  - name appears on the left side of an $\doteq$ definition
  - concept expression appears on the right side
- ...similar distinction between *basic and defined roles* later

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** $\doteq$ ..using Human, hasChild, Parent..
- **Grandparent** $\doteq$ ..using only Human, hasChild..
- **Uncle** $\doteq$ ..using Male, hasSibling, hasChild..

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** $\doteq$ ..using only Human, hasChild..
- **Uncle** $\doteq$ ..using Male, hasSibling, hasChild..

Roles

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

Roles

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

Null concept

- **Male** $\sqcap$ **Female** $\sqsubseteq$ $\perp$
  - null concept ("bottom"): $\perp$
  - subsumption (sub concept): $\sqsubseteq$
  - equivalence: $\equiv$
- $\doteq$ is used for *definitions* (or just $\equiv$)
- $\equiv$ are used for *equivalence axioms*
- $\sqsubseteq$ are used for *subsumption axioms*
  - or: containment / specialisation axioms
- Note the use of ... $\sqsubseteq$ $\perp$ ("subsumption of bottom")
  - to say that something is not the case

Null concept

- **Male** $\sqcap$ **Female** $\sqsubseteq \bot$
    - null concept ("bottom"): $\bot$
    - subsumption (sub concept): $\sqsubseteq$
    - equivalence: $\equiv$
- $\doteq$ is used for *definitions* (or just $\equiv$)
- $\equiv$ are used for *equivalence axioms*
- $\sqsubseteq$ are used for *subsumption axioms*
    - or: containment / specialisation axioms
- Note the use of ... $\sqsubseteq \bot$ ("subsumption of bottom")
    - to say that something is not the case
- *This was our first proper axiom!*
    - so far we have just defined *concepts*
    - we have not used them in proper *axioms*

More uses of roles

- **HappyFather** $\doteq$ **Father** $\sqcap$ $\forall$ **hasChild.HappyPerson**
  - universal restriction: $\forall$
- **MotherOfOne** $\doteq$ **Mother** $\sqcap$ **=1 hasChild.**$\top$
- **Polygamist** $\doteq$ **≥3 hasSpouse.**$\top$
  - number restrictions: **=, ≥, ≤**
- **Narsissist** $\doteq$ $\exists$ **hasLoveFor.<u>Self</u>**
  - **self references**: **<u>Self</u>**
- **MassMurderer** $\doteq$ ...using hasKilled, Human...

More uses of roles

- **HappyFather** $\doteq$ **Father** $\sqcap$ $\forall$ **hasChild.HappyPerson**
  - universal restriction: $\forall$
- **MotherOfOne** $\doteq$ **Mother** $\sqcap$ **=1 hasChild.**$\top$
- **Polygamist** $\doteq$ **≥3 hasSpouse.**$\top$
  - number restrictions: **=, ≥, ≤**
- **Narsissist** $\doteq$ $\exists$ **hasLoveFor.<u>Self</u>**
  - **self references**: **<u>Self</u>**
- **MassMurderer** $\doteq$ **≥4 hasKilled.Human**

Inverse and transitive roles

- $\mathbf{Child} \doteq \mathbf{Human} \sqcap \exists\, \mathbf{hasChild}^{-}.\top$
- $\mathbf{hasParent} \doteq \mathbf{hasChild}^{-}$
- $\mathbf{hasSibling} \doteq \mathbf{hasSibling}^{-}$
- $\mathbf{BlueBlood} \doteq \forall\, \mathbf{hasParent}^{*}.\mathbf{BlueBlood}$
  - inverse role: $\mathbf{hasChild}^{-}$
  - symmetric role: $\mathbf{hasSibling}^{-}$
  - transitive role: $\mathbf{hasParent}^{*}$
- **Niece** $\doteq$ ..Woman, hasChild, hasSibling..

Inverse and transitive roles

- $\mathbf{Child} \doteq \mathbf{Human} \sqcap \exists \mathbf{hasChild}^{-}.\top$
- $\mathbf{hasParent} \doteq \mathbf{hasChild}^{-}$
- $\mathbf{hasSibling} \doteq \mathbf{hasSibling}^{-}$
- $\mathbf{BlueBlood} \doteq \forall\, \mathbf{hasParent}^{*}.\mathbf{BlueBlood}$
  - inverse role: $\mathbf{hasChild}^{-}$
  - symmetric role: $\mathbf{hasSibling}^{-}$
  - transitive role: $\mathbf{hasParent}^{*}$
- $\mathbf{Niece} \doteq \mathbf{Woman} \sqcap \exists\, \mathbf{hasChild}^{-}.\mathbf{hasSibling}.\top$
- *We just started to define roles!*
  - until now, 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”
    - i.e., they compute slowly or intractably
  - ...with some exceptions
- **hasDaughter** ≐ ..using hasChild, Female..

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)`**

TBox

- *Terminology box* (TBox):
  - a collection of axioms and definitions
  - axioms are equivalences or subsumptions:
    - *equivalence axioms* ($\equiv$):
      - composite concept (role) expressions on both sides
    - *subsumption* axioms ($\sqsubseteq$):
      - composite concept (role) expressions on both sides
  - terminology boxes can also contain definitions:
    - *definition axioms* ($\doteq$):
      - defined / named concept (role) on the l.h.s.
      - composite concept (role) expression on the r.h.s
    - make it easier to write other axioms

Acyclic, definitional TBox

$$
\begin{array}{rcl}
\mathsf{Woman} & \equiv & \mathsf{Person} \sqcap \mathsf{Female} \\
\mathsf{Man} & \equiv & \mathsf{Person} \sqcap \neg\mathsf{Woman} \\
\mathsf{Mother} & \equiv & \mathsf{Woman} \sqcap \exists \mathsf{hasChild}.\mathsf{Person} \\
\mathsf{Father} & \equiv & \mathsf{Man} \sqcap \exists \mathsf{hasChild}.\mathsf{Person} \\
\mathsf{Parent} & \equiv & \mathsf{Father} \sqcup \mathsf{Mother} \\
\mathsf{Grandmother} & \equiv & \mathsf{Mother} \sqcap \exists \mathsf{hasChild}.\mathsf{Parent} \\
\mathsf{MotherWithManyChildren} & \equiv & \mathsf{Mother} \sqcap \geqslant 3\, \mathsf{hasChild} \\
\mathsf{MotherWithoutDaughter} & \equiv & \mathsf{Mother} \sqcap \forall \mathsf{hasChild}.\neg\mathsf{Woman} \\
\mathsf{Wife} & \equiv & \mathsf{Woman} \sqcap \exists \mathsf{hasHusband}.\mathsf{Man}
\end{array}
$$

Acyclic, and
contains only definitions!

TBox

- *Acyclic TBoxes:*
  - contains only definitions
  - subsumption axioms can (sometimes) be removed:
    - $T \sqsubseteq C$ is transformed into $T \doteq \bar{T} \sqcap C$
      - Example:
        `Male` $\sqsubseteq$ **`Human`** is transformed into
        `Male` $\doteq$ **`Maleness`** $\sqcap$ **`Human`**
    - *when only a single l.h.s. term*
- *Expanded concepts* (or *roles*)
  - every defined concept (or role) can be *expanded* into an expression of only named basic concepts (or roles)
  - defined only in terms of named *basic concepts* (and *roles*)
  - expanded, definitional TBox

Expanded definitional TBox

Only named basic concepts on the right hand sides!

$$
\begin{array}{rcl}
\mathsf{Woman} & \equiv & \mathsf{Person} \sqcap \mathsf{Female} \\
\mathsf{Man} & \equiv & \mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female}) \\
\mathsf{Mother} & \equiv & (\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person} \\
\mathsf{Father} & \equiv & (\mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female})) \sqcap \exists \mathsf{hasChild}.\mathsf{Person} \\
\mathsf{Parent} & \equiv & ((\mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female})) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \\
& & \sqcup ((\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \\
\mathsf{Grandmother} & \equiv & ((\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \\
& & \sqcap \exists \mathsf{hasChild}.(((\mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female})) \\
& & \qquad \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \\
& & \qquad \sqcup ((\mathsf{Person} \sqcap \mathsf{Female}) \\
& & \qquad\quad \sqcap \exists \mathsf{hasChild}.\mathsf{Person})) \\
\mathsf{MotherWithManyChildren} & \equiv & ((\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \sqcap \geqslant 3\, \mathsf{hasChild} \\
\mathsf{MotherWithoutDaughter} & \equiv & ((\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \\
& & \sqcap \forall \mathsf{hasChild}.(\neg(\mathsf{Person} \sqcap \mathsf{Female})) \\
\mathsf{Wife} & \equiv & (\mathsf{Person} \sqcap \mathsf{Female}) \\
& & \sqcap \exists \mathsf{hasHusband}.(\mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female}))
\end{array}
$$

Statements about individuals

- So far axioms about concepts and roles (*TBox*)
- Also 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 *assertion axioms* about individuals/roles are used to create knowledge bases:
  - concepts, roles in the *TBox* (aka “the tags”)
  - individuals, roles in the *ABox* (“the tagged data”)

Syntaxes differ a bit...

- So far axioms about concepts and roles (*TBox*)
- Also 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 *assertion axioms* about individuals/roles are used to create knowledge bases:
  - concepts, roles in the *TBox* (aka "the tags")
  - individuals, roles in the *ABox* ("the tagged data")

Summary 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 *expressions*, A is a *defined concept*!

- Individual assertion axioms (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 ($\mathcal{K} \models \mathtt{C} \sqsubseteq \mathtt{D}$)?
  - are C and D *consistent* ($\mathcal{K} \models \mathtt{a:(C} \sqcap \mathtt{D)}$)?
  - does *a* *belong* to C ($\mathcal{K} \models \mathtt{a:C}$)?
  - is *a* *R-related* to *b* ($\mathcal{K} \models \mathtt{<a,b>:R}$)?
- *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

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

Decision problems for concepts

- There are four basic decision problems for concepts:
  - consistency: whether there is an individual $\mathtt{a}$ so that

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

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

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

    $\mathcal{T} \models \mathtt{C} \sqsubseteq \mathtt{D}$, $\mathtt{D} \sqsubseteq \mathtt{C}$
  - disjunction: $\mathcal{T} \models \mathtt{C} \sqcap \mathtt{D} \sqsubseteq \bot$
- *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: $\mathcal{A} \vDash$ **a:C**,

    $\nvDash \mathcal{A} \sqcap \neg$**(a:C)**

    *is individual **a** member of class/concept **C**?*
  - role checking: $\mathcal{A} \vDash$ **<a,b>:R**,

    $\nvDash \mathcal{A} \sqcap \neg$**(<a,b>:R)**

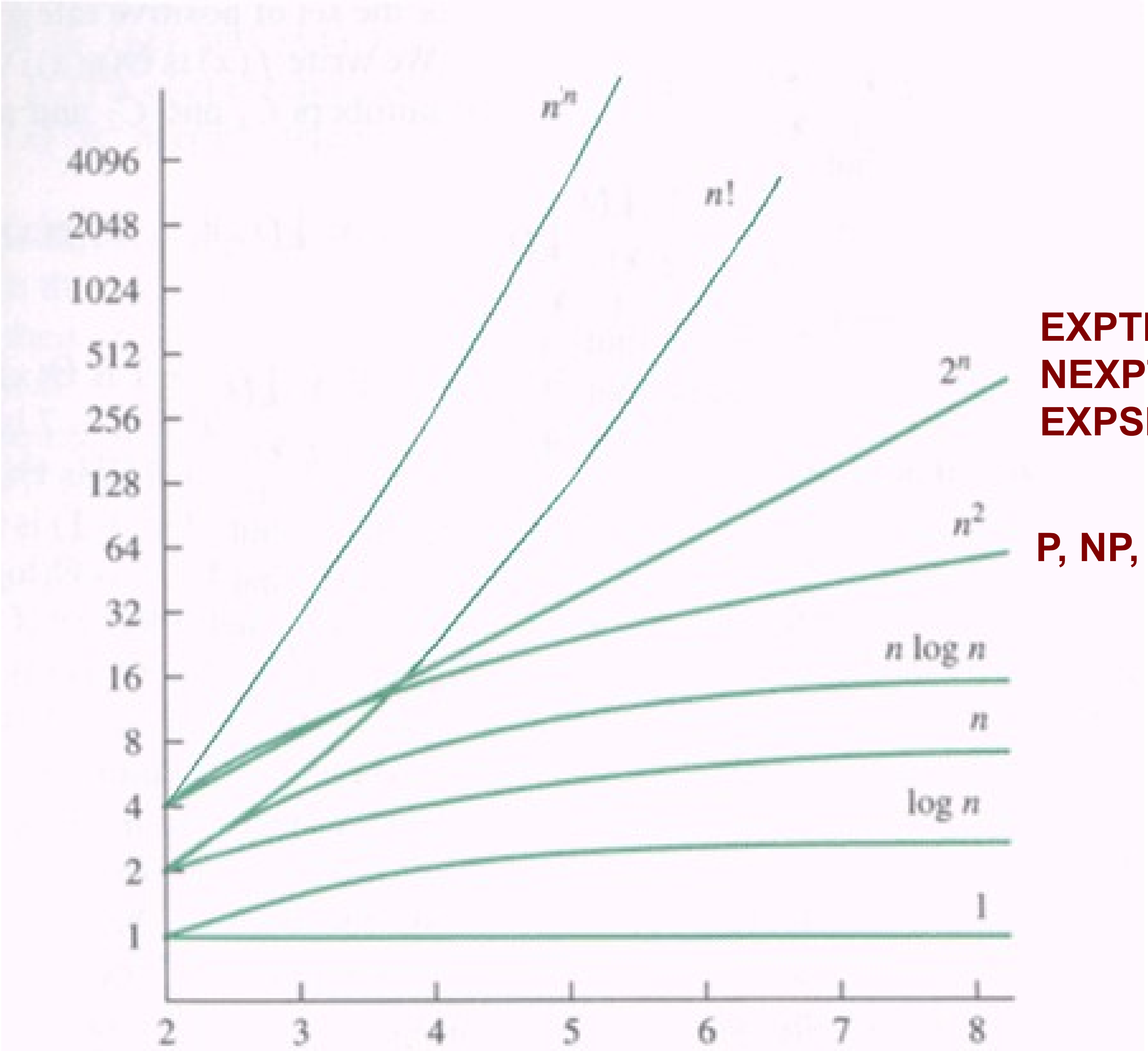
    *is individual **a** **R**-related to individual **b**?*
  - classifications (not yes/no):

    to which classes/concepts does **a** belong?

    all individuals of class/concept **C**?
- *Everything boils 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 EL – quick reasoning, fragment of OWL2 DL
    - OWL2 RL – rule language, fragment of OWL2 DL
      - OWL LD – linked data, fragment of OWL2 RL
    - 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

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