Welcome to INFO216:
Knowledge Graphs
Spring 2022

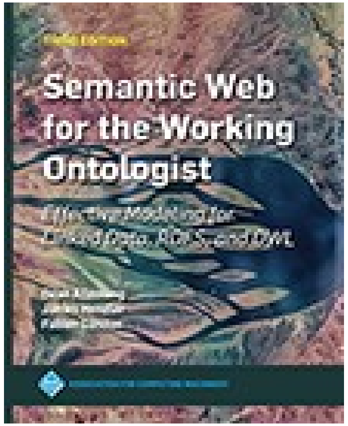
Andreas L Opdahl
<Andreas.Opdahl@uib.no>

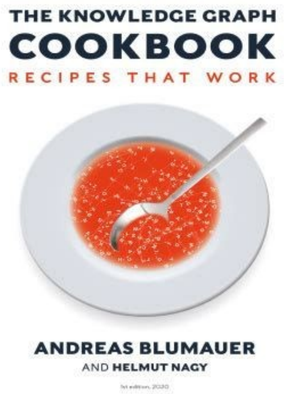
Session 10: Reasoning about KGs (DL)

- Themes:
  - description logic
  - decision problems

Readings

- Materials at http://wiki.uib.no/info216 (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. (Woman(x) ⇔ Human(x) ∧ Female(x))**
- *Description logics* are about *concepts*:
  - **Woman ≐ Human ⊓ 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 many *decision problems* (rather) quickly
- Suitable for describing *concepts* (“begreper”)
  - formal basis for *OWL DL*
  - can be used to:
    - describe *concepts* (“Tbox”) and their *roles* (“Rbox”)
    - describe *individuals* and their relations (“ABox”)

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: $\doteq$
  - conjunction (and): $\sqcap$
  - disjunction (or): $\sqcup$
  - negation (not): $\neg$
  - nested expressions: **(   )**
- **Childless** $\doteq$ ..using Human and Parent..

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 ≐ Human ⊓ ¬ Parent**

Types of concepts ("begreper")

- **Woman ≐ Human ⊓ Female**
- **Man ≐ Human ⊓ ¬ Woman**
- **Parent ≐ Mother ⊔ Father**
  - atomic (or basic, primitive) concepts:
    Human, Female, Woman...
    - only used on the r.h.s. of definitions
  - concept expressions (complex concepts):
    ¬ Woman, Human ⊓ Female...
    - only used on the r.h.s. of definitions
  - defined (and named) concepts:
    Woman, Man...
    - defined on the *l.h.s. (left-hand side)* of definitions

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
  - expressed using other concepts (and roles)
  - must appear on the r.h.s. (right-hand side) of a $\doteq$ definition
  - 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

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

An atomic
(or basic) role

Roles

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

Roles

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

Null concept

- **Male** $\sqcap$ **Female** $\sqsubseteq \bot$
  - null concept ("bottom"): $\bot$
  - subsumption (sub concept): $\sqsubseteq$
- $\sqsubseteq$ is used for *subsumption axioms*
  - or: containment / specialisation axioms
- $\doteq$ is used for *definitions* (or just $\equiv$)
  - $\equiv$ is also used for *equivalence axioms*
- Note the use of ... $\sqsubseteq \bot$ ("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): $\sqsubseteq$
- $\sqsubseteq$ is used for *subsumption axioms*
  - or: containment / specialisation axioms
- $\doteq$ is used for *definitions* (or just $\equiv$)
  - $\equiv$ is also used for *equivalence 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*

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

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

Inverse and transitive roles

- $\texttt{Child} \doteq \texttt{Human} \sqcap \exists\texttt{hasChild}^{-}.\top$
- $\texttt{hasParent} \doteq \texttt{hasChild}^{-}$
- $\texttt{BlueBlood} \doteq \forall\texttt{hasParent}^{*}.\texttt{BlueBlood}$
  - inverse role: $\texttt{hasChild}^{-}$
  - transitive role: $\texttt{hasParent}^{*}$
- $\texttt{Niece} \doteq \texttt{Woman} \sqcap \exists\texttt{hasChild}^{-}.\texttt{hasSibling}.\top$
- *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 OWL-DL and "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 OWL-DL and “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
  - *definitions* ($\doteq$):
    - 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

- $\texttt{Source} \doteq \exists\ \texttt{hasSource}^{-}.\texttt{Content}$
- $\texttt{TrustedContent} \doteq \exists\ \texttt{hasSource}.\texttt{TrustedSource}$
- $\texttt{VerifiedContent} \doteq \exists\ \texttt{verifiedBy}.\texttt{FactChecker}$
- $\texttt{DebunkedContent} \doteq \exists\ \texttt{debunkedBy}.\texttt{FactChecker}$
- $\texttt{UnreliableSource} \doteq \exists\ \texttt{hasSource}^{-}.\texttt{DebunkedContent}$
- $\texttt{VerifyingSource} \doteq \exists\ \texttt{hasSource}^{-}.\texttt{VerifiedContent} \sqcap \forall\ \texttt{hasSource}^{-}.\texttt{VerifiedContent}$

Acyclic, definitional TBox

- $\mathtt{Source} \doteq \exists\, \mathtt{hasSource}^{-}.\mathtt{Content}$
- $\mathtt{TrustedContent} \doteq \exists\, \mathtt{hasSource}.\mathtt{TrustedSource}$
- $\mathtt{VerifiedContent} \doteq \exists\, \mathtt{verifiedBy}.\mathtt{FactChecker}$
- $\mathtt{DebunkedContent} \doteq \exists\, \mathtt{debunkedBy}.\mathtt{FactChecker}$
- $\mathtt{UnreliableSource} \doteq \exists\, \mathtt{hasSource}^{-}.\mathtt{DebunkedContent}$
- $\mathtt{VerifyingSource} \doteq \exists\, \mathtt{hasSource}^{-}.\mathtt{VerifiedContent} \sqcap \forall\, \mathtt{hasSource}^{-}.\mathtt{VerifiedContent}$

Concept expressions of atomic concepts

Defined concepts

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

- $\texttt{Source} \doteq \exists\ \texttt{hasSource}^{-}.\texttt{Content}$
- $\texttt{TrustedContent} \doteq \exists\ \texttt{hasSource}.\texttt{TrustedSource}$
- $\texttt{VerifiedContent} \doteq \exists\ \texttt{verifiedBy}.\texttt{FactChecker}$
- $\texttt{DebunkedContent} \doteq \exists\ \texttt{debunkedBy}.\texttt{FactChecker}$
- $\texttt{UnreliableSource} \doteq \exists\ \texttt{hasSource}^{-}.\ \exists\ \texttt{debunkedBy}.\texttt{FactChecker}$
- $\texttt{VerifyingSource} \doteq \exists\ \texttt{hasSource}^{-}.\ \exists\ \texttt{verifiedBy}.\texttt{FactChecker} \sqcap \forall\ \texttt{hasSource}^{-}.\ \exists\ \texttt{verifiedBy}.\texttt{FactChecker}$

Only basic concepts on the right hand sides!

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
    ```

- A TBox + an ABox (+ 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")

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)`
- A TBox + an ABox (+ 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:
  - given a *source ontology* that describes media content along with its sources and trusrtworthiness
  - we can *answer questions* like, e.g.:
    - is trusted content a type of content?
    - can content be both verified and debunked?
    - is all verified content trusted?
  - *competency questions* are what we want an ontology to answer
  - *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*: $\mathcal{K} \models \mathtt{a}:(\mathtt{C} \sqcap \mathtt{D})$?
  - does *a belong* to C: $\mathcal{K} \models \mathtt{a}:\mathtt{C}$?
  - is *a R-related* to *b:* $\mathcal{K} \models \langle \mathtt{a},\mathtt{b} \rangle : \mathtt{R}$?
- 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!*

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

Decision problems for concepts

- Four important decision problems for concepts:
  - consistency:
    can there be an individual **a** so that
    $\mathcal{T} \models \mathtt{a:C}$
  - subsumption:
    $\mathcal{T} \models \mathtt{C} \sqsubseteq \mathtt{D}$
  - equivalence:
    $\mathcal{T} \models \mathtt{C} \equiv \mathtt{D}$, also written $\mathtt{C} \equiv_{\mathcal{T}} \mathtt{D}$,
  - 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 concepts

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

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

    $\mathcal{T}$ □ $a{:}C \quad \leftrightarrow \quad \mathcal{T} \models C \sqsubseteq \bot$
  - subsumption:

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

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

    $\mathcal{T} \models C \sqcap D \sqsubseteq \bot$
- $\mathcal{T}$ 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**?
    - $\mathcal{A} \models a:C$ $\qquad\qquad$ $\not\models \mathcal{A} \sqcap \neg(a:C)$
  - role checking:
    - is individual **a** **R**-related to individual **b**?
    - $\mathcal{A} \models <a,b>:R$ $\qquad\qquad$ $\not\models \mathcal{A} \sqcap \neg(<a,b>:R)$
  - 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

Decision problems in practice

- Description logic is implemented by *reasoning engines/OWL reasoner*
  - *e.g., HermiT and Pellet*
  - distinct from *inference engines*, such as OWL-RL
- Protegé-OWL
  - comes with HermiT, more plugins can be installed
- Owlready2 (an OWL programming API built around)
  - comes with HermiT and Pellet, HermiT is default
- Solves decision problems, e.g.,
  - classifiy individuals
  - find subclass relationships (subsumptions)
  - find unsatisfiable classes (concepts)
  - detect inconsistencies

Tableau algorithm

- A simple reasoning procedure
- Tests satisfiability of a concept C_0
  - C_0 is possibly expanded
  - negation normal form (NNF)
- Starts with ABox $A_0 = \{ C_0(x) \}$
- Applies transformation rules that preserve consistency

- Halts a branch
  - when no more rules can be applied
  - when the branch contains a contradiction
- If all possible branches contain contradictions:
  - C_0 is unsatisfiable
- Or else:
  - C_0 is satisfiable

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 $z_i \neq z_j$ $(1 \leq i < j \leq n)$ are contained in $\mathcal{A}$.
Action: $\mathcal{A}' = \mathcal{A} \cup \{R(x, y_i) \mid 1 \leq i \leq n\} \cup \{y_i \neq y_j \mid 1 \leq i < j \leq n\}$, 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 $y_i \neq y_j$ is not in $\mathcal{A}$ for some $i \neq j$.
Action: For each pair y_i, y_j such that $i > j$ and $y_i \neq y_j$ 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.

Next week:
Formal ontologies
(OWL-DL)