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)

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 ○

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 *roles* and their relations ("Rbox")
    - describe *individuals* and their *roles* ("ABox")

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

Types of concepts ("begreper")

- $\textbf{Woman} \doteq \textbf{Human} \sqcap \textbf{Female}$
- $\textbf{Man} \doteq \textbf{Human} \sqcap \neg \textbf{Woman}$
- $\textbf{Parent} \doteq \textbf{Mother} \sqcup \textbf{Father}$
    - atomic (or basic, primitive) concepts:
      Human, Female, Woman…
        - only used on the r.h.s. of definitions
    - concept expressions (complex concepts):
      $\neg$ **Woman**, **Human** $\sqcap$ **Female…**
        - only used on the r.h.s. of definitions
    - defined (and named) concepts:
      Woman, Man…
        - defined on the l.h.s. of definitions

l.h.s. = left-hand side, r.h.s. = right-hand side

Atomic and defined concepts

- Atomic (or basic) concepts
  - given, always named
  - cannot appear on the l.h.s. of a $\doteq$ definition
  - correspond to simple OWL-NamedClasses
- Concept expressions
  - defined in terms of other concepts (and roles)
  - correspond to complex OWL-Classes
- Defined concepts can also be named
  - must appear on the l.h.s. of a $\doteq$ definition
  - concept_name $\doteq$ concept_expression
- ...similar distinction between atomic and defined roles

Roles

An atomic (or basic) role

- **Mother** ≐ **Female** ⊓ **∃hasChild.**⊤
- **Bachelor** ≐ **Male** ⊓ **¬∃hasSpouse.**⊤
- **Uncle** ≐ **Male** ⊓ **∃hasSibling.Parent**
    - roles: **hasChild**, **hasSibling...**
    - universal concept ("top"): ⊤
    - existential restriction: **∃**
- **Grandparent** ≐ ..using Human, hasChild, Parent..
- **Grandparent** ≐ ..using only Human, hasChild..
- **Uncle** ≐ ..using Male, hasSibling, hasChild..

Roles

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

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 ≐** ....using Male, hasSibling, hasChild....

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 ≐ Male ⊓ ∃ hasSibling.∃ hasChild.⊤**

Null concept

- **Male ⊓ Female ⊑ ⊥**
  - null concept ("bottom"): ⊥
  - subsumption (sub concept): ⊑
- ≐ is used for *definitions* (or just ≡)
  - ≡ is used for *equivalence axioms*
- ⊑ is used for *subsumption axioms*
  - or: containment / specialisation axioms
- Note the use of ... ⊑ ⊥ ("subsumption of bottom") to say that something is not the case

Null concept

- **Male ⊓ Female ⊑ ⊥**
  - null concept ("bottom"): ⊥
  - subsumption (sub concept): ⊑
- ⊑ is used for *subsumption axioms*
  - or: containment / specialisation axioms
- ≐ is used for *definitions* (or just ≡)
  - ≡ is also used for *equivalence axioms*
- Note the use of ... ⊑ ⊥ ("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*

Axioms

- $\doteq$ is used for *definitions*
- $\equiv$ is used for *equivalence axioms*
  - and sometimes for *definitions* too...
- Axioms are equivalences or subsumptions:
  - *subsumption* axioms ($\sqsubseteq$):
    - composite concept (role) expressions on both sides
  - *equivalence axioms* ($\equiv$):
    - composite concept (role) expressions on both sides
    - corresponds to: **C** $\sqsubseteq$ **D**, **D** $\sqsubseteq$ **C**
- expression $\sqsubseteq$ $\bot$ ("subsumption of bottom") is used to say that something is *not* the case

More role definitions

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

More uses of roles

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

Inverse and transitive roles

- **Child ≐ Human ⊓ ∃hasChild⁻.⊤**
- **hasParent ≐ hasChild⁻**
- **hasSibling ≐ hasSibling⁻**
- **BlueBlood ≐ ∀hasParent*.BlueBlood**
  - inverse role: **hasChild⁻**
  - symmetric role: **hasSibling⁻**
  - transitive role: **hasParent***
- **Niece** ≐ ..Woman, hasChild, hasSibling..

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 $\doteq$ Woman $\sqcap$ $\exists$hasChild$^-$.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)**
- Not always supported by reasoning engines
  - they can 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)**
- Not always supported by reasoning engines
  - they can have “bad decision problems”
    - i.e., they compute slowly or intractably
  - ...with some exceptions
- **hasDaughter ≐ (hasChild | Female)**

TBox

- *Terminology box* (TBox):
  - a collection of definitions
  - *definition axioms* ($\doteq$):
    - concept_name $\doteq$ concept_expression
    - defined and named concept on the l.h.s.
    - complex concept expression on the r.h.s
  - *defined names*
    - must appear on the l.h.s. of some $\doteq$ definition
  - *atomic (basic, primitive) names*
    - can only appear on the r.h.s. of $\doteq$ definitions

Acyclic, definitional TBox

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

Note: The example uses $\equiv$
instead of $\doteq$ for definitions

Acyclic, definitional TBox

Defined concepts

Atomic concepts

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

Note: The example uses **$\equiv$** instead of $\doteq$ for definitions

Acyclic and unequivocal!

TBox

- Acyclicity: no cyclic definitions in the TBox
- Unequivocality: each named defined term is only used on the l.h.s. of a single definition
- Concept expansion:
  - every concept can be written as an expression of only atomic concepts
  - algorithm:
    - start with the expression that defines the concept
    - recursively replace all the defined concepts used in the expression with their definitions
    - halt when only atomic concepts remain

Expanded definitional TBox

Only basic concepts on the right hand sides!

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

Note: The example uses $\equiv$ instead of $\doteq$ for definitions

RBox

- *Role box* (RBox):
  - a collection of definitions *of roles*
  - otherwise similar to TBoxes:
    - atomic (basic, primitive) roles
    - role expressions
    - named defined roles
    - role expansion
  - not always necessary (i.e., only atomic roles)

ABox

- So far definitions of concepts and roles (*TBox, RBox*)
- 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

- Together, a TBox, an ABox and possibly an RBox constitute a *knowledge base (KB)*
  - concepts, roles in the *TBox* (aka “the tags”)
  - roles in the *RBox* (also “tags”)
  - individuals, roles in the *ABox* (“the tagged data”)

Syntaxes differ a bit...

- So far definitions of concepts and roles (*TBox, RBox*)
- 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)
- Together, a TBox, an ABox and possibly an RBox constitute a *knowledge base ($\mathcal{K}$)*
  - concepts, roles in the *TBox* (aka "the tags")
  - roles in the *RBox* (also "tags")
  - individuals, roles in the *ABox* ("the tagged data")

Summary of axioms

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

    corresponds to: $C \sqsubseteq D$, $D \sqsubseteq C$

C and D are *expressions*!

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

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

- Knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ or $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$
  - TBox: $\mathcal{T}$ RBox: $\mathcal{R}$ 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: $\mathcal{K} \models$ **C ⊑ D** ?
  - are C and D *consistent*: $\mathcal{K} \models$ **a:(C ⊓ D)**?
  - does *a* *belong* to C: $\mathcal{K} \models$ **a:C** ?
  - is *a* *R-related* to *b:* $\mathcal{K} \models$ **<a,b>:R** ?

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

- Given a knowledge base $\mathcal{K}$, reasoning engines are designed to give yes / no answer
  - ...but not all decision problems are *decidable*
  - ...or have tractable *complexity*
  - *depends on the expressions used!*

Decision problems for concepts

- Four important decision problems for concepts:
  - consistency:
    can an individual **a** exist so that
    $\mathcal{T} \models \mathbf{a:C}$
  - subsumption:
    $\mathcal{T} \models \mathbf{C} \sqsubseteq \mathbf{D}$
  - equivalence:
    $\mathcal{T} \models \mathbf{C} \equiv \mathbf{D}$, also written $\mathbf{C} \equiv_{\mathcal{T}} \mathbf{D}$,
  - disjunction:
    $\mathcal{T} \models \mathbf{C} \sqcap \mathbf{D} \sqsubseteq \bot$
- $\mathcal{T}$ can always be *emptied*, by expanding all its concepts

Decision problems for concepts

- *All four can be reduced to subsumption or consistency!*
  - consistency:

    $\mathcal{T} \models \mathtt{a:C} \quad \leftrightarrow \quad \mathcal{T} \not\models \mathtt{C} \sqsubseteq \bot$

    $\mathcal{T} \not\models \mathtt{a:C} \quad \leftrightarrow \quad \mathcal{T} \models \mathtt{C} \sqsubseteq \bot$
  - subsumption:

    $\mathcal{T} \models \mathtt{C} \sqsubseteq \mathtt{D} \quad \leftrightarrow \quad \mathcal{T} \models \mathtt{C} \sqcap \neg\mathtt{D} \sqsubseteq \bot$
  - equivalence:

    $\mathcal{T} \models \mathtt{C} \equiv \mathtt{D} \quad \leftrightarrow \quad \mathcal{T} \models \mathtt{C} \sqsubseteq \mathtt{D},\ \mathtt{D} \sqsubseteq \mathtt{C}$
  - disjunction:

    $\mathcal{T} \models \mathtt{C} \sqcap \mathtt{D} \sqsubseteq \bot$
- $\mathcal{T}$ can always be *emptied*, by expanding all its concepts

Decision problems for individuals

- Decision problems for individuals and roles:
  - instance checking:
    - is individual **a** member of class/concept **C**?
    - $\mathcal{A} \models \texttt{a:C} \quad \leftrightarrow \quad \not\models \mathcal{A} \sqcap \neg(\texttt{a:C})$
  - role checking:
    - is individual **a** **R**-related to individual **b**?
    - $\mathcal{A} \models \texttt{<a,b>:R} \quad \leftrightarrow \quad \not\models \mathcal{A} \sqcap \neg(\texttt{<a,b>:R})$
  - 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

Tableau algorithm

- A simple reasoning procedure
- Tests satisfiability of a concept C_0
  - C_0 is possible expanded
  - negation normal form (NNF)
- Starts with ABox $A_0 = \{ C_0(x) \}$
- Applies transformation rules that preserve consistency
- Halt when not more rules can be applied
  - ...and halt a branch that contains a contradiction
- If all possible branches contain contradictions:
  - C_0 is unsatisfiable
- C_0 is satisfiable otherwise

The $\rightarrow_{\sqcap}$-rule
Condition: $\mathcal{A}$ contains $(C_1 \sqcap C_2)(x)$, but it does not contain both $C_1(x)$ and $C_2(x)$.
Action: $\mathcal{A}' = \mathcal{A} \cup \{C_1(x), C_2(x)\}$.

The $\rightarrow_{\sqcup}$-rule
Condition: $\mathcal{A}$ contains $(C_1 \sqcup C_2)(x)$, but neither $C_1(x)$ nor $C_2(x)$.
Action: $\mathcal{A}' = \mathcal{A} \cup \{C_1(x)\}$, $\mathcal{A}'' = \mathcal{A} \cup \{C_2(x)\}$.

The $\rightarrow_{\exists}$-rule
Condition: $\mathcal{A}$ contains $(\exists R.C)(x)$, but there is no individual name z such that $C(z)$ and $R(x, z)$ are in $\mathcal{A}$.
Action: $\mathcal{A}' = \mathcal{A} \cup \{C(y), R(x, y)\}$ where y is an individual name not occurring in $\mathcal{A}$.

The $\rightarrow_{\forall}$-rule
Condition: $\mathcal{A}$ contains $(\forall R.C)(x)$ and $R(x, y)$, but it does not contain $C(y)$.
Action: $\mathcal{A}' = \mathcal{A} \cup \{C(y)\}$.

The $\rightarrow_{\geq}$-rule
Condition: $\mathcal{A}$ contains $(\geqslant n\, R)(x)$, and there are no individual names $z_1, \ldots, z_n$ such that $R(x, z_i)$ $(1 \leq i \leq n)$ and $z_i \dot{\neq} z_j$ $(1 \leq i < j \leq n)$ are contained in $\mathcal{A}$.
Action: $\mathcal{A}' = \mathcal{A} \cup \{R(x, y_i) \mid 1 \leq i \leq n\} \cup \{y_i \dot{\neq} y_j \mid 1 \leq i < j \leq n\}$, where $y_1, \ldots, y_n$ are distinct individual names not occurring in $\mathcal{A}$.

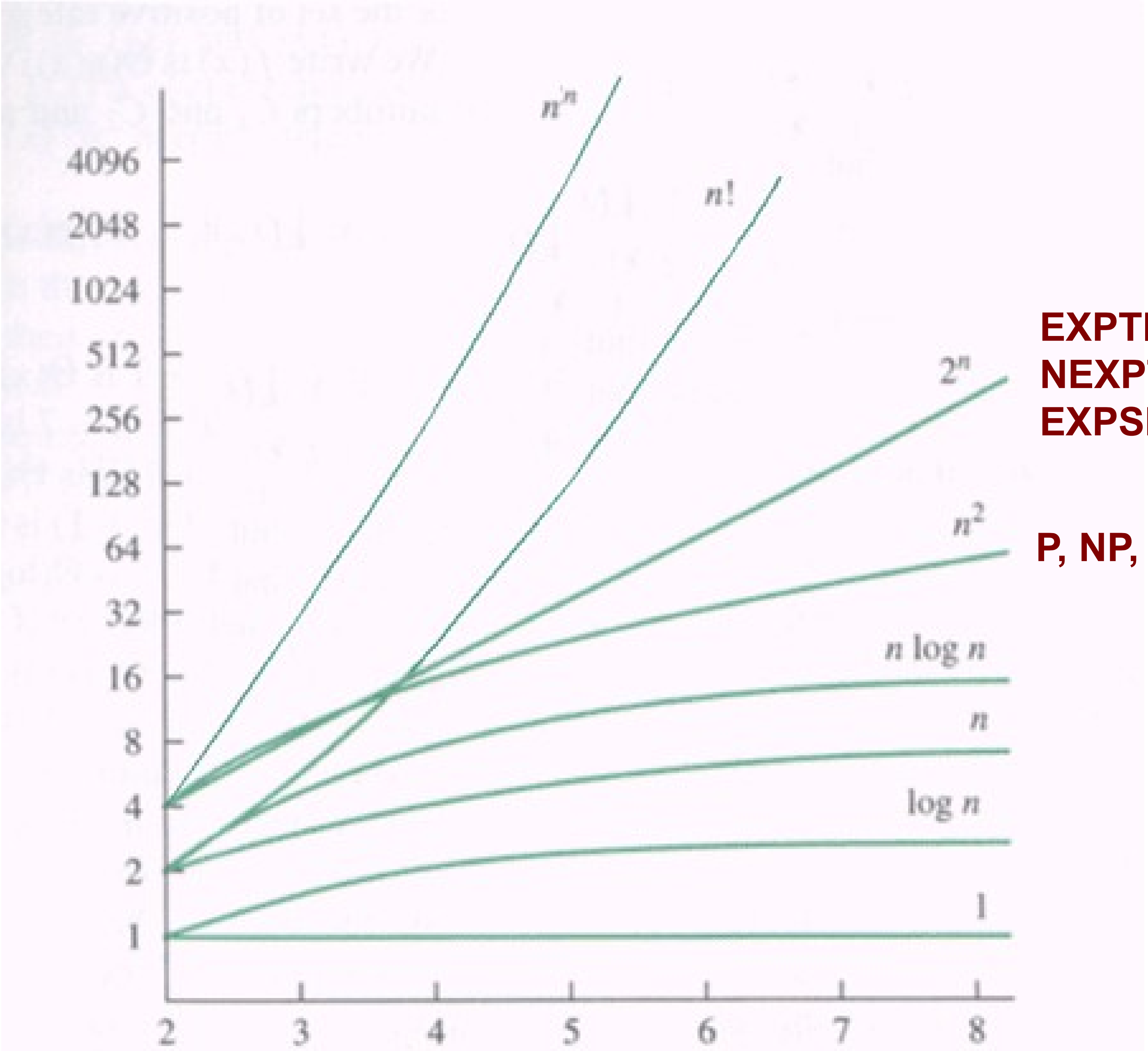
The $\rightarrow_{\leq}$-rule
Condition: $\mathcal{A}$ contains distinct individual names $y_1, \ldots, y_{n+1}$ such that $(\leqslant n\, R)(x)$ and $R(x, y_1), \ldots, R(x, y_{n+1})$ are in $\mathcal{A}$, and $y_i \dot{\neq} y_j$ is not in $\mathcal{A}$ for some $i \neq j$.
Action: For each pair y_i, y_j such that $i > j$ and $y_i \dot{\neq} y_j$ is not in $\mathcal{A}$, the ABox $\mathcal{A}_{i,j} = [y_i/y_j]\mathcal{A}$ is obtained from $\mathcal{A}$ by replacing each occurrence of y_i by y_j.

Complexity

- *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

Complexity

- Decidability is 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:

  $$\textbf{Male} \doteq \textbf{Human} \sqcap \neg\textbf{Female}$$

- 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