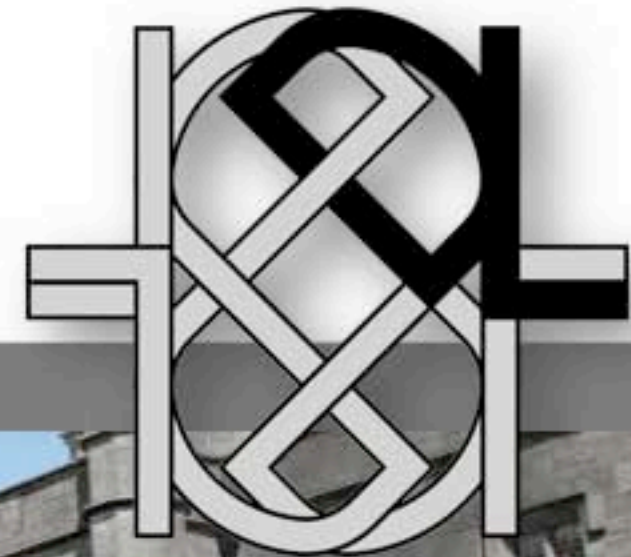


Foundations of Description Logics ...and OWL

Sebastian Rudolph
Karlsruhe Institute of Technology

Copying, distribution, and adaption is allowed for non-commercial purposes (Licence agreement CC-BY-NC).

Institut für Angewandte Informatik und Formale Beschreibungsverfahren

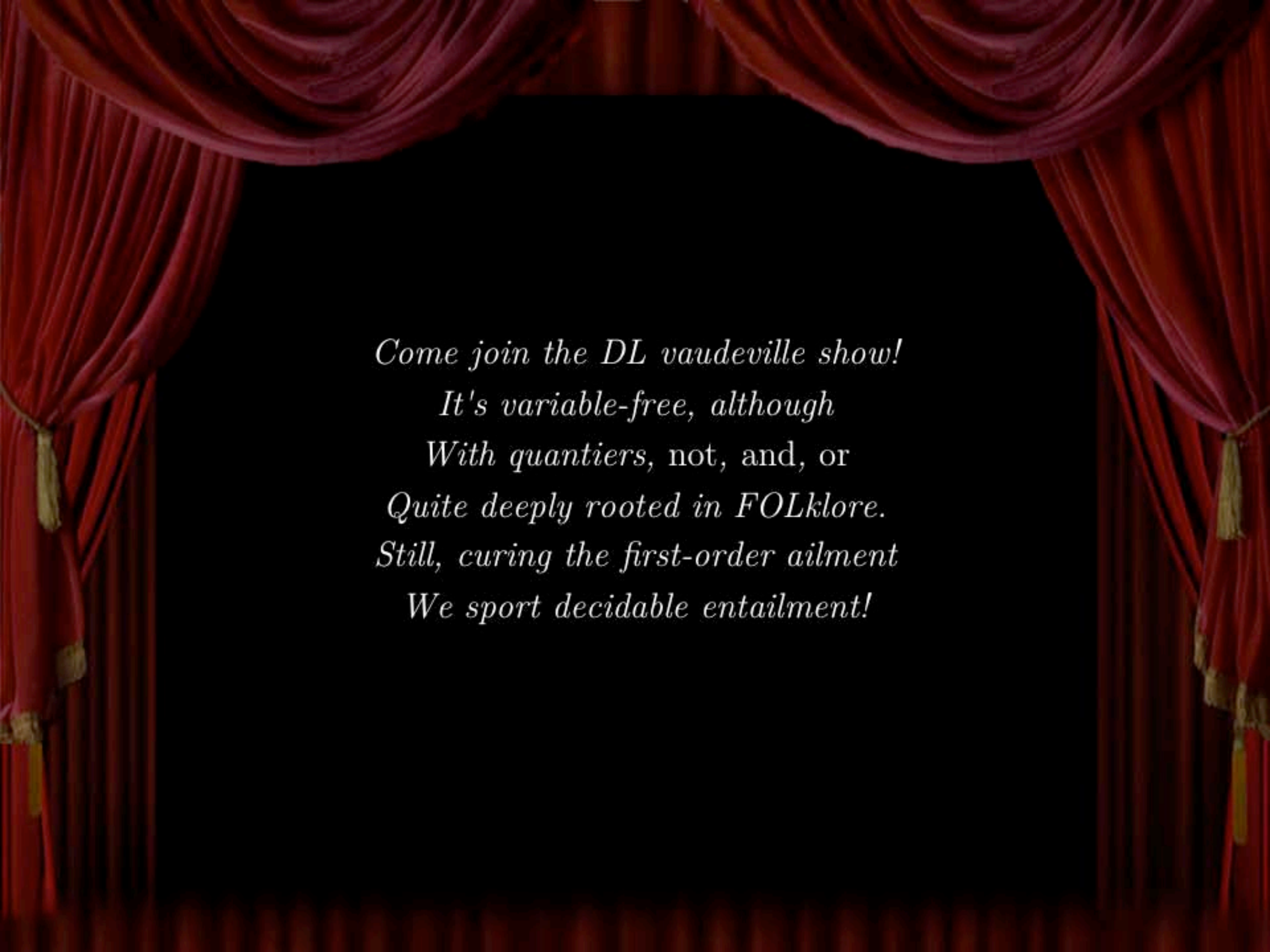


RW2011
Galway City, Ireland
August 23 - August 27

Outline

- Introduction: about DLs and the Semantic Web
- Syntax of Description Logics
- Semantics of Description Logics
- Description Logic Nomenclature
- Equivalences, Normalization, Emulation
- Modeling Power of DLs
- DL Reasoning Tasks
- DL Reasoning Algorithms
- DLs and OWL

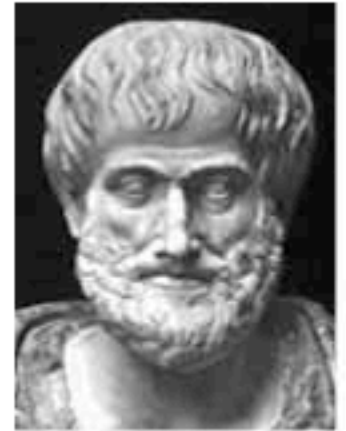


The image features a pair of red, draped curtains with gold tassels, framing a central black area. The text is centered in the black area and is written in a white, italicized serif font.

*Come join the DL vaudeville show!
It's variable-free, although
With quantifiers, not, and, or
Quite deeply rooted in FOLklore.
Still, curing the first-order ailment
We sport decidable entailment!*

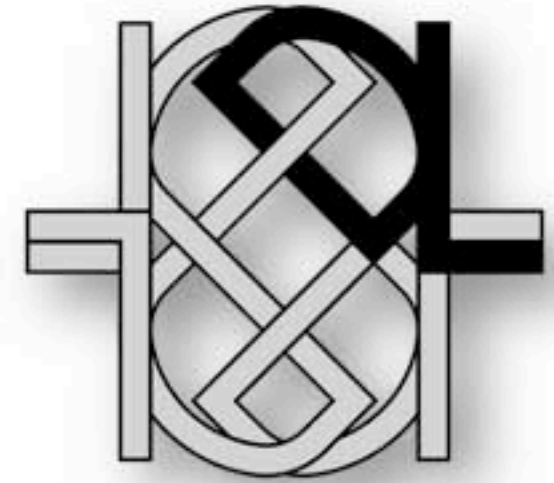
Logic-Based Knowledge Representation

- logic-based knowledge representation already since 2500+ years
- idea to make knowledge explicit by logical “computation“ several hundred years old
- 1930s: disillusion due to results about fundamental limits for the existence of generic algorithms
- adoption of computers and AI as a new area of research leads to intensified studies
- 1980s: logic-based expert systems are applied broadly in practice



Description Logics

- Description Logics (DLs) one of today's main KR paradigms
- influenced standardization of Semantic Web languages, in particular the web ontology language OWL



- comprehensive tool support available

Fact++

Pellet

Hermit



Description Logics

- origin of DLs: **semantic networks** and **frame-based systems**
- downside of the former: only intuitive semantics – diverging interpretations
- DLs provide a **formal semantics** on logical grounds
- can be seen as **decidable** fragments of first-order logic (FOL), closely related to modal logics
- significant portion of DL-related research devoted to clarifying the computational effort of reasoning tasks in terms of their worst-case **complexity**
- despite high complexities, even for expressive DLs exist optimized reasoning algorithms with good average case behaviour

Syntax of Description Logics

Deluxe DL delivery

Will come in boxes (number: three),

Precisely marked with \mathcal{A} , \mathcal{T} , \mathcal{R} .

The first exhibits solid grounding,

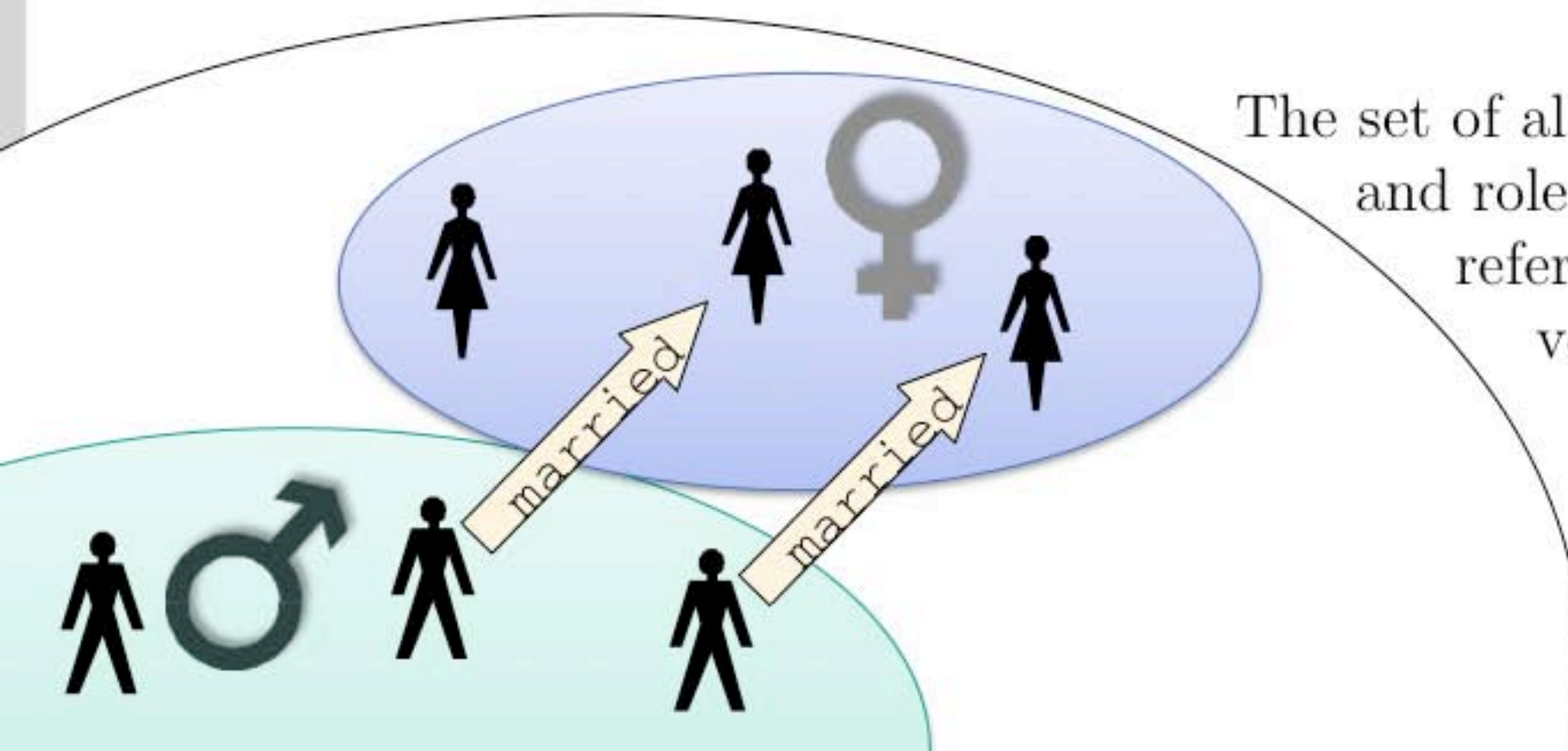
The next allows for simple counting,

The third one's strictly regular.



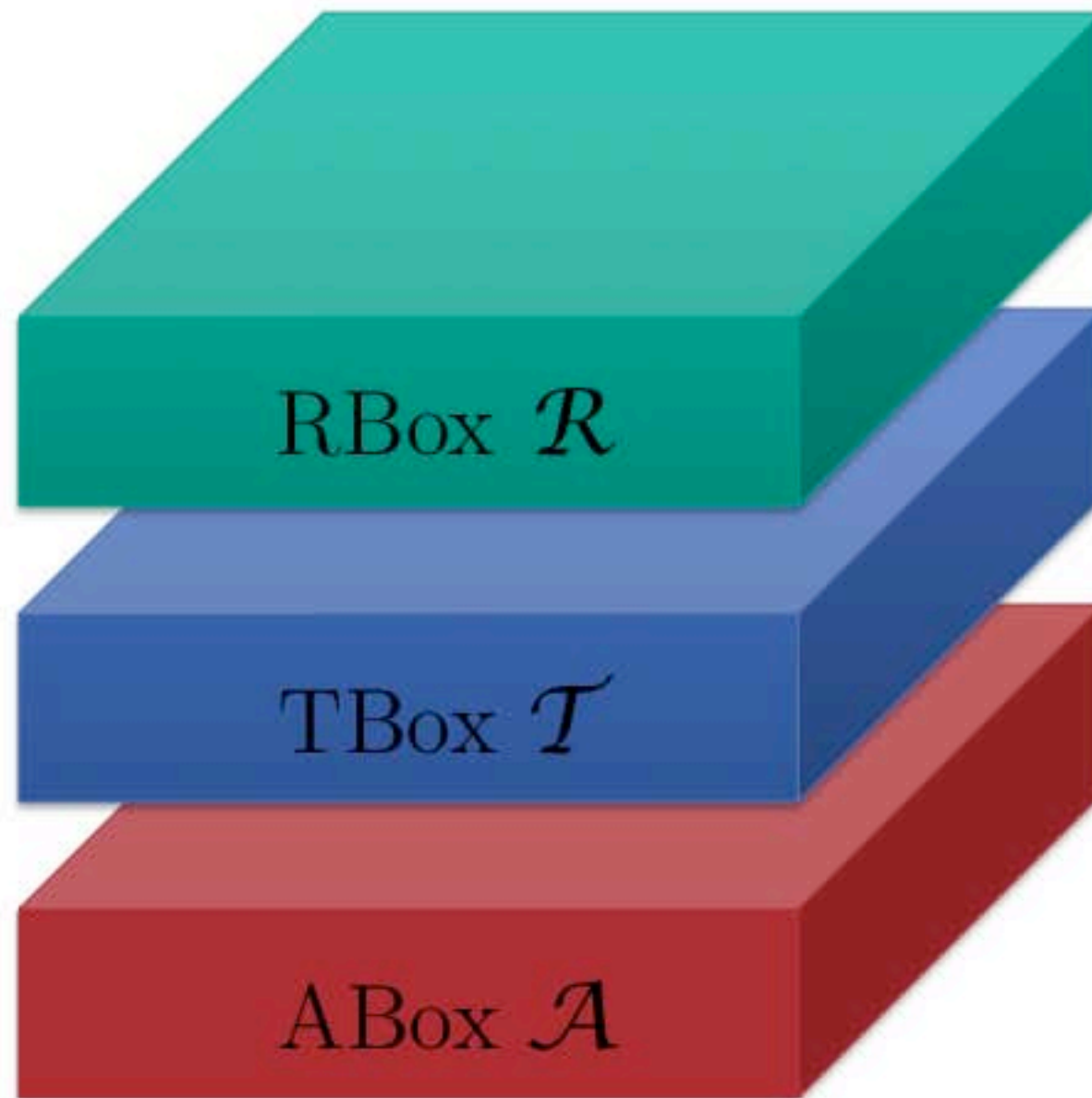
DL Building Blocks

- **individual names:** markus, rhine, sun, excalibur
 - aka: constants (FOL), resources (RDF)
- **concept names:** Female, Mammal, Country
 - aka: unary predicates (FOL), classes (RDFS)
- **role names:** married, fatherOf, locatedIn
 - aka: binary predicates (FOL), properties (RDFS)



The set of all individual, concept and role names is commonly referred to as signature or vocabulary.

Constituents of a DL Knowledge Base



information about roles and their dependencies

information about concepts and their taxonomic dependencies

information about individuals and their concept and role memberships

Roles and Role Inclusion Axioms

- A *role* can be
 - a role name \mathbf{r} or
 - an inverted role name \mathbf{r}^- or
 - the universal role u .
- A *role inclusion axiom* (RIA) is a statement of the form

$$r_1 \circ \dots \circ r_n \sqsubseteq r$$

where r_1, \dots, r_n, r are roles.

Role Simplicity

- Given a set of RIAs, roles are divided into *simple* and *non-simple* roles.
- Roughly, roles are non-simple if they may occur on the rhs of a complex RIA.
- More precisely,
 - for any RIA $r_1 \circ r_2 \circ \dots \circ r_n \sqsubseteq r$ with $n > 1$, r is non-simple,
 - for any RIA $s \sqsubseteq r$ with s non-simple, r is non-simple, and
 - all other properties are simple.

- Example:

$$q \circ p \sqsubseteq r \quad r \circ p \sqsubseteq r \quad r \sqsubseteq s \quad p \sqsubseteq r \quad q \sqsubseteq s$$

non-simple: r, s simple: p, q

The Regularity Condition on RIA sets

- For technical reasons, the set of all RIAs of a knowledge base is required to be *regular*.
- regularity restriction:
 - there must be a strict linear order $<$ on the roles such that
 - every RIA has one of the following forms with $s_i < r$ for all $i=1,2,\dots,n$:

$$r \circ r \sqsubseteq r$$

$$r^- \sqsubseteq r$$

$$s_1 \circ s_2 \circ \dots \circ s_n \sqsubseteq r$$

$$r \circ s_1 \circ s_2 \circ \dots \circ s_n \sqsubseteq r$$

$$s_1 \circ s_2 \circ \dots \circ s_n \circ r \sqsubseteq r$$

- Example 1: $r \circ s \sqsubseteq r$ $s \circ s \sqsubseteq s$ $r \circ s \circ r \sqsubseteq t$
 - regular with order $s < r < t$
- Example 2: $r \circ t \circ s \sqsubseteq t$
 - not regular because form not admissible
- Example 3: $r \circ s \sqsubseteq s$ $s \circ r \sqsubseteq r$
 - not regular because no adequate order exists

RBox

- A *role disjointness* statement has the form

$$\text{Dis}(\mathbf{s}_1, \mathbf{s}_2)$$

where \mathbf{s}_1 and \mathbf{s}_2 are simple roles.

- An *RBox* consists of regular set of RIAs and a set of role disjointness statements.



Concept Expressions

- We define *concept expressions* inductively as follows:
 - every concept name is a concept expression,
 - \top and \perp are concept expressions,
 - for $\mathbf{a}_1, \dots, \mathbf{a}_n$ individual names, $\{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ is a concept expression,
 - for C and D concept expressions, $\neg C$ and $C \sqcap D$ and $C \sqcup D$ are concept expressions,
 - for r a role and C a concept expression, $\exists r.C$ and $\forall r.C$ are concept expressions,
 - for s a simple role, C a concept expression and n a natural number, $\exists r.\text{Self}$ and $\leq ns.C$ and $\geq ns.C$ are concept expressions.

RBox

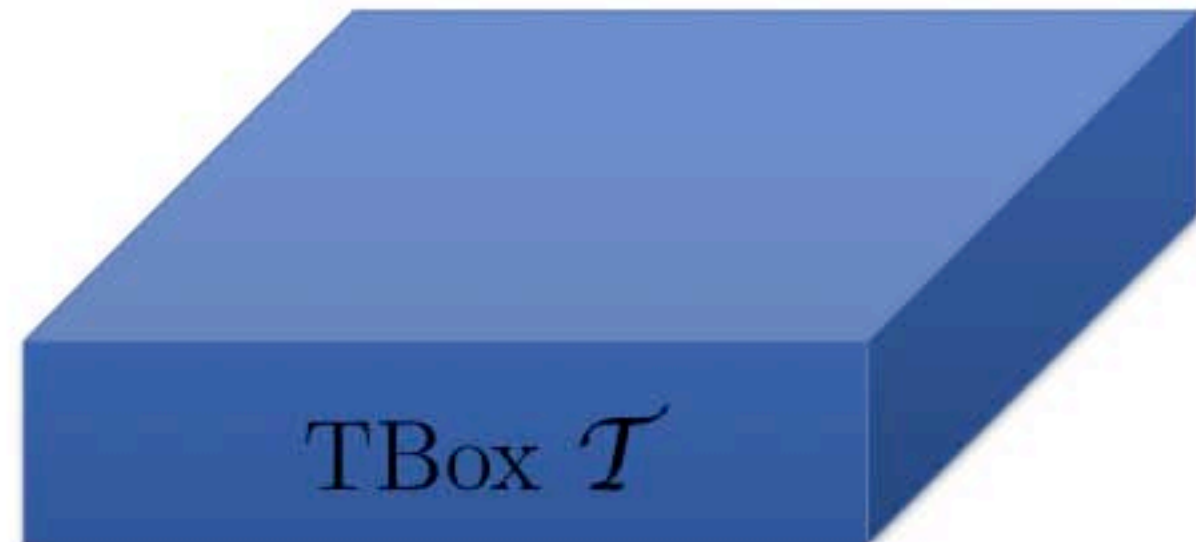
- A *general concept inclusion* (GCI) has the form

$$C \sqsubseteq D$$

where C and D are concept expressions.

- A *TBox* consists of a set of GCIs.

N.B.: Definition of TBox
presumes already known
RBox due to role simplicity
constraints.



ABox

- An *individual assertion* can have any of the following forms
 - $C(\mathbf{a})$, called *concept assertion*,
 - $r(\mathbf{a}, \mathbf{b})$, called *role assertion*,
 - $\neg r(\mathbf{a}, \mathbf{b})$, called *negated role assertion*,
 - $\mathbf{a} \approx \mathbf{b}$, called *equality statement*, or
 - $\mathbf{a} \not\approx \mathbf{b}$, called *inequality statement*.
- An *ABox* consists of a set of individual assertions.



An Example Knowledge Base

RBox \mathcal{R}

$\text{owns} \sqsubseteq \text{caresFor}$

“If somebody owns something, they care for it.”

TBox \mathcal{T}

$\text{Healthy} \sqsubseteq \neg \text{Dead}$

“Healthy beings are not dead.”

$\text{Cat} \sqsubseteq \text{Dead} \sqcup \text{Alive}$

“Every cat is dead or alive.”

$\text{HappyCatOwner} \sqsubseteq \exists \text{owns.Cat} \sqcap \forall \text{caresFor.Healthy}$

“A happy cat owner owns a cat and all beings he cares for are healthy.”

ABox \mathcal{A}

$\text{HappyCatOwner}(\text{schrödinger})$

“Schrödinger is a happy cat owner.”

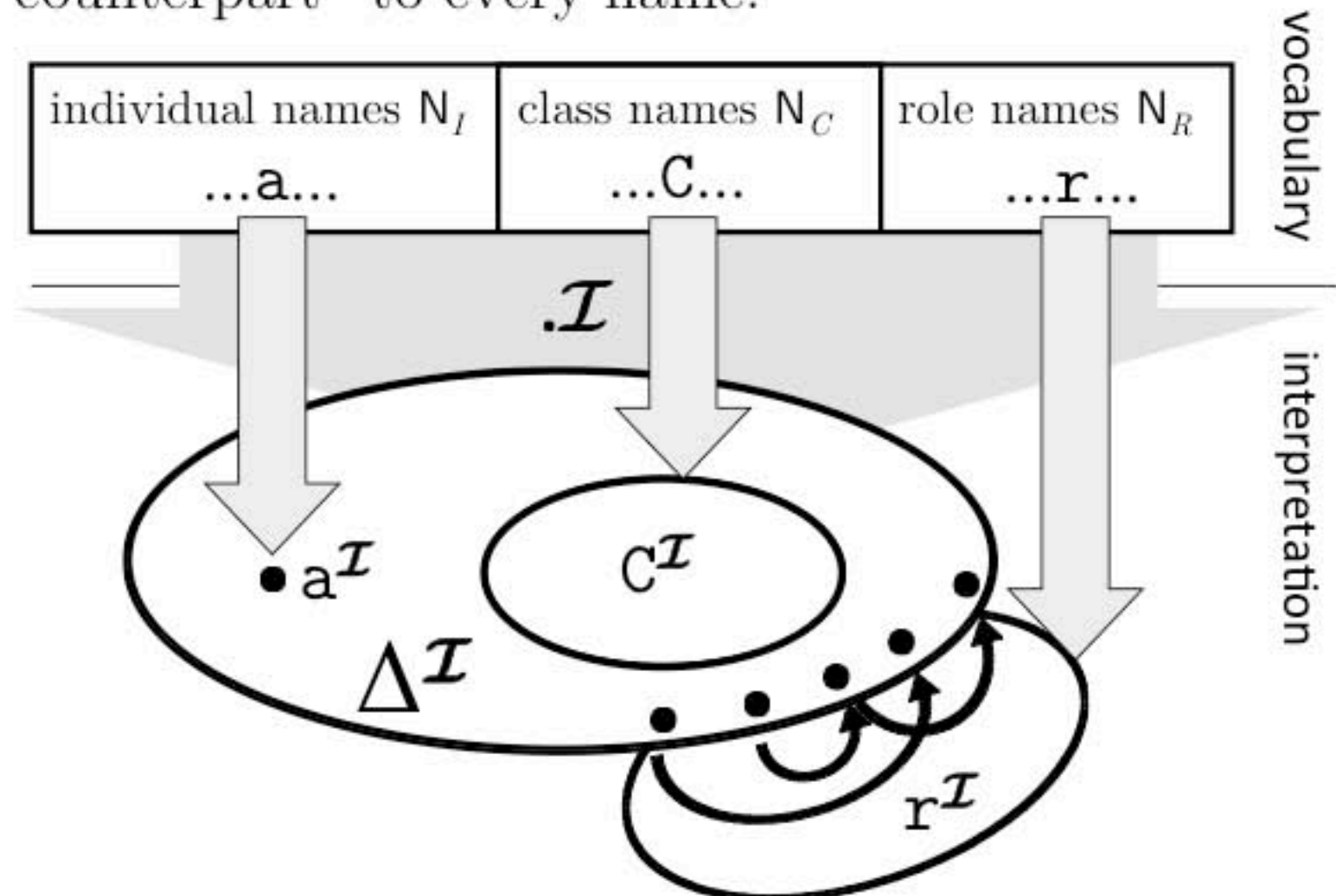
Semantics of Description Logics

*Semantics has wide applications
To relationship-based
altercations,
For semantics unveils
What a statement entails
Depending on interpretations.*



Interpretations

- Semantics for DLs is defined in a **model theoretic** way, i.e. based on „abstract possible worlds“, called **interpretations**.
- A DL interpretation \mathcal{I} fixes a domain set $\Delta^{\mathcal{I}}$ and a mapping $\cdot^{\mathcal{I}}$ associating a „semantic counterpart“ to every name.



N.B.: Different names can be mapped to the same semantic counterpart: no unique name assumption.

N.B.: $\Delta^{\mathcal{I}}$ can be infinite.

Interpretations: an Example

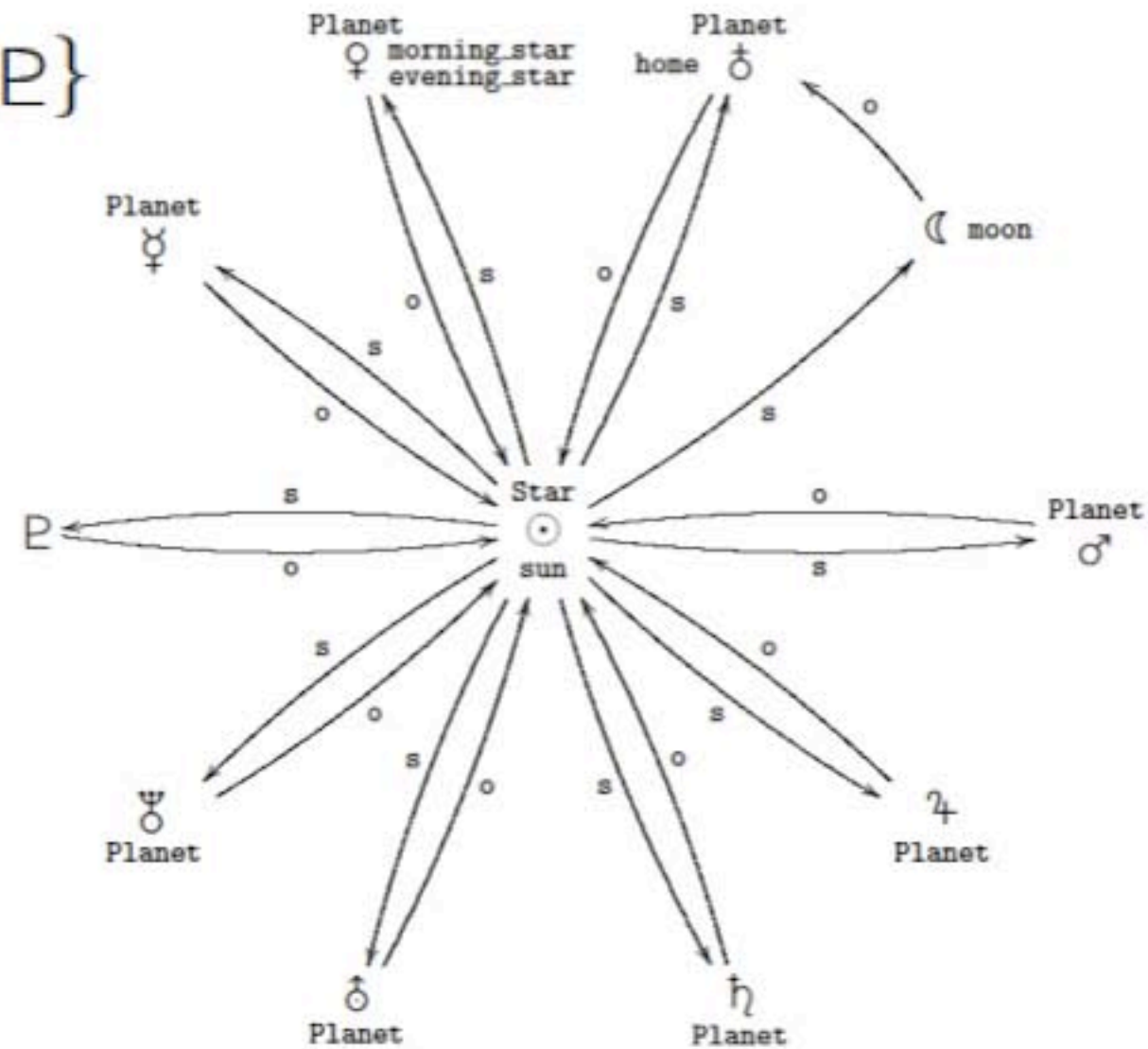
$$N_I = \{\text{sun, morning_star, evening_star, moon, home}\}.$$

$$N_C = \{\text{Planet, Star}\}.$$

$$N_R = \{\text{orbitsAround, shinesOn}\}.$$

$$\Delta^I = \{\odot, \text{♃}, \text{♄}, \text{♅}, \text{♆}, \text{♁}, \text{♂}, \text{♁}, \text{♂}, \text{♁}, \text{♂}, \text{♁}\}$$

$$\begin{aligned} \text{sun}^I &= \odot \\ \text{morning_star}^I &= \text{♃} \\ \text{evening_star}^I &= \text{♄} \\ \text{moon}^I &= \text{♁} \\ \text{home}^I &= \text{♅} \end{aligned}$$

$$\begin{aligned} \text{Planet}^I &= \{\text{♃}, \text{♄}, \text{♅}, \text{♆}, \text{♁}, \text{♂}, \text{♁}, \text{♂}\} \\ \text{Star}^I &= \{\odot\} \\ \text{orbitsAround}^I &= \{ \langle \text{♃}, \odot \rangle, \langle \text{♄}, \odot \rangle, \langle \text{♅}, \odot \rangle, \langle \text{♆}, \odot \rangle, \langle \text{♁}, \odot \rangle, \\ &\quad \langle \text{♂}, \odot \rangle, \langle \text{♁}, \odot \rangle, \langle \text{♂}, \odot \rangle, \langle \text{♁}, \odot \rangle, \langle \text{♂}, \odot \rangle, \langle \text{♁}, \odot \rangle, \langle \text{♂}, \odot \rangle \} \\ \text{shinesOn}^I &= \{ \langle \odot, \text{♃} \rangle, \langle \odot, \text{♄} \rangle, \langle \odot, \text{♅} \rangle, \langle \odot, \text{♆} \rangle, \langle \odot, \text{♁} \rangle, \langle \odot, \text{♂} \rangle, \\ &\quad \langle \odot, \text{♁} \rangle, \langle \odot, \text{♂} \rangle, \langle \odot, \text{♁} \rangle, \langle \odot, \text{♂} \rangle, \langle \odot, \text{♁} \rangle \} \end{aligned}$$


Interpretation of Concept Expressions

- Given an interpretation, we can determine the semantic counterparts for concept expressions along the following inductive definitions:
- mapping is extended to complex class expressions:

$$\top^I = \Delta^I$$

$$\perp^I = \{\}$$

$$\{a_1, \dots, a_n\}^I = \{a_1^I, \dots, a_n^I\}$$

$$(\neg C)^I = \Delta^I \setminus C^I$$

$$(C \sqcap D)^I = C^I \cap D^I$$

$$(C \sqcup D)^I = C^I \cup D^I$$

$$\exists r.C = \{ x \mid \exists y. (x,y) \in r^I \wedge y \in C^I \}$$

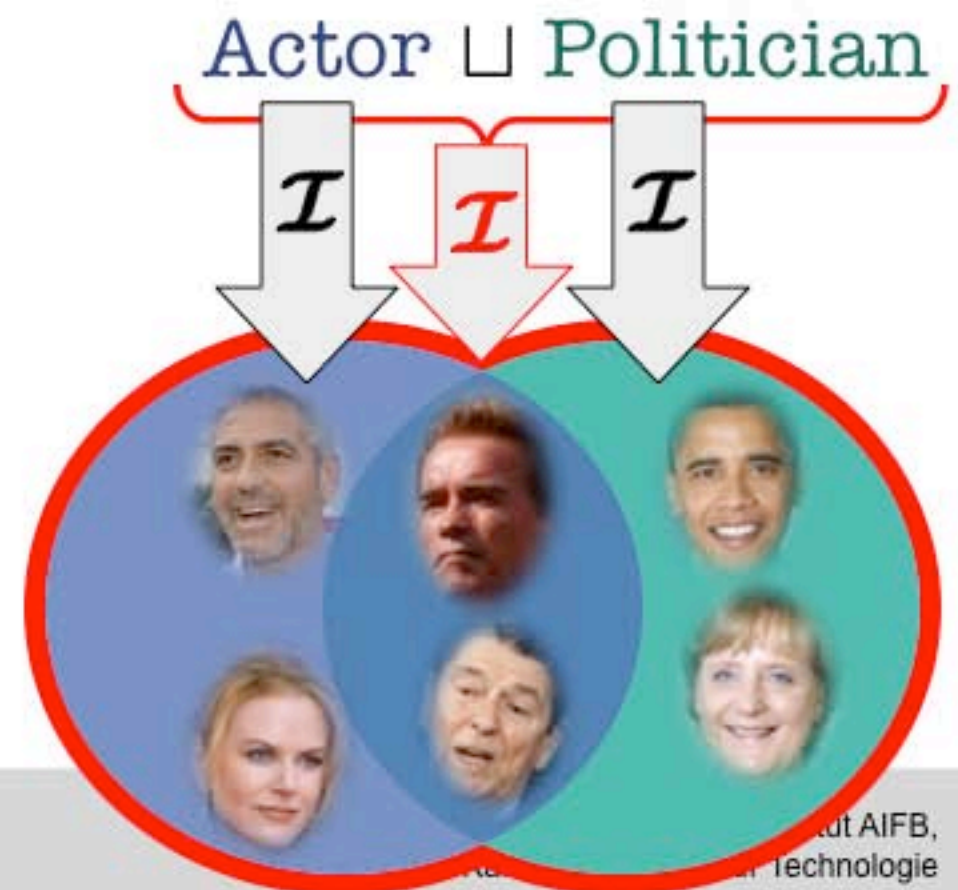
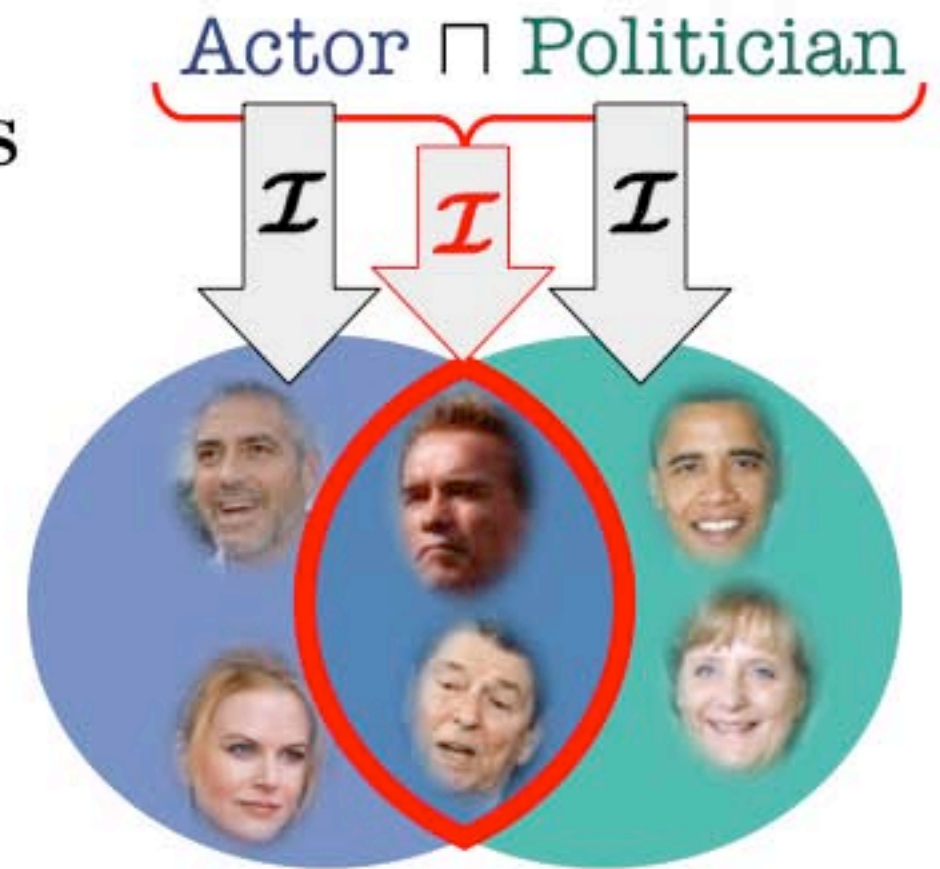
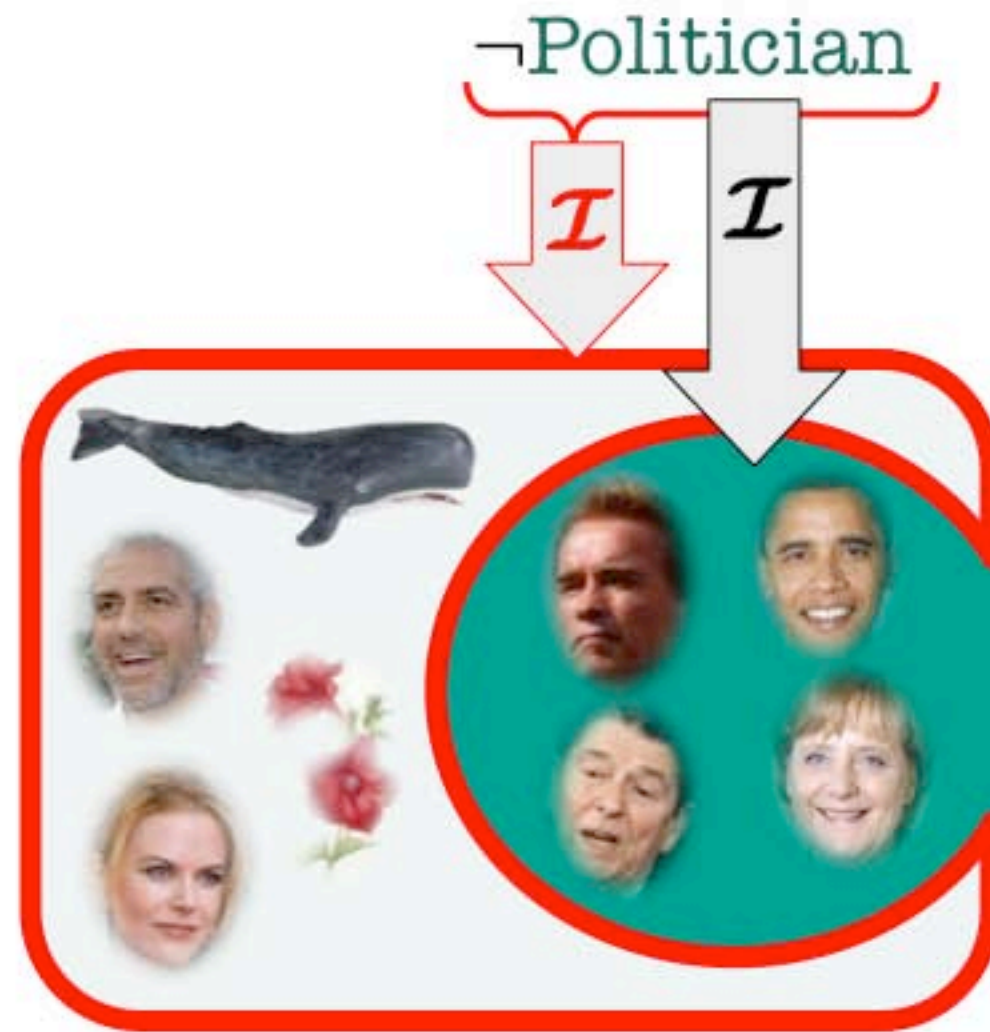
$$\forall r.C = \{ x \mid \forall y. (x,y) \in r^I \rightarrow y \in C^I \}$$

$$\exists s.\mathbf{Self} = \{ x \mid (x,x) \in s^I \}$$

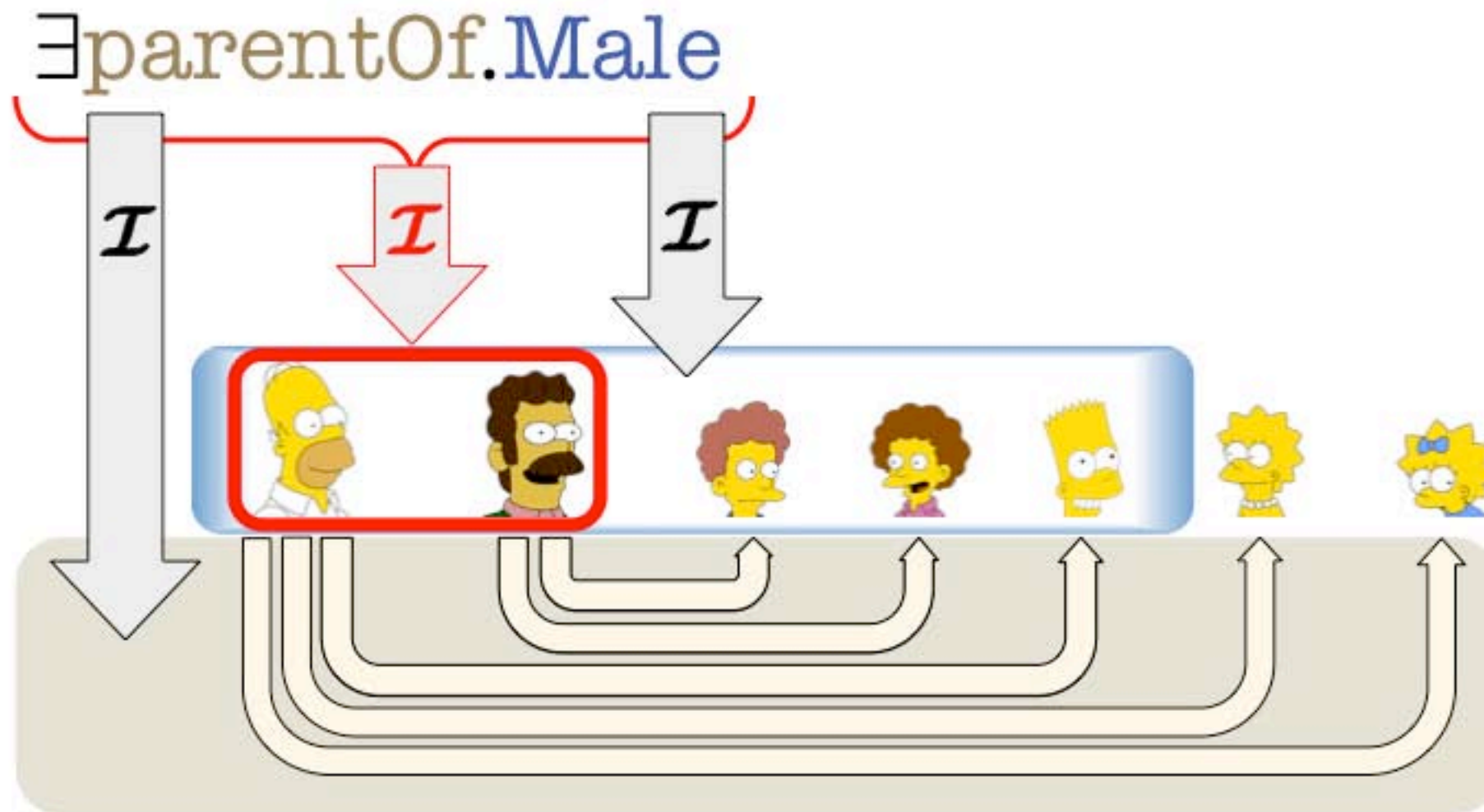
$$\geq ns.C = \{ x \mid \#\{ y \mid (x,y) \in s^I \wedge y \in C^I \} \geq n \}$$

$$\leq ns.C = \{ x \mid \#\{ y \mid (x,y) \in s^I \wedge y \in C^I \} \leq n \}$$

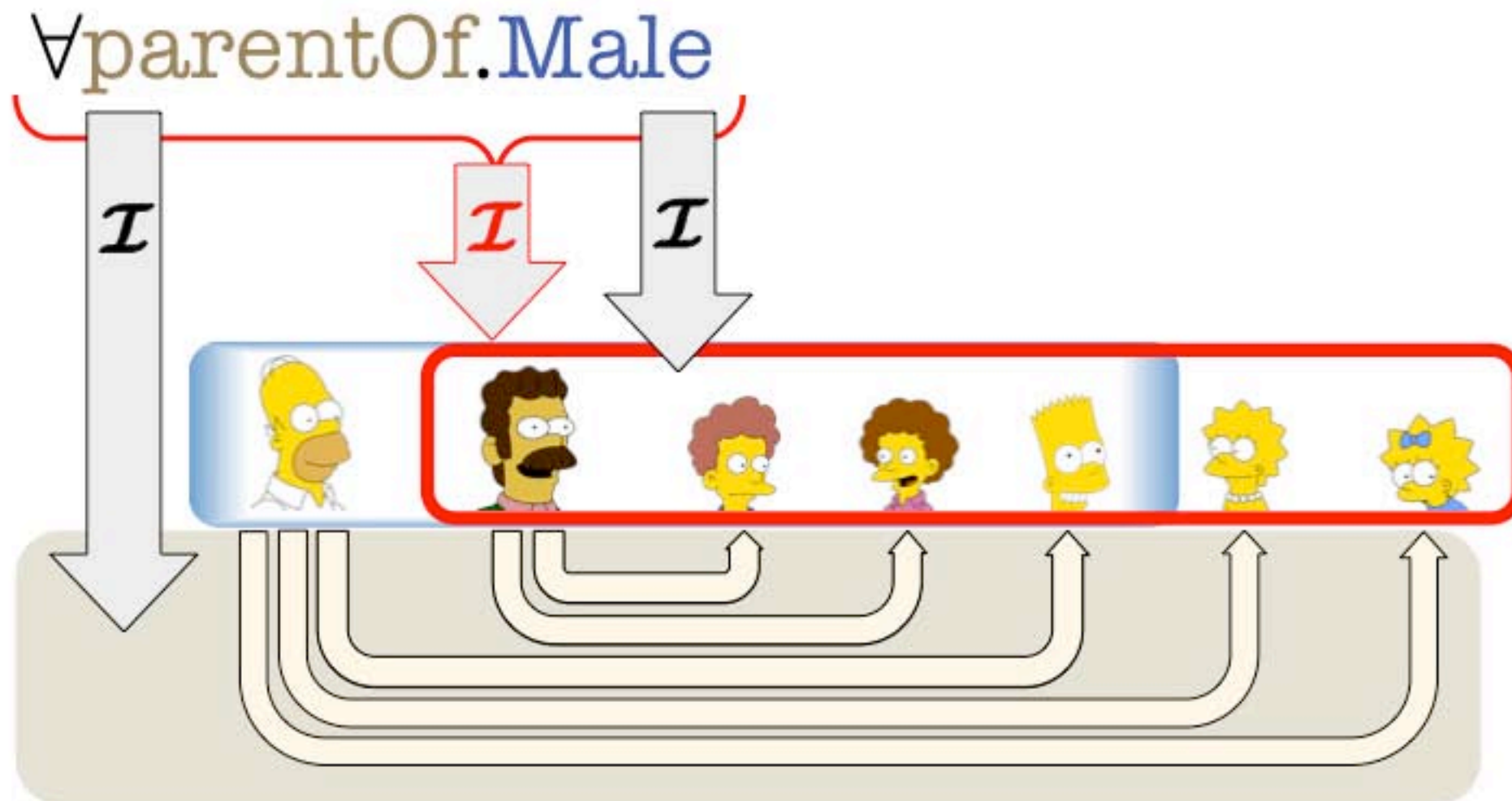
Boolean Concept Expressions



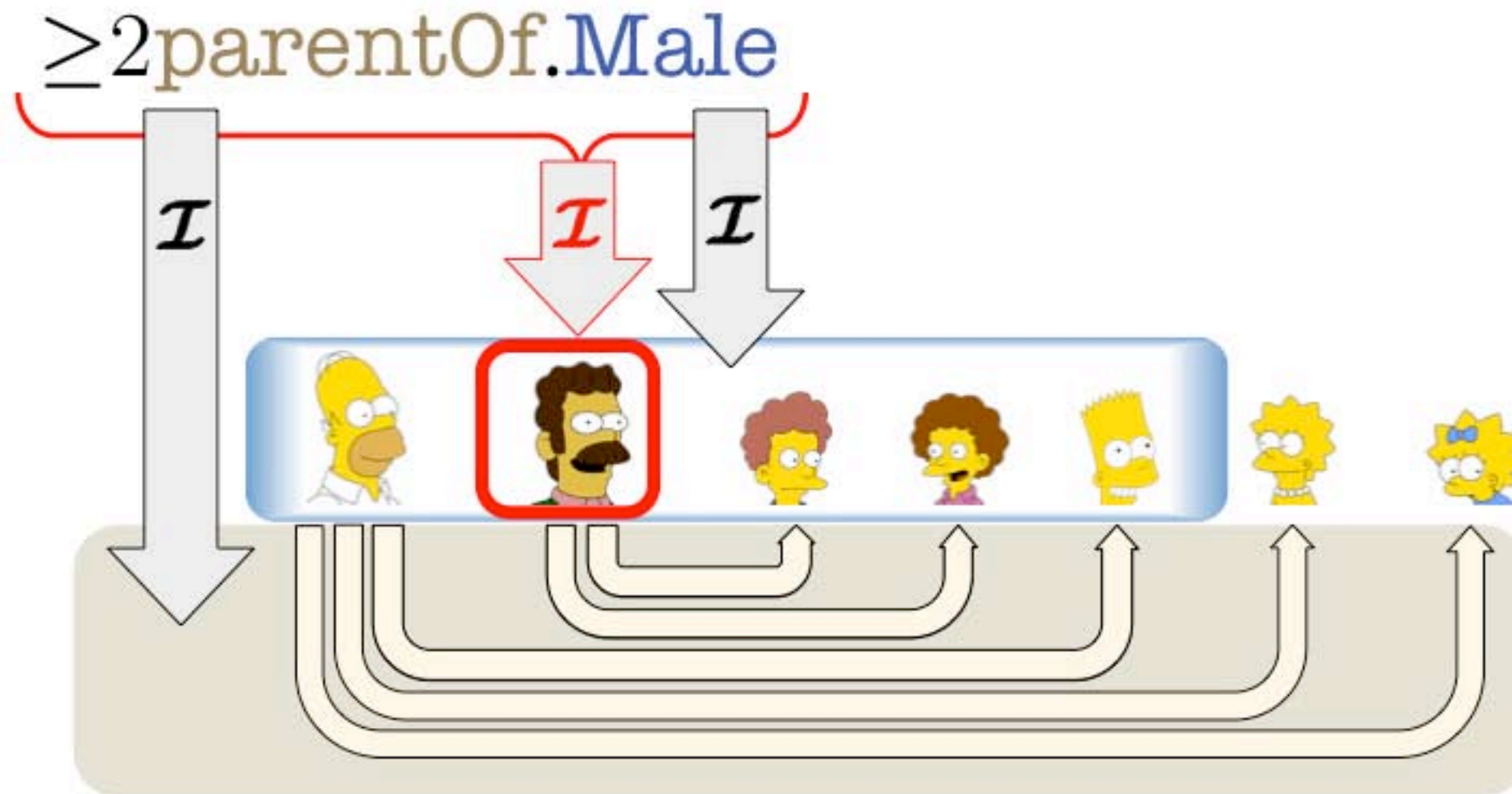
Existential Role Restrictions



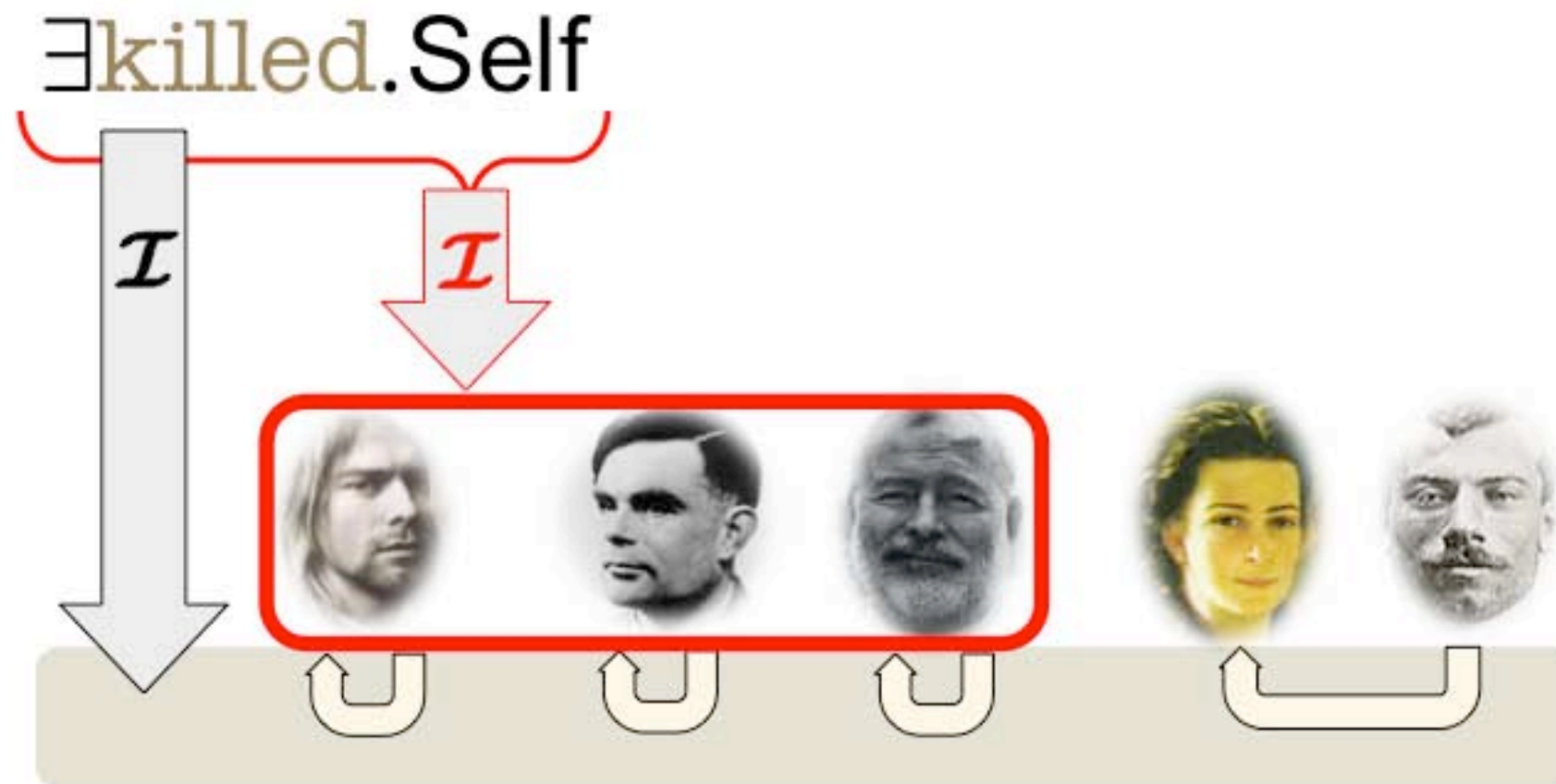
Universal Role Restrictions



Qualified Number Restrictions



Self-Restrictions



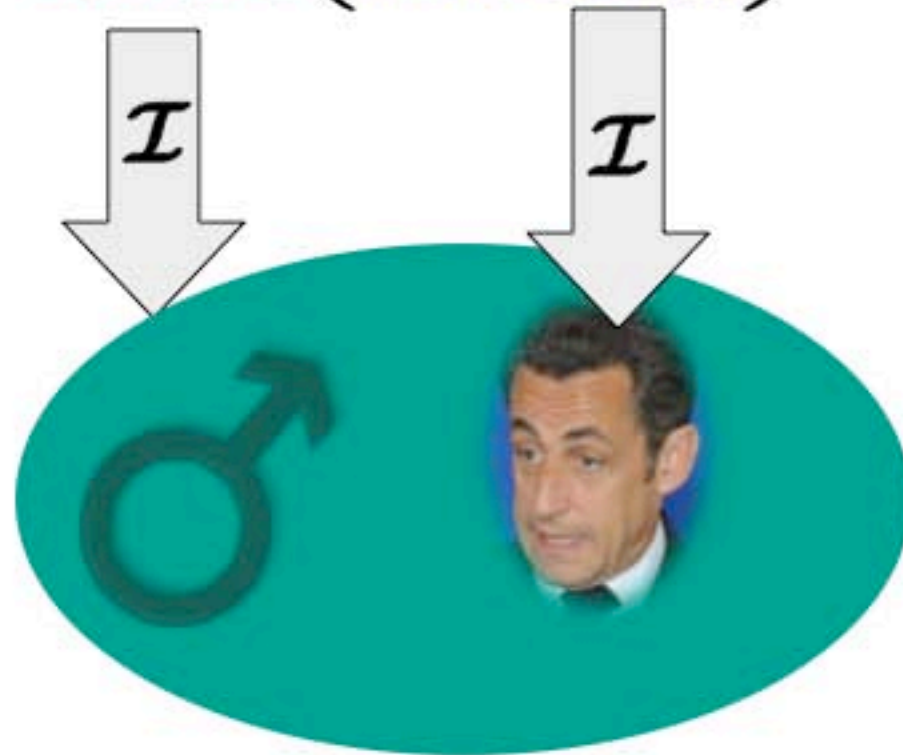
Semantics of Axioms

Given a way to determine a semantic counterpart for all expressions, we now define the criteria for checking if an interpretation \mathcal{I} satisfies an axiom alpha α (written: $\mathcal{I} \models \alpha$).

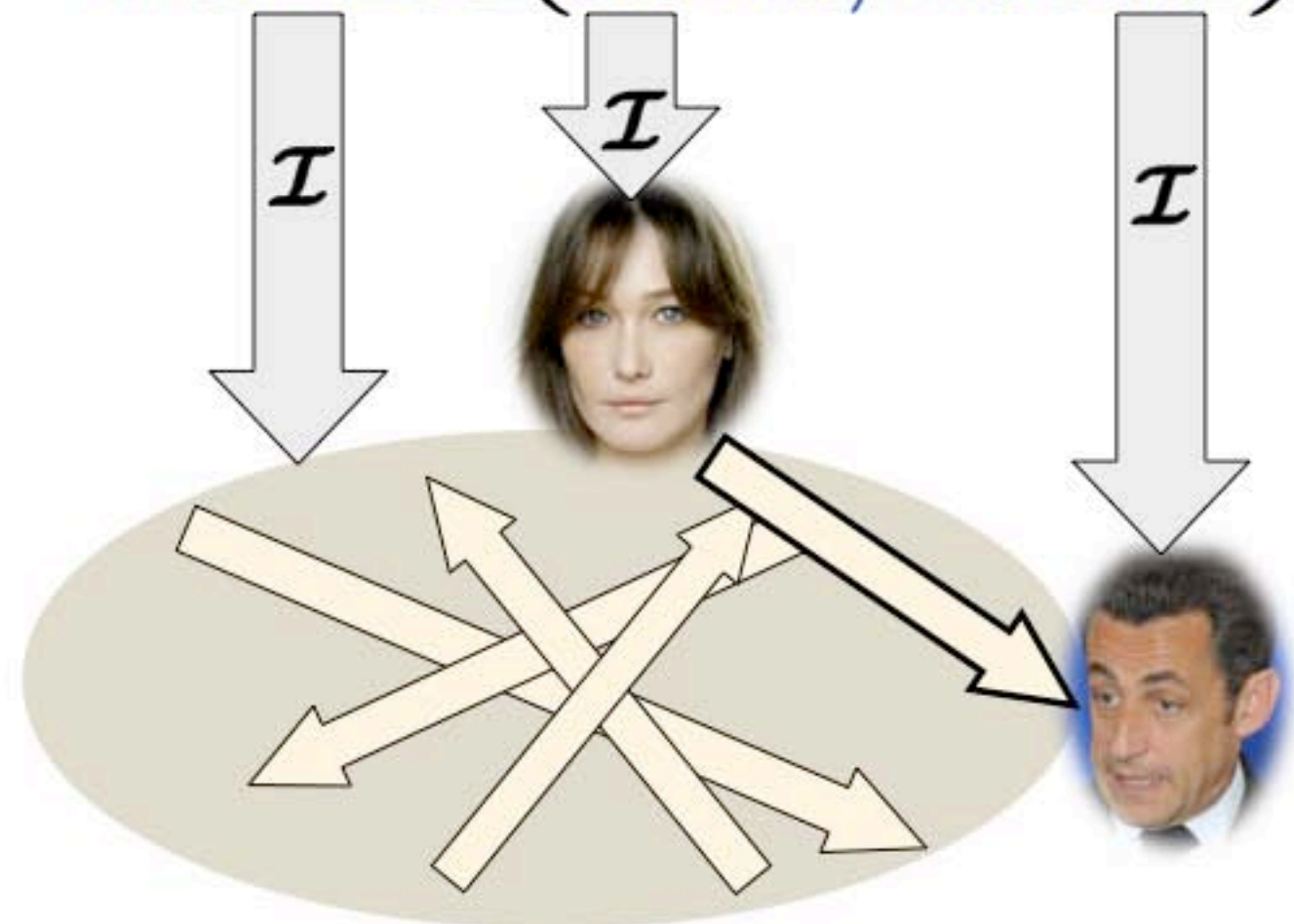
- $\mathcal{I} \models r_1 \circ \dots \circ r_n \sqsubseteq r$ if $r_1^{\mathcal{I}} \circ \dots \circ r_n^{\mathcal{I}} \subseteq r^{\mathcal{I}}$
- $\mathcal{I} \models \text{Dis}(\mathbf{s}_1, \mathbf{s}_2)$ if $\mathbf{s}_1^{\mathcal{I}} \cap \mathbf{s}_2^{\mathcal{I}} = \{\}$
- $\mathcal{I} \models C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I} \models C(\mathbf{a})$ if $\mathbf{a}^{\mathcal{I}} \in D^{\mathcal{I}}$
- $\mathcal{I} \models r(\mathbf{a}, \mathbf{b})$ if $(\mathbf{a}^{\mathcal{I}}, \mathbf{b}^{\mathcal{I}}) \in r^{\mathcal{I}}$
- $\mathcal{I} \models \neg_r(\mathbf{a}, \mathbf{b})$ if $(\mathbf{a}^{\mathcal{I}}, \mathbf{b}^{\mathcal{I}}) \notin r^{\mathcal{I}}$
- $\mathcal{I} \models \mathbf{a} \approx \mathbf{b}$ if $\mathbf{a}^{\mathcal{I}} = \mathbf{b}^{\mathcal{I}}$
- $\mathcal{I} \models \mathbf{a} \not\approx \mathbf{b}$ if $\mathbf{a}^{\mathcal{I}} \neq \mathbf{b}^{\mathcal{I}}$

Concept and Role Membership

Male(nicolas)

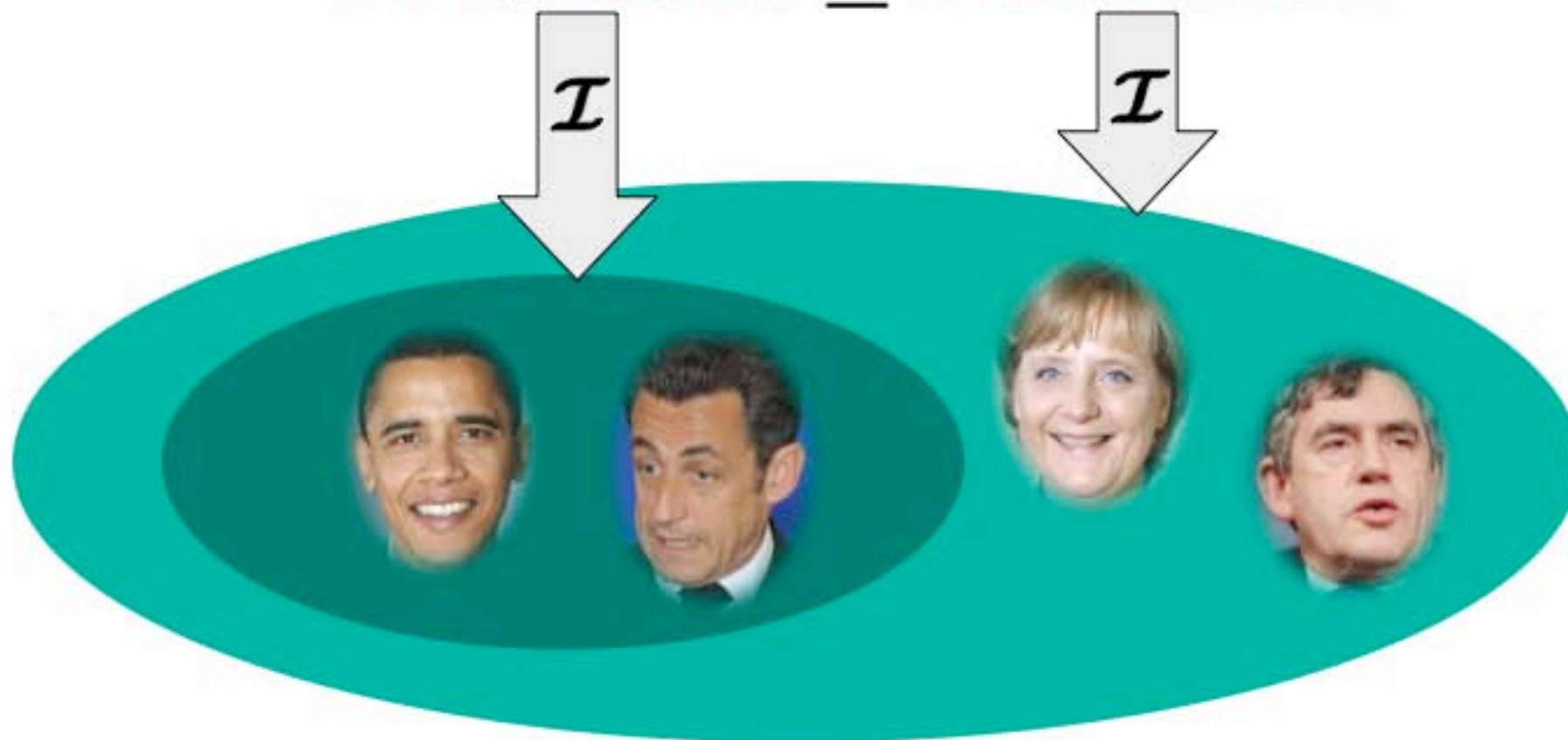


married(carla,nicolas)



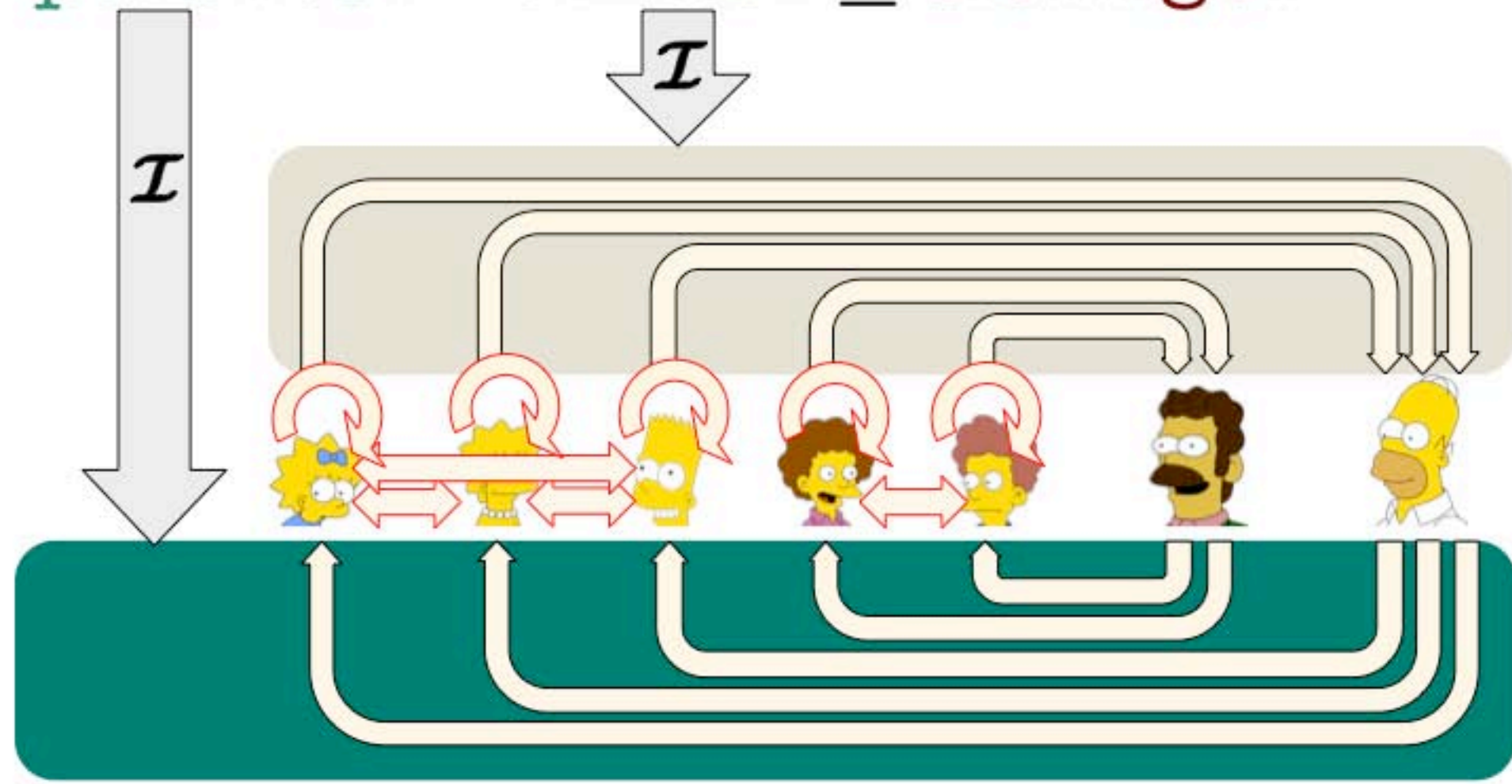
General Inclusion Axioms

President \sqsubseteq Politician



Role Inclusion Axioms

parentOf \circ childOf \sqsubseteq siblingOf



(Un)Satisfiability of Knowledge Bases

- A KB is *satisfiable* (also: *consistent*) if there exists an interpretation that satisfies all its axioms (a *model* of the KB). Otherwise it is *unsatisfiable* (also: *inconsistent* or *contradictory*).
- Is the following KB satisfiable?

$\text{Reindeer} \sqcap \exists \text{hasNose.Red}(\text{rudolph})$	$\text{Reindeer} \sqsubseteq \text{Mammal}$
$\forall \text{worksFor} \neg . (\neg \text{Reindeer} \sqcup \text{Flies})(\text{santa})$	$\text{Mammal} \sqcap \text{Flies} \sqsubseteq \text{Bat}$
$\text{worksFor}(\text{rudolph}, \text{santa})$	$\text{Bat} \sqsubseteq \forall \text{worksFor} . \{\text{batman}\}$
$\text{santa} \neq \text{batman}$	



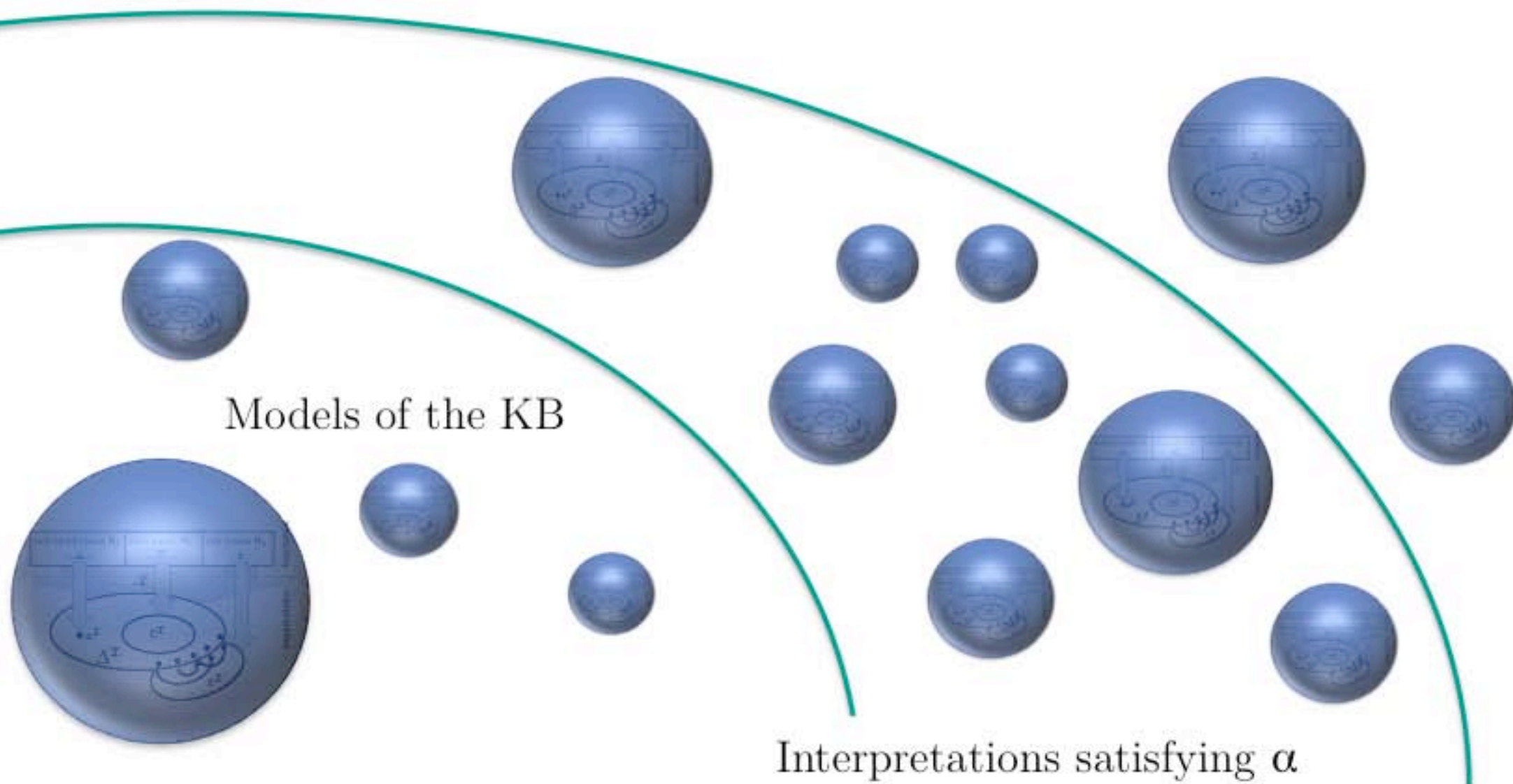
Sebastian Rudolph


 Foundations of Descriptive Logics and OWL
 Reasoning Web Summer School 2011

 Institut AIFB,
 Karlsruher Institut für Technologie

Entailment of Axioms

- A KB entails an axiom α if the axiom α is satisfied by every model of the knowledge base.



Decidability of DLs

DLs are *decidable*, i.e. there exists an algorithm that

- takes a knowledge base and an axiom as input,
- terminates after finite time,
- provides as output the correct answer to the question whether the KB entails the axiom.



Semantics via Translation into FOL

Since DLs can be seen as fragments of FOL, we can alternatively define the semantics by providing a translation of DL axioms into FOL formulae.

$$\begin{aligned}
 \tau(r_1 \circ \dots \circ r_n \sqsubseteq r) &= \forall x_0 \dots x_n (\bigwedge_{1 \leq i \leq n} \tau_{\mathbf{R}}(r_i, x_{i-1}, x_i)) \rightarrow \tau_{\mathbf{R}}(r, x_0, x_n) \\
 \tau(\text{Dis}(r, r')) &= \forall x_0 x_1 (\tau_{\mathbf{R}}(r, x_0, x_1) \rightarrow \neg \tau_{\mathbf{R}}(r', x_0, x_1)) \\
 \tau(C \sqsubseteq D) &= \forall x_0 (\tau_{\mathbf{C}}(C, x_0) \rightarrow \tau_{\mathbf{C}}(D, x_0)) \\
 \tau(C(\mathbf{a})) &= \tau_{\mathbf{C}}(C, x_0)[x_0/\mathbf{a}] \\
 \tau(r(\mathbf{a}, \mathbf{b})) &= \tau_{\mathbf{R}}(r, x_0, x_1)[x_0/\mathbf{a}][x_1/\mathbf{b}] \\
 \tau(\neg r(\mathbf{a}, \mathbf{b})) &= \neg \tau(r(\mathbf{a}, \mathbf{b})) \\
 \tau(a \approx b) &= a = b \\
 \tau(a \not\approx b) &= \neg(a = b)
 \end{aligned}$$

Concept/role expressions are translated into formulae with one/two free variable(s).

$$\tau_{\mathbf{C}}(\mathbf{A}, x_i) = \mathbf{A}(x_i)$$

$$\tau_{\mathbf{R}}(u, x_i, x_j) = \mathbf{true}$$

$$\tau_{\mathbf{C}}(\top, x_i) = \mathbf{true}$$

$$\tau_{\mathbf{R}}(\mathbf{r}, x_i, x_j) = \mathbf{r}(x_i, x_j)$$

$$\tau_{\mathbf{C}}(\perp, x_i) = \mathbf{false}$$

$$\tau_{\mathbf{R}}(\mathbf{r}^-, x_i, x_j) = \mathbf{r}(x_j, x_i)$$

$$\tau_{\mathbf{C}}(\{\mathbf{a}_1, \dots, \mathbf{a}_n\}, x_i) = \bigvee_{1 \leq j \leq n} x_i = \mathbf{a}_j$$

$$\tau_{\mathbf{C}}(\neg C, x_i) = \neg \tau_{\mathbf{C}}(C, x_i)$$

$$\tau_{\mathbf{C}}(C \sqcap D, x_i) = \tau_{\mathbf{C}}(C, x_i) \wedge \tau_{\mathbf{C}}(D, x_i)$$

$$\tau_{\mathbf{C}}(C \sqcup D, x_i) = \tau_{\mathbf{C}}(C, x_i) \vee \tau_{\mathbf{C}}(D, x_i)$$

$$\tau_{\mathbf{C}}(\exists r.C, x_i) = \exists x_{i+1}. (\tau_{\mathbf{R}}(r, x_i, x_{i+1}) \wedge \tau_{\mathbf{C}}(C, x_{i+1}))$$

$$\tau_{\mathbf{C}}(\forall r.C, x_i) = \forall x_{i+1}. (\tau_{\mathbf{R}}(r, x_i, x_{i+1}) \rightarrow \tau_{\mathbf{C}}(C, x_{i+1}))$$

$$\tau_{\mathbf{C}}(\exists r.\mathbf{Self}, x_i) = \tau_{\mathbf{R}}(r, x_i, x_i)$$

$$\tau_{\mathbf{C}}(\geq nr.C, x_i) = \exists x_{i+1} \dots x_{i+n}. \left(\bigwedge_{i+1 \leq j < k \leq i+n} (x_j \neq x_k) \right. \\ \left. \wedge \bigwedge_{i+1 \leq j \leq i+n} (\tau_{\mathbf{R}}(r, x_i, x_j) \wedge \tau_{\mathbf{C}}(C, x_j)) \right)$$

$$\tau_{\mathbf{C}}(\leq nr.C, x_i) = \neg \tau_{\mathbf{C}}(\geq (n+1)r.C, x_i)$$

Description Logics Nomenclature

What's in a name? That which we call, say, SHIQ,

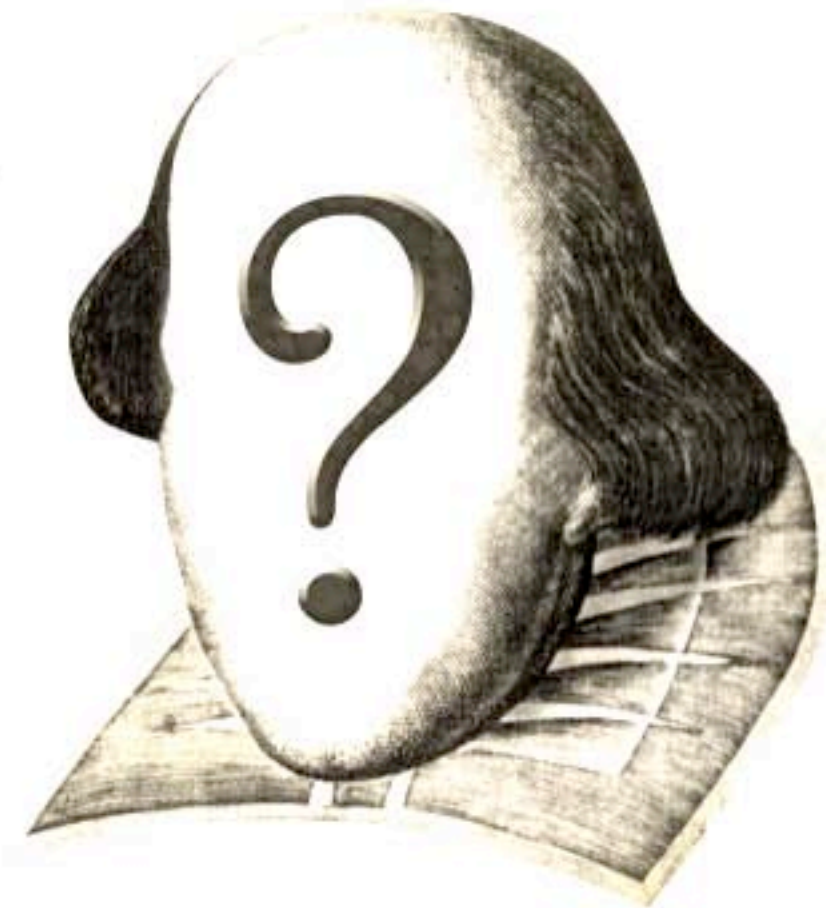
By any other name would do the trick.

While DL names might leave the novice SHOQed,

Some principles of ALCHemy unlocked

Enable understanding in a minute:

Though it be madness, yet there's method in it.



Naming Scheme for Expressive DLs

$$((ALC | S)[H] | SR)[O][I][F | N | Q]$$

- S subsumes ALC
- SR subsumes S , SH , ALC and $ALCH$
- N makes F obsolete
- Q makes N (and F) obsolete

We treat here the very expressive description logic $SRQIQ$ which subsumes all the other ones in this scheme.

DL Syntax – Overview

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	Closed class	$\{i_1, \dots, i_n\}$
R	Self	$\exists r.\text{Self}$

Roles		
I	Atomic	r
	Inverse	r^-

Ontology (=Knowledge Base)

Concept Axioms (TBox)	
Subclass	$C \sqsubseteq D$
Equivalent	$C \equiv D$

Role Axioms (RBox)		
SH	Subrole	$r \sqsubseteq s$
	Transitivity	$\text{Trans}(r)$
SR	Role Chain	$r \circ r' \sqsubseteq s$
	R. Disjointness	$\text{Disj}(s, r)$

Assertional Axioms (ABox)	
Instance	$C(a)$
Role	$r(a, b)$
Same	$a \approx b$
Different	$a \not\approx b$

Equivalences, Emulation, Normalization

*Don't give told consequences lip,
Nor 'bout equivalences quip,
'Cause often it's the formal norm
That statements be in normal form.*



Concept Equivalences

Two concept expressions C and D are called *equivalent* (written: $C \equiv D$), if for **every** interpretation \mathcal{I} holds $C^{\mathcal{I}} = D^{\mathcal{I}}$.

$$\begin{array}{ll}
 C \sqcap D \equiv D \sqcap C & C \sqcup D \equiv D \sqcup C \\
 (C \sqcap D) \sqcap E \equiv C \sqcap (D \sqcap E) & (C \sqcup D) \sqcup E \equiv C \sqcup (D \sqcup E) \\
 C \sqcap C \equiv C & C \sqcup C \equiv C
 \end{array}$$

$$\begin{array}{ll}
 (C \sqcup D) \sqcap E \equiv (C \sqcap E) \sqcup (D \sqcap E) & (C \sqcup D) \sqcap C \equiv C \\
 (C \sqcap D) \sqcup E \equiv (C \sqcup E) \sqcap (D \sqcup E) & (C \sqcap D) \sqcup C \equiv C
 \end{array}$$

$$\begin{array}{lll}
 \neg\neg C \equiv C & \neg\exists r.C \equiv \forall r.\neg C & \geq 0r.C \equiv \top \\
 \neg(C \sqcap D) \equiv \neg D \sqcup \neg C & \neg\forall r.C \equiv \exists r.\neg C & \geq 1r.C \equiv \exists r.C \\
 \neg(C \sqcup D) \equiv \neg D \sqcap \neg C & \neg\leq nr.C \equiv \geq (n+1)r.C & \leq 0r.C \equiv \forall r.\neg C \\
 \neg\geq (n+1)r.C \equiv \leq nr.C & &
 \end{array}$$

Negation Normal Form

- Iterated rewriting of concept expressions along the mentioned equivalences allows to convert every concept expression into one with negation only in front of concept names, nominal concepts and Self-restrictions.

$$nnf(C) := C \text{ if } C \in \{A, \neg A, \{a_1, \dots, a_n\}, \neg\{a_1, \dots, a_n\}, \exists r.\text{Self}, \neg\exists r.\text{Self}, \top, \perp\}$$

$$nnf(\neg\neg C) := nnf(C)$$

$$nnf(\neg\top) := \perp$$

$$nnf(\neg\perp) := \top$$

$$nnf(C \sqcap D) := nnf(C) \sqcap nnf(D) \quad nnf(\neg(C \sqcap D)) := nnf(\neg C) \sqcup nnf(\neg D)$$

$$nnf(C \sqcup D) := nnf(C) \sqcup nnf(D) \quad nnf(\neg(C \sqcup D)) := nnf(\neg C) \sqcap nnf(\neg D)$$

$$nnf(\forall r.C) := \forall r.nnf(C) \quad nnf(\neg\forall r.C) := \exists r.nnf(\neg C)$$

$$nnf(\exists r.C) := \exists r.nnf(C) \quad nnf(\neg\exists r.C) := \forall r.nnf(\neg C)$$

$$nnf(\leq_n r.C) := \leq_n r.nnf(C) \quad nnf(\neg\leq_n r.C) := \geq_{(n+1)} r.nnf(C)$$

$$nnf(\geq_n r.C) := \geq_n r.nnf(C) \quad nnf(\neg\geq_n r.C) := \leq_{(n-1)} r.nnf(C)$$

Axiom and KB Equivalences

- Lloyd-Topor equivalences

$$\{A \sqcup B \sqsubseteq C\} \iff \{A \sqsubseteq C, B \sqsubseteq C\}$$

$$\{A \sqsubseteq B \sqcap C\} \iff \{A \sqsubseteq B, A \sqsubseteq C\}$$

- turning GCIs into universally valid concept descriptions

$$C \sqsubseteq D \iff \top \sqsubseteq \neg C \sqcup D$$

- internalisation of ABox into TBox

$$C(a) \iff \{a\} \sqsubseteq C$$

$$r(a, b) \iff \{a\} \sqsubseteq \exists r. \{b\}$$

$$\neg r(a, b) \iff \{a\} \sqsubseteq \neg \exists r. \{b\}$$

$$a \approx b \iff \{a\} \sqsubseteq \{b\}$$

$$a \not\approx b \iff \{a\} \sqsubseteq \neg \{b\}$$

Emulation

- Sometimes the knowledge base is required to be in some specific form which cannot be obtained by equivalent transformations alone. In that cases, one can try to obtain a KB that is equivalent “up to additional vocabulary” (called *fresh names*).

Example:

- ABox is *extensionally reduced* if all concept assertions are contain concept names only.
- Any KB can be turned into one with extensionally reduced ABox by repeating the following procedure:
 - Pick a concept assertion $C(\mathbf{a})$ where C is not a concept name
 - remove $C(\mathbf{a})$ from the Abox and add $\mathbf{A}(\mathbf{a})$ instead, where \mathbf{A} is not used elsewhere in the KB
 - add $\mathbf{A} \sqsubseteq C$ to the TBox

Emulation

A knowledge base \mathcal{KB}' *emulates* a knowledge base \mathcal{KB} if two conditions are satisfied:

- Every model of \mathcal{KB}' is a model of \mathcal{KB} , formally: given an interpretation \mathcal{I} , we have that $\mathcal{I} \models \mathcal{KB}'$ implies $\mathcal{I} \models \mathcal{KB}$.
- For every model \mathcal{I} of \mathcal{KB} there is a model \mathcal{I}' of \mathcal{KB}' that has the same domain as \mathcal{I} , and coincides with \mathcal{I} on the vocabulary used in \mathcal{KB} .

Using emulation allows to model many things that are not directly expressible in the used DL.

Example: “ $A \sqsubseteq B$ holds or $C \sqsubseteq D$ holds“ can be emulated by

$$\top \sqsubseteq \exists r. \{o\} \quad \{o\} \sqsubseteq \forall r^-. (\neg A \sqcup B) \sqcup \forall r^-. (\neg C \sqcup D)$$

where o is a fresh individual name and r a fresh role name.

For more (and more intricate) examples, see also Pascal’s lecture.

Some Exercises



Institut für Angewandte Informatik und Formale Beschreibungsverfahren



RW2011
Galway City, Ireland
August 23 - August 27

“Computing“ Extensions

- $N_I = \{\text{zero}\}$.
- $N_C = \{\text{Prime}, \text{Positive}\}$.
- $N_R = \{\text{hasSuccessor}, \text{lessThan}, \text{multipleOf}\}$.

Now, we define \mathcal{I} as follows: let $\Delta^{\mathcal{I}} = \mathbb{N} = \{0, 1, 2, \dots\}$, i.e., the set of all natural numbers including zero. Furthermore, we let $\text{zero}^{\mathcal{I}} = 0$, as well as $\text{Prime}^{\mathcal{I}} = \{n \mid n \text{ is a prime number}\}$ and $\text{Positive}^{\mathcal{I}} = \{n \mid n > 0\}$. For the roles, we define

- $\text{hasSuccessor}^{\mathcal{I}} = \{\langle n, n+1 \rangle \mid n \in \mathbb{N}\}$
- $\text{lessThan}^{\mathcal{I}} = \{\langle n, n' \rangle \mid n < n', n, n' \in \mathbb{N}\}$
- $\text{multipleOf}^{\mathcal{I}} = \{\langle n, n' \rangle \mid \exists k. n = k \cdot n', n, n', k \in \mathbb{N}\}$

Exercise 1. Describe – both verbally and formally – the extension of the following concepts with respect to the interpretation \mathcal{I} defined in Example 16:

- (a) $\forall \text{hasSuccessor}^- . \text{Positive}$
- (b) $\exists \text{multipleOf} . \text{Self}$
- (c) $\exists \text{multipleOf} . \exists \text{hasSuccessor}^- . \exists \text{hasSuccessor}^- . \{\text{zero}\}$
- (d) $\geq 10 \text{ lessThan}^- . \text{Prime}$
- (e) $\neg \text{Prime} \sqcap \leq 2 \text{ multipleOf} . \top$
- (f) $\exists \text{lessThan} . \text{Prime}$
- (g) $\forall \text{multipleOf} . (\exists \text{hasSuccessor}^- . \{\text{zero}\})$
 $\sqcup \exists \text{multipleOf} . \exists \text{hasSuccessor}^- . \exists \text{hasSuccessor}^- . \{\text{zero}\})$

Determining Axiom Satisfaction

- $N_I = \{\text{zero}\}$.
- $N_C = \{\text{Prime}, \text{Positive}\}$.
- $N_R = \{\text{hasSuccessor}, \text{lessThan}, \text{multipleOf}\}$.

Now, we define \mathcal{I} as follows: let $\Delta^{\mathcal{I}} = \mathbb{N} = \{0, 1, 2, \dots\}$, i.e., the set of all natural numbers including zero. Furthermore, we let $\text{zero}^{\mathcal{I}} = 0$, as well as $\text{Prime}^{\mathcal{I}} = \{n \mid n \text{ is a prime number}\}$ and $\text{Positive}^{\mathcal{I}} = \{n \mid n > 0\}$. For the roles, we define

- $\text{hasSuccessor}^{\mathcal{I}} = \{\langle n, n+1 \rangle \mid n \in \mathbb{N}\}$
- $\text{lessThan}^{\mathcal{I}} = \{\langle n, n' \rangle \mid n < n', n, n' \in \mathbb{N}\}$
- $\text{multipleOf}^{\mathcal{I}} = \{\langle n, n' \rangle \mid \exists k. n = k \cdot n', n, n', k \in \mathbb{N}\}$

Exercise 2. *Decide whether the following axioms are satisfied by the interpretation \mathcal{I} from Example 16.*

- (a) $\text{hasSuccessor} \sqsubseteq \text{lessThan}$
- (b) $\exists \text{hasSuccessor}^- . \exists \text{hasSuccessor}^- . \{\text{zero}\} \sqsubseteq \text{Prime}$
- (c) $\top \sqsubseteq \forall \text{multipleOf}^- . \{\text{zero}\}$
- (d) $\text{Dis}(\text{divisibleBy}, \text{lessThan}^-)$
- (e) $\text{multipleOf} \circ \text{multipleOf} \sqsubseteq \text{multipleOf}$
- (f) $\top \sqsubseteq \leq 1 \text{hasSuccessor} . \text{Positive}$
- (g) $\text{zero} \not\approx \text{zero}$
- (h) $\leq 1 \text{multipleOf}^- . \top(\text{zero})$
- (i) $\top \sqsubseteq \forall \text{lessThan} . \exists \text{lessThan} . (\text{Prime} \sqcap \exists \text{hasSuccessor} . \exists \text{hasSuccessor} . \text{Prime})$

(Non-)Concept Equivalences

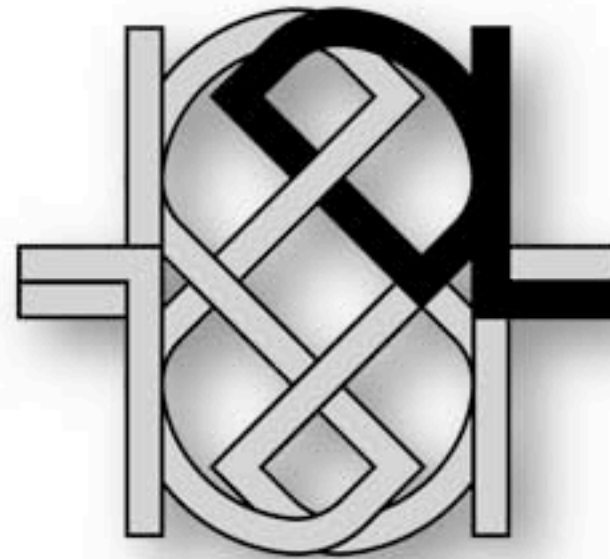
Exercise 11. *Show that the following equivalences are not valid:*

- (a) $\exists r.(C \sqcap D) \equiv \exists r.C \sqcap \exists r.D,$
- (b) $C \sqcap (D \sqcup E) \equiv (C \sqcap D) \sqcup E,$
- (c) $\exists r.\{a\} \sqcap \exists r.\{b\} \equiv \geq 2.\{a, b\},$
- (d) $\exists r.\top \sqcap \exists s.\top \equiv \exists r.\exists r^-. \exists s.\top.$

Exercise 20. Find a way to emulate $C(\mathbf{a}) \vee D(\mathbf{b})$ in \mathcal{SHIQ} .

Exercise 21. Consider whether it is possible to emulate \mathcal{ABox} statements of the shape $\neg r(\mathbf{a}, \mathbf{b})$, $\mathbf{a} \approx \mathbf{b}$, and $\mathbf{a} \not\approx \mathbf{b}$ with an \mathcal{ALCHIQ} knowledge base by using only \mathcal{ABox} statements of the form $C(\mathbf{a})$ and $r(\mathbf{a}, \mathbf{b})$.

Thank You!

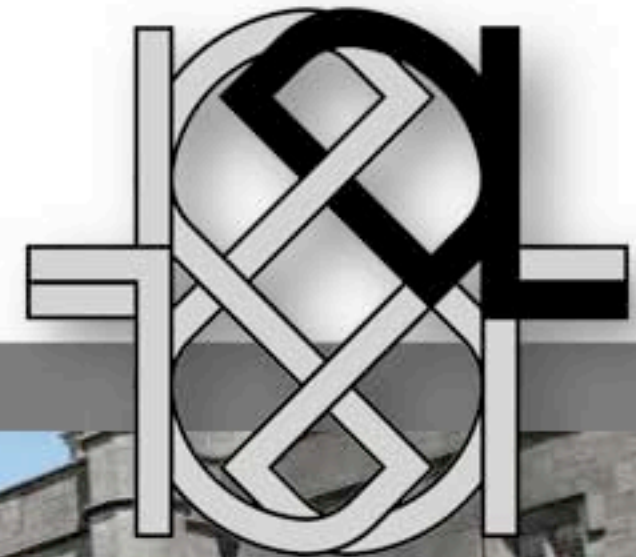


Stay tuned for the second part!

Foundations of Description Logics ...and OWL (ctd.)

Sebastian Rudolph
Karlsruhe Institute of Technology

Institut für Angewandte Informatik und Formale Beschreibungsverfahren



Modeling with DLs

*While frowning on plurality,
 The pope likes cardinality:
 It can enforce infinity,
 And hence endorse divinity.
 But, theologically speaking,
 The papal theory needs tweaking
 For it demands divine assistance
 to prove "the three are one"-consistence.*



Frequent Modeling Features

- domain: $\exists \text{authorOf}. \top \sqsubseteq \text{Person}$
- range: $\top \sqsubseteq \forall \text{authorOf}. \text{Publication}$
- or $\exists \text{authorOf}^{-}. \top \sqsubseteq \text{Publication}$

- concept disjointness: $\text{Male} \sqcap \text{Female} \sqsubseteq \perp$
- or $\text{Male} \sqsubseteq \neg \text{Female}$

- role symmetry: $\text{marriedWith}^{-} \sqsubseteq \text{marriedWith}$

- role transitivity: $\text{partOf} \circ \text{partOf} \sqsubseteq \text{partOf}$

Number Restrictions

- allow for defining that a role is functional:

$$\top \sqsubseteq \leq 1 \text{hasFather} . \top$$

- ...or inverse functional:

$$\top \sqsubseteq \leq 1 \text{hasFather}^{-} . \top$$

- allow for enforcing an infinite domain:

$$(\forall \text{succ}^{-} . \top)(\text{zero}) \quad \top \sqsubseteq \exists \text{succ} . \top \quad \top \sqsubseteq \leq 1 . \text{succ}^{-} . \top$$

- Consequently, DLs with number restrictions and inverses do not have the finite model property.

Nominal Concept and Universal Role

- allow to restrict the size of concepts:

$$\text{AtMostTwo} \sqsubseteq \{\text{one}, \text{two}\} \qquad \text{AtMostTwo} \sqsubseteq \leq 2 u. \top$$

- even allow to restrict the size of the domain:

$$\top \sqsubseteq \{\text{one}, \text{two}\} \qquad \top \sqsubseteq \leq 2 u. \top$$

Self-Restriction

- allows to define a role as reflexive
 $\top \sqsubseteq \exists \text{knows}.\text{Self}$
- allows to define a role as irreflexive
 $\exists \text{betterThan}.\text{Self} \sqsubseteq \perp$

- together with inverses, we can even axiomatize equality:
 $\top \sqsubseteq \exists \text{equals}.\text{Self} \qquad \top \sqsubseteq \leq 1 \text{equals}.\top$

Open vs. Closed World Assumption

- CWA: Closed World Assumption
The knowledge base contains all information, non-derivable axioms are assumed to be false.
- OWA: Open World Assumption
The knowledge base may be incomplete. The truth of non-derivable axioms is simply unknown.
- With DLs, the OWA is applied (as for FOL in general), certain closed-world information can be axiomatized via number restrictions and nominals

Are all children of Bill male?

No idea, since we do not know all children of Bill.

If we assume that we know everything about Bill, then all of his children are male.

		DL answers	Prolog
child(bill,bob)	? $\models \forall \text{child. Man}(\text{Bill})$	don't know	yes
Man(bob)			
$\leq 1 \text{child. } \top(\text{Bill})$? $\models \forall \text{child. Man}(\text{Bill})$	yes	<i>Now we know everything about Bill's children.</i>

Reasoning Tasks and Their Reducibility

A knowledge base with statements in it

Seeks a model sound and nice

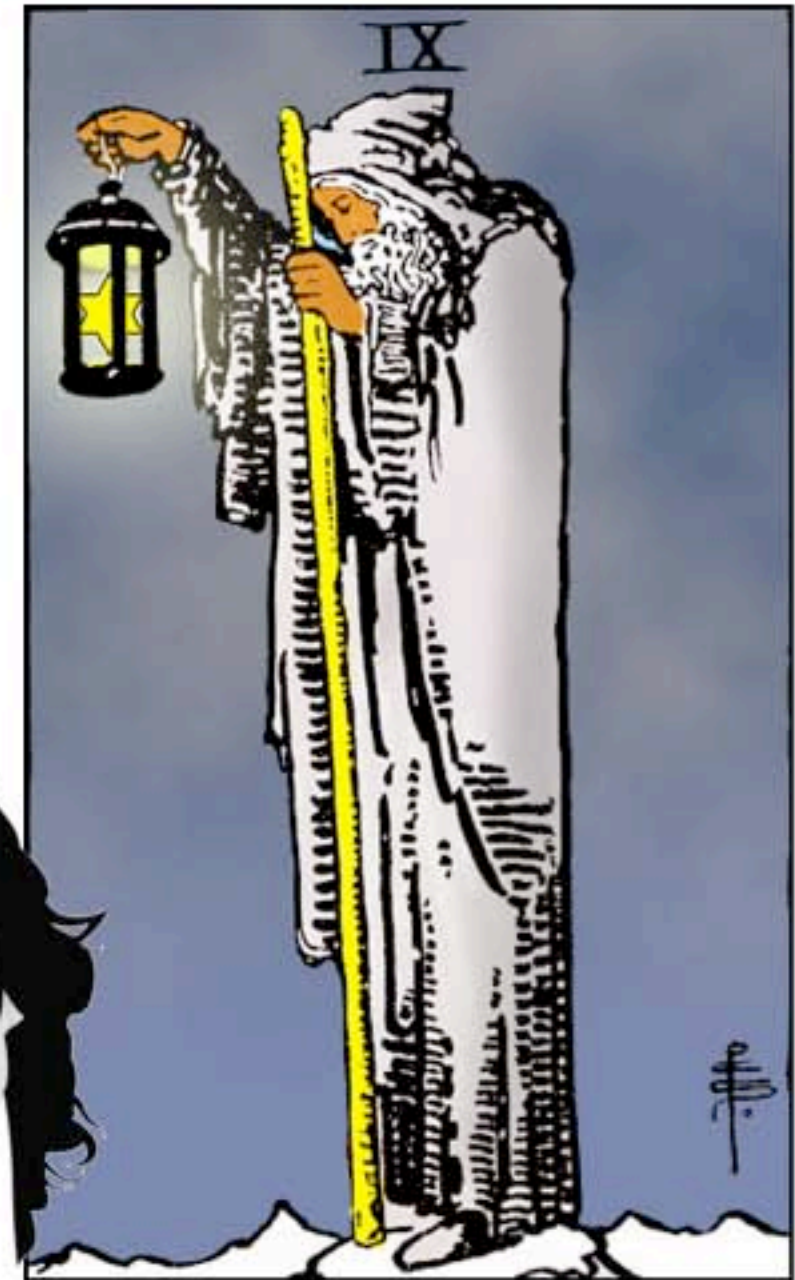
No matter, finite or infinite,

It asks a hermit for advice.

Yet, shattering is the reaction:

“Inconsistency detection,

You can't get no satisfaction.”



Standard DL Inference Problems

Given a knowledge base KB, we might want to know:

- whether the KB is consistent,
- whether the KB entails a certain axiom
(such as **Alive(schrödinger)**),
- whether a given concept is (un)satisfiable
(such as **Dead \sqcap Alive**),
- all the individuals known to be instances a certain concept
- the subsumption hierarchy of all atomic concepts

Knowledge Base Consistency

- basic inferencing task
- directly needed in the process of KB engineering in order to detect severe modelling errors
- other tasks can be reduced to checking KB (in)consistency

Entailment Checking

- used in the KB modelling process to check, whether the specified knowledge has the intended consequences
- used for querying the KB if certain propositions are necessarily true
- can be reduced to checking KB inconsistency (along the idea of indirect proof) by
 - negating the axiom the entailment of which is to be checked
 - adding the negated axiom to the knowledge base
 - checking for inconsistency of the KB
- if axiom cannot be negated directly, its negation can be emulated

Entailment Checking

α	\mathcal{A}_α
$r_1 \circ \dots \circ r_n \sqsubseteq r$	$\{\neg r(c_0, c_n), r_1(c_0, c_1), \dots, r_n(c_{n-1}, c_n)\}$
$\text{Dis}(r, r')$	$\{r(c_1, c_2), r'(c_1, c_2)\}$
$C \sqsubseteq D$	$\{(C \sqcap \neg D)(c)\}$ or: $\{\top \sqsubseteq \exists u(C \sqcap \neg D)\}$
$C(a)$	$\{\neg C(a)\}$
$r(a, b)$	$\{\neg r(a, b)\}$
$\neg r(a, b)$	$\{r(a, b)\}$
$a \approx b$	$\{a \not\approx b\}$
$a \not\approx b$	$\{a \approx b\}$

Table 1. Definition of axiom sets \mathcal{A}_α such that $\mathcal{KB} \models \alpha$ exactly if $\mathcal{KB} \cup \mathcal{A}_\alpha$ is unsatisfiable. Individual names c with possible subscripts are supposed to be fresh. For GCIs (third line), the first variant is normally employed, however, we also give a variant which is equivalent instead of just emulating.

Concept satisfiability

- A concept expression C is called *satisfiable* with respect to a knowledge base, if there is a model of this KB where $C^{\mathcal{I}}$ is not empty.
- Unsatisfiable atomic concepts normally indicate modeling errors in the KB.
- Checking concept satisfiability can be reduced to checking (non-)entailment: C is satisfiable wrt. a KB if the KB does **not** entail the axiom $C \sqsubseteq \perp$.

Instance Retrieval

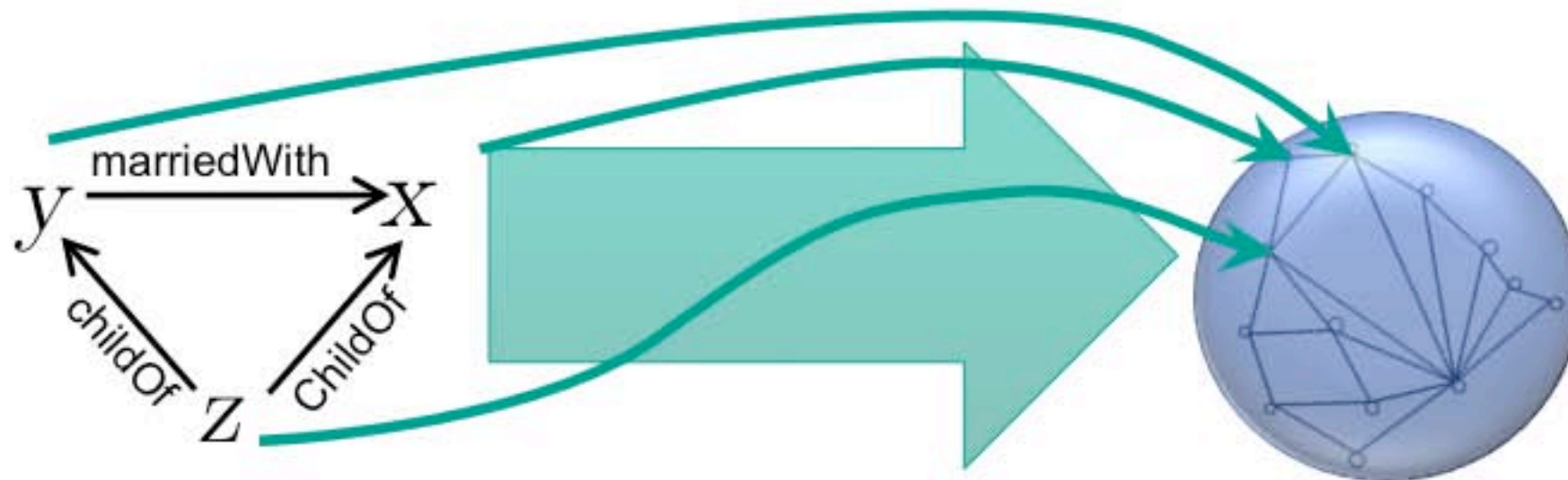
- Asking for all the named individuals known to be in a certain concept (role) is a typical querying or retrieval task.
- It can be reduced to checking entailment of as many individual assertions as there are named individuals in the knowledge base.
- Depending on the used system and inferencing algorithm, this can be done in a much more efficient way (e.g. by translation into a database query).

Classification

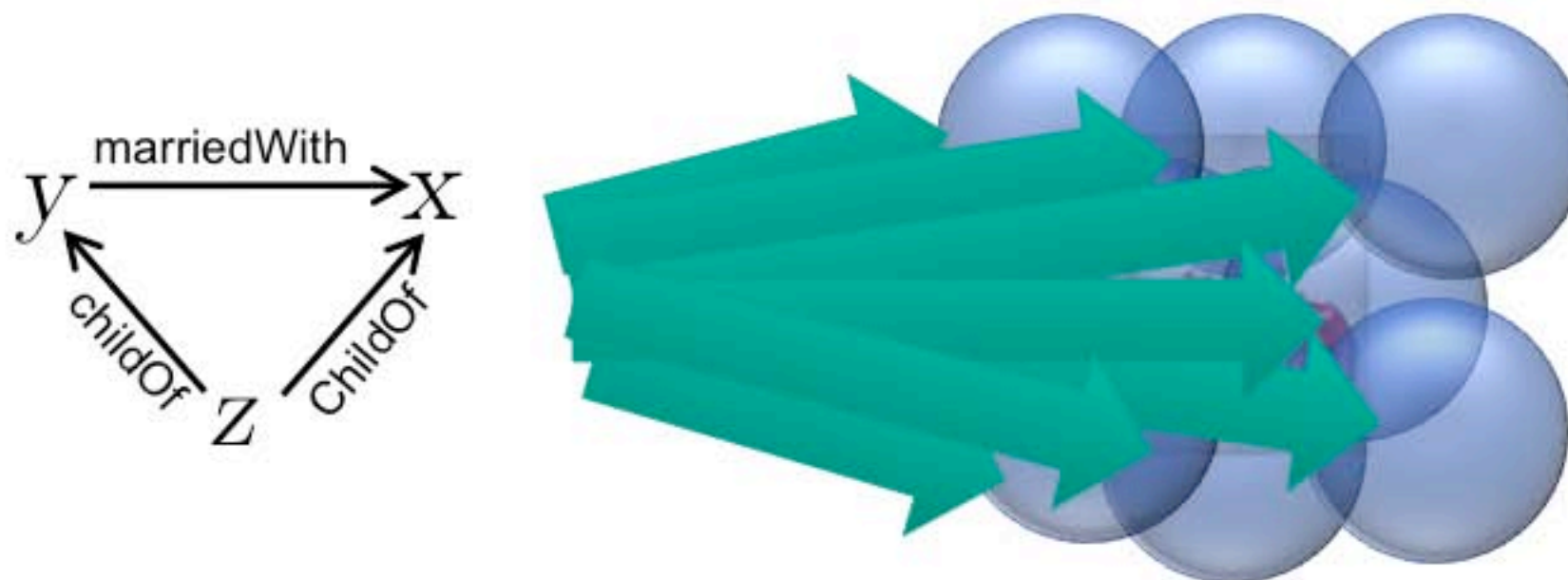
- Classification of a knowledge base aims at determining for any two concept names A , B , whether $A \sqsubseteq B$ is a consequence of the KB.
- This is useful at KB design time for checking the inferred concept hierarchy. Also, computing this hierarchy once and storing it can speed up further queries.
- Classification can be reduced to checking entailment of GCI's.
- While this requires quadratically many checks, one can often do much better in practice by applying optimizations and exploiting that subsumption is a preorder.

Conjunctive Query Answering

- in databases: just one model (the database itself); this is rather easy



- in logics: one knowledge base, many models; not so easy



Conjunctive Query Answering

- cannot be reduced to the other standard reasoning tasks
- but the other reasoning tasks can be reduced to CQ answering
- often much harder than entailment checks in terms of computational complexity
- for *SR₀IQ*, decidability of CQ answering even not yet proven (although conjectured)
- *DL-safe queries* (all variables bound to named individuals) are much easier and often sufficient in practice

Further Reasoning Tasks

- *induction*
given ABox data find GCIs or concept expressions covering it
- *abduction*
given a KB and a wanted but not yet entailed consequence
find (basic) axioms allowing to deduce it
- *explanation*
given an entailment, find a small and intuitive formal
evidence (such as a minimal sub-KB giving rise to this
entailment or a proof)
- *module extraction*
given a KB, divide it into parts with no or only very logical
interdependencies

Is it consequence-driven

Automatically given

What we base our system upon?

*Or do, fueled by Rousseau,
we say “Guerre aux tableaux!*

Et vive la resolution!”?



Types of Reasoning Procedures

- Roughly, DL inferencing algorithms can be separated into two groups:
 - *Model-based algorithms* attempt to show satisfiability by constructing a model (or a representation of it).
Examples: tableaux, automata, type elimination
 - *Proof-based algorithms* apply deduction rules to the KB in order to infer new axioms.
Examples: resolution, consequence-based

NB.: Both strategies are known from FOL theorem proving, but additional care has to be invested to ensure decidability (in particular completeness and termination of the respective algorithm).

Tableaux

- Tableaux methods perform a “bottom-up“ construction of a model:
 - Initialize an interpretation by all explicitly known (i.e. named) individuals and their known properties.
 - Most probably, this “model draft“ will violate some of the axioms.
 - We iteratively “repair“ it by adding new information about concept or role memberships and/or introducing new (i.e. anonymous) individuals; this may require case distinction and backtracking.
 - If we arrive at an interpretation satisfying all axioms, satisfiability has been shown.
 - If every repairing attempt eventually results in an overt inconsistency, unsatisfiability has been shown.

N.B.: Since the finite model property cannot be taken for granted, not the full model is constructed but a representation of it (cf. “blocking“).

Automata

- Automata-based approaches normally rely on some variant of the tree-model property.
- The KB is translated into a tree automaton that recognizes all tree models of the given KB.
- Then, satisfiability of the KB can be checked by determining whether the tree language recognized by the automaton is non-empty.
- Depending on the expressiveness of the underlying KB, elaborate automata have to be applied that are able to recognize infinite trees, can jump back to certain elements and traverse the tree in both ways.

Type Elimination

- Type elimination builds model representations in a “top-down” manner.
- The model representation consists of a set of small pieces (aka *types*, can be single individuals, pairs of individuals,...) and their concept/role membership information.
- Starting from all possible pieces, one successively eliminates those which contradict the KB (and the other still present types)
- If one ends up with a stable nonempty set of types, the KB is satisfiable, if the result is empty, it is unsatisfiable.

Resolution

- Resolution is a proof-based method commonly applied in FOL theory proving, having the resolution rule at its core:

$$\text{Res} \frac{A_1 \vee \dots \vee A_i \vee \dots A_n \quad B_1 \vee \dots \vee B_j \vee \dots B_m}{A_1 \vee \dots \vee A_{i-1} \vee A_{i+1} \vee \dots A_n \vee B_1 \vee \dots \vee B_{j-1} \vee B_{j+1} \vee \dots B_m}$$

(with A_i and B_j being negated versions of each other)

- KB is translated into FOL and resolution is applied
- resolution per se is not a decision procedure, special care has to be taken to guarantee termination (yet, completeness)
- this is achieved by specifying when which literals are eligible for being resolved
- if the empty disjunction can be derived, the KB is unsatisfiable

Consequence-Based Reasoning

- Consequence-based approaches are proof-based methods applying DL deduction rules exhaustively to the given KB (no prior translation to FOL).
- Normalization of the KB and careful design of the deduction calculus ensure that only finitely many new axioms can be derived, yet the procedure is still complete in a certain sense.
- Approach especially useful for Horn-DLs and for reasoning tasks where many mutually dependent consequences have to be checked (such as classification).

Description Logics and OWL

*In fact, in terms of syntax, OWL
Just tends to be a bulky fowl,
However, if it mates with Turtle
This union turns out rather fertile;
I deem the offspring of this love
As graceful as a turtledove.*



How Do DLs and OWL Relate

- OWL is essentially
 - *SROIQ* in disguise
 - plus extended datatype support
 - plus extralogical features such as annotations, versioning etc.

- OWL speak is different from DL terminology:

OWL	DL	FOL
class name	concept name	unary predicate
class	concept	formula with one free variable
object property name	role name	binary predicate
object property	role	formula with two free variables
ontology	knowledge base	theory
axiom	axiom	sentence
vocabulary	vocabulary / signature	signature

Translating DL into OWL

- Next to the logic part, an OWL ontology features a preamble and a declaration part:

$$[[\mathcal{KB}]] = \text{Pre} + \text{Dec}(\mathcal{KB}) + \sum_{\alpha \in \mathcal{KB}} [[\alpha]]$$

$$\text{Pre} = \begin{cases} \text{@prefix owl:} <\text{http://www.w3.org/2002/07/owl\#}> . \\ \text{@prefix rdfs:} <\text{http://www.w3.org/2000/01/rdf-schema\#}> . \\ \text{@prefix rdf:} <\text{http://www.w3.org/1999/02/22-rdf-syntax-ns\#}> . \\ \text{@prefix xsd:} <\text{http://www.w3.org/2001/XMLSchema\#}> . \end{cases}$$

$$\begin{aligned} \text{Dec}(\mathcal{KB}) = & \sum_{A \in N_C(\mathcal{KB})} A \text{ rdf:type owl:Class} . \\ & + \sum_{r \in N_R(\mathcal{KB})} r \text{ rdf:type owl:ObjectProperty} . \end{aligned}$$

Translating DL axioms into OWL

- Following the Semantic Web rationale, OWL axioms are expressed in terms of RDF, i.e. as triples. As far as possible, RDFS vocabulary is reused.

$$\llbracket r_1 \circ \dots \circ r_n \sqsubseteq r \rrbracket = \llbracket r \rrbracket_{\mathbf{R}} \text{ owl:propertyChainAxiom } (\llbracket r_1 \rrbracket_{\mathbf{R}} \cdots \llbracket r_n \rrbracket_{\mathbf{R}}) .$$

$$\llbracket \text{Dis}(r, r') \rrbracket = \llbracket r \rrbracket_{\mathbf{R}} \text{ owl:propertyDisjointWith } \llbracket r' \rrbracket_{\mathbf{R}} .$$

$$\llbracket C \sqsubseteq D \rrbracket = \llbracket C \rrbracket_{\mathbf{C}} \text{ rdfs:subClassOf } \llbracket D \rrbracket_{\mathbf{C}} .$$

$$\llbracket C(a) \rrbracket = a \text{ rdf:type } \llbracket C \rrbracket_{\mathbf{C}} .$$

$$\llbracket r(a, b) \rrbracket = a \text{ } r \text{ } b .$$

$$\llbracket r^-(a, b) \rrbracket = b \text{ } r \text{ } a .$$

$$\begin{aligned} \llbracket \neg r(a, b) \rrbracket = & [] \text{ rdf:type owl:NegativePropertyAssertion ;} \\ & \text{ owl:assertionProperty } \llbracket r \rrbracket_{\mathbf{R}} ; \\ & \text{ owl:sourceIndividual } a \text{ ; owl:targetValue } b . \end{aligned}$$

$$\llbracket a \approx b \rrbracket = a \text{ owl:sameAs } b .$$

$$\llbracket a \not\approx b \rrbracket = a \text{ owl:differentFrom } b .$$

$$[u]_R = \text{owl:topObjectProperty}$$

$$[r]_R = r$$

$$[r -]_R = [\text{owl:inverseOf } :r]$$

$$[A]_C = A$$

$$[\top]_C = \text{owl:Thing}$$

$$[\perp]_C = \text{owl:Nothing}$$

$$[\{a_1, \dots, a_n\}]_C = [\text{rdf:type owl:Class ; owl:oneOf (:a}_1 \dots :a_n)]$$

$$[\neg C]_C = [\text{rdf:type owl:Class ; owl:complementOf } [C]_C]$$

$$[C_1 \sqcap \dots \sqcap C_n]_C = [\text{rdf:type owl:Class ; owl:intersectionOf } ([C_1]_C \dots [C_n]_C)]$$

$$[C_1 \sqcup \dots \sqcup C_n]_C = [\text{rdf:type owl:Class ; owl:unionOf } ([C_1]_C \dots [C_n]_C)]$$

$$[\exists r.C]_C = [\text{rdf:type owl:Restriction ; } \\ \text{owl:onProperty } [r]_R ; \text{owl:someValuesFrom } [C]_C]$$

$$[\forall r.C]_C = [\text{rdf:type owl:Restriction ; } \\ \text{owl:onProperty } [r]_R ; \text{owl:allValuesFrom } [C]_C]$$

$$[\exists r.\text{Self}]_C = [\text{rdf:type owl:Restriction ; } \\ \text{owl:onProperty } [r]_R ; \text{owl:hasSelf "true"^^xsd:boolean}]$$

$$[\geq n r.C]_C = [\text{rdf:type owl:Restriction ; } \\ \text{owl:minQualifiedCardinality } n^{\text{^^xsd:nonNegativeInteger}} ; \\ \text{owl:onProperty } [r]_R ; \text{owl:onClass } [C]_C]$$

$$[\leq n r.C]_C = [\text{rdf:type owl:Restriction ; } \\ \text{owl:maxQualifiedCardinality } n^{\text{^^xsd:nonNegativeInteger}} ; \\ \text{owl:onProperty } [r]_R ; \text{owl:onClass } [C]_C]$$

An OWL Ontology in RDF/Turtle serialisation

<p>RBox \mathcal{R}</p> <p>$\text{owns} \sqsubseteq \text{caresFor}$ "If somebody owns something, they care for it."</p>
<p>TBox \mathcal{T}</p> <p>$\text{Healthy} \sqsubseteq \neg \text{Dead}$ "Healthy beings are not dead."</p> <p>$\text{Cat} \sqsubseteq \text{Dead} \sqcup \text{Alive}$ "Every cat is dead or alive."</p> <p>$\text{HappyCatOwner} \sqsubseteq \exists \text{owns.Cat} \sqcap \forall \text{caresFor.Healthy}$ "A happy cat owner owns a cat and all beings he cares for are healthy."</p>
<p>ABox \mathcal{A}</p> <p>$\text{HappyCatOwner}(\text{schrödinger})$ "Schrödinger is a happy cat owner."</p>

```
@prefix : <http://www.example.org/#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

:owns rdf:type owl:ObjectProperty .
:caresFor rdf:type owl:ObjectProperty .
:Cat rdf:type owl:Class .
:Dead rdf:type owl:Class .
:Alive rdf:type owl:Class .
:Healthy rdf:type owl:Class .
:HappyCatOwner rdf:type owl:Class .

:owns rdfs:subPropertyOf :caresFor .

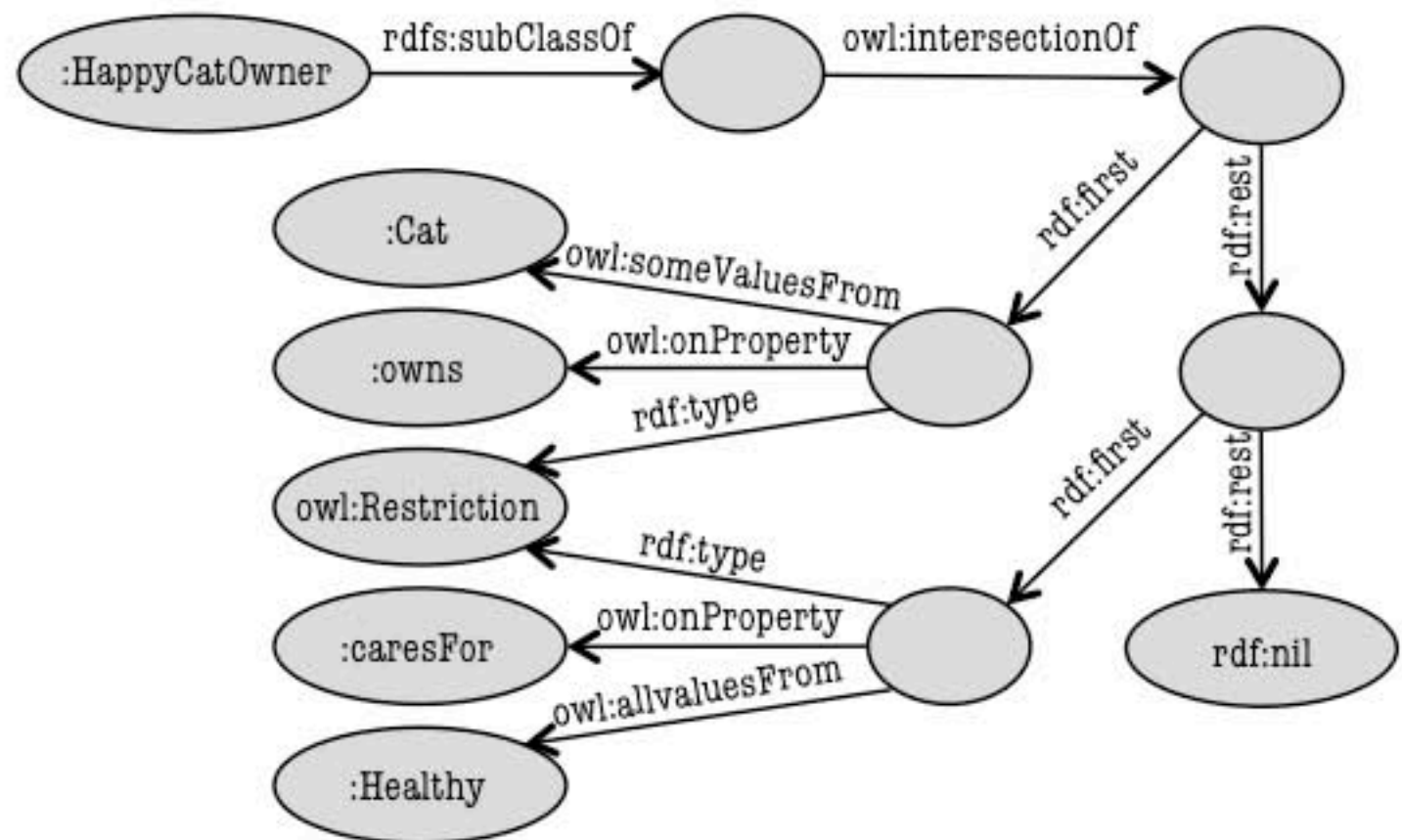
:Healthy rdfs:subClassOf [ owl:complementOf :Dead ] .
:Cat rdfs:subClassOf [ owl:unionOf (:Dead :Alive) ] .
:HappyCatOwner rdfs:subClassOf
  [ owl:intersectionOf
    ( [ rdf:type owl:Restriction ;
        owl:onProperty :owns ; owl:someValuesFrom :Cat ]
      [ rdf:type owl:Restriction ;
        owl:onProperty :caresFor ; owl:allValuesFrom :Healthy ] )
  ] .

:schrödinger rdf:type :HappyCatOwner .
```

Behind the Scenes...

```

:HappyCatOwner rdfs:subClassOf
  [ owl:intersectionOf
    ( [ rdf:type owl:Restriction ;
        owl:onProperty :owns ; owl:someValuesFrom :Cat ]
      [ rdf:type owl:Restriction ;
        owl:onProperty :caresFor ; owl:allValuesFrom :Healthy] )
  ] .
  
```



Paraphrasing other OWL Axioms in DL

Axiom type	Turtle notation	DL paraphrase
Class Equivalence	<code>[C]C owl:equivalentClass [D]C .</code>	$C \sqsubseteq D, D \sqsubseteq C$
Class Disjointness	<code>[C]C owl:disjointWith [D]C .</code>	$C \cap D \sqsubseteq \perp$
Disjoint Classes	<code>[] rdf:type owl:AllDisjointClasses ; owl:members ([C₁]C ... [C_n]C) .</code>	$C_i \cap C_j \sqsubseteq \perp$ for all $1 \leq i < j \leq n$
Disjoint Union	<code>[C]C owl:disjointUnionOf ([C₁]C ... [C_n]C) .</code>	$\bigsqcup_{i < j} C_i \sqsubseteq C$ $C_i \cap C_j \sqsubseteq \perp$ for all $1 \leq i < j \leq n$
Property Equivalence	<code>[r]R owl:equivalentProperty [s]R .</code>	$r \sqsubseteq s, s \sqsubseteq r$
Disjoint Properties	<code>[] rdf:type owl:AllDisjointProperties ; owl:members ([r₁]R ... [r_n]R) .</code>	$\text{Dis}(r_i, r_j)$ for all $1 \leq i < j \leq n$
Inverse Properties	<code>[r]R owl:inverseOf [s]R .</code>	$\text{Inv}(r) \sqsubseteq s$
Property Domain	<code>[r]R rdfs:domain [C]C .</code>	$\exists r. \top \sqsubseteq C$
Property Range	<code>[r]R rdfs:range [C]C .</code>	$\top \sqsubseteq \forall r. C$
Functional Property	<code>[r]R rdf:type owl:FunctionalProperty .</code>	$\top \sqsubseteq \leq 1r. \top$
Inverse Functional Property	<code>[r]R rdf:type owl:InverseFunctionalProperty .</code>	$\top \sqsubseteq \leq 1\text{Inv}(r). \top$
Reflexive Property	<code>[r]R rdf:type owl:ReflexiveProperty .</code>	$\top \sqsubseteq \exists r. \text{Self}$
Irreflexive Property	<code>[r]R rdf:type owl:IrreflexiveProperty .</code>	$\exists r. \text{Self} \sqsubseteq \perp$
Symmetric Property	<code>[r]R rdf:type owl:SymmetricProperty .</code>	$\text{Inv}(r) \sqsubseteq r$
Asymmetric Property	<code>[r]R rdf:type owl:AsymmetricProperty .</code>	$\text{Dis}(\text{Inv}(r), r)$
Transitive Property	<code>[r]R rdf:type owl:TransitiveProperty .</code>	$r \circ r \sqsubseteq r$
Different Individuals	<code>[] rdf:type owl:AllDifferent ; owl:members (a₁ ... a_n) .</code>	$a_i \not\approx a_j$ for all $1 \leq i < j \leq n$

OWL Profiles



- **Design principle for profiles:**
Identify maximal OWL sublanguages that are still implementable in PTime.
- Main source of intractability: **non-determinism** (requires guessing/backtracking)
 - `owl:unionOf`, or `owl:complementOf` + `owl:intersectionOf`
 - Max. cardinality restrictions
 - Combining existentials (`owl:someValuesFrom`) and universals (`owl:allValuesFrom`) in superclasses
 - Non-unary finite class expressions (`owl:oneOf`) or datatype expressions

→ features that are not allowed in any OWL profile

Many further features can lead to non-determinism – care needed!

- **OWL profile based on description logic $\mathcal{EL}++$**
 - Intuition: focus on terminological expressivity used for light-weight ontologies
 - Allow `owl:someValuesFrom` (existential) but not `owl:allValuesFrom` (universal)
 - Property domains, class/property hierarchies, class intersections, disjoint classes/properties, property chains, `owl:hasSelf`, `owl:hasValue`, and keys fully supported
 - No inverse or symmetric properties
 - `rdfs:range` allowed but with some restrictions
 - No `owl:unionOf` or `owl:complementOf`
 - Various restrictions on available datatypes

- **OWL profile that can be used to query data-rich applications:**
 - Intuition: use OWL concepts as light-weight queries, allow query answering using rewriting in SQL on top of relational DBs
 - Different restrictions on subclasses and superclasses of `rdfs:SubclassOf`:
 - subclasses can only be class names or `owl:someValuesFrom` (existential) with unrestricted (`owl:Thing`) filler
 - superclasses can be class names, `owl:someValuesFrom` or `owl:intersectionOf` with superclass filler (recursive), or `owl:complementOf` with subclass filler
 - Property hierarchies, disjointness, inverses, (a)symmetry supported, restrictions on range and domain
 - Disjoint or equivalence of classes only for subclass-type expressions
 - No `owl:unionOf`, `owl:allValuesFrom`, `owl:hasSelf`, `owl:hasKey`, `owl:hasValue`, `owl:oneOf`, `owl:sameAs`, `owl:propertyChainAxiom`, `owl:TransitiveProperty`, cardinalities, functional properties
 - Some restrictions on available datatypes

- **OWL profile that resembles an OWL-based rule language:**
 - Intuition: subclass axioms in OWL RL can be understood as rule-like implications with head (superclass) and body (subclass)
 - Different restrictions on subclasses and superclasses of `rdfs:SubclassOf`:
 - subclasses can only be class names, `owl:oneOf`, `owl:hasValue`, `owl:intersectionOf`, `owl:unionOf`, `owl:someValuesFrom` if applied only to subclass-type expressions
 - superclasses can be class names, `owl:allValuesFrom` or `owl:hasValue`; also max. cardinalities of 0 or 1 are allowed, all with superclass-type filler expressions only
 - Property domains and ranges only for subclass-type expressions; property hierarchies, disjointness, inverses, (a)symmetry, transitivity, chains, (inverse) functionality, irreflexivity fully supported
 - Disjoint classes and classes in keys need subclass-type expressions, equivalence only for expressions that are sub- and superclass-type, no restrictions on `owl:sameAs`
 - Some restrictions on available datatypes

Do We Really Need So Many OWLs?

- Three new OWL profiles with somewhat complex descriptions ... why not just one?
 - The union of any two of the profiles is no longer light-weight!
QL+RL, QL+EL, RL+EL all ExpTime-hard
 - Restricting to fewer profiles = giving up potentially useful feature combinations
 - Rationale: profiles are “maximal” (well, not quite) well-behaved fragments of OWL 2
→ Pick suitable feature set for applications
 - In particular, nobody is forced to implement *all* of a profile



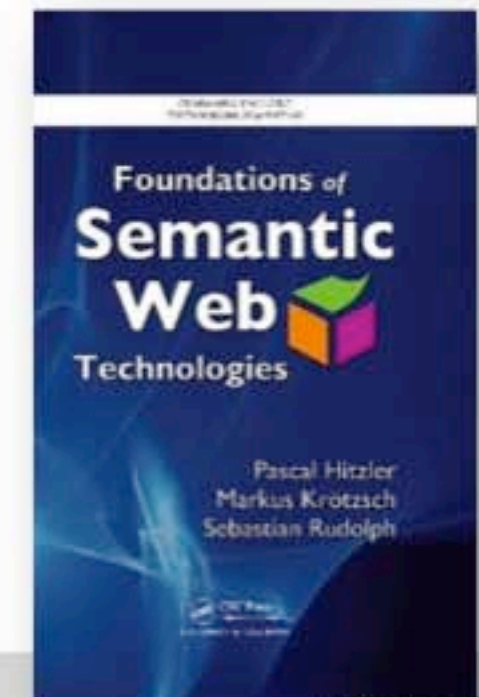
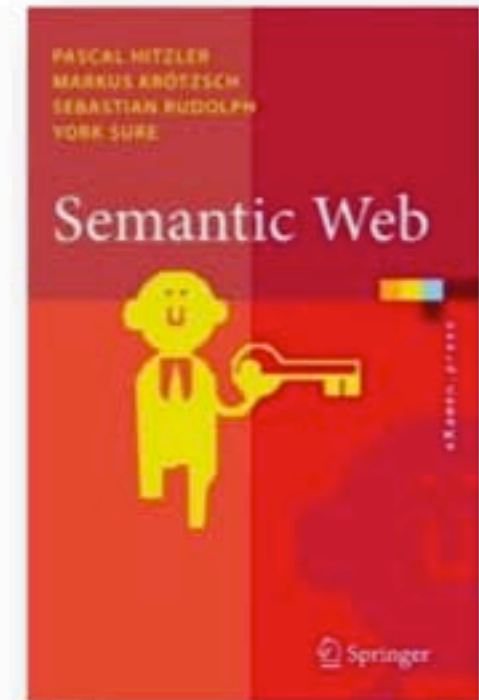
OWL in Practice: Tools



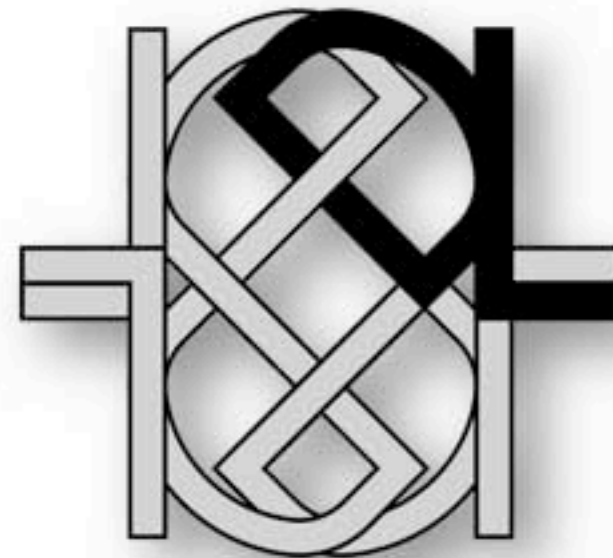
- Editors (<http://semanticweb.org/wiki/Editors>)
 - Most common editor: Protégé 4
 - Other tools: TopBraid Composer (\$), NeOn toolkit
 - Special purpose apps, esp. for light-weight ontologies (e.g. FOAF editors)
- Reasoners (<http://semanticweb.org/wiki/Reasoners>)
 - OWL DL: Pellet, HermiT, FaCT++, RacerPro (\$)
 - OWL EL: CEL, SHER, snorocket (\$)
 - OWL RL: OWLIM, Jena, Oracle Prime (part of O 11g) (\$),
 - OWL QL: Owlgres, QuOnto, Quill
- Many tools use the OWL API library (Java)
- Note: many other Semantic Web tools are found online

References – Textbooks

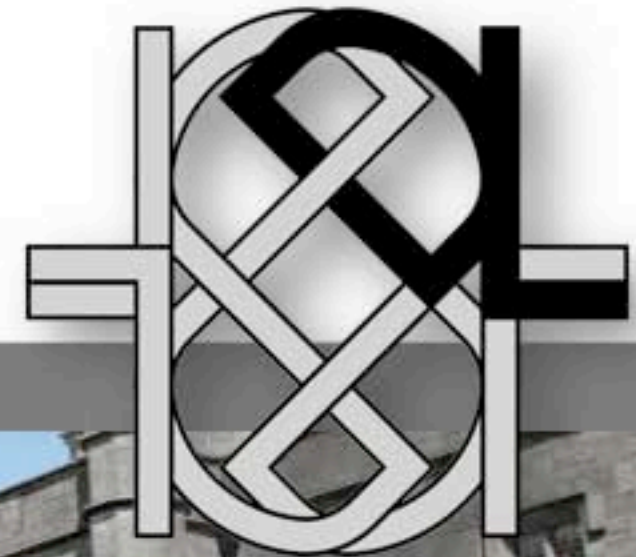
- Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, York Sure, Semantic Web – Grundlagen. Springer, 2008.
<http://www.semantic-web-grundlagen.de/>
(In German.)
- Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, Foundations of Semantic Web Technologies. Chapman & Hall/CRC, 2009.
<http://www.semantic-web-book.org/>
(Ask for a flyer from us.)



Thank You!



Bonus Track 1: Tableaux in Action



Institut für Angewandte Informatik und Formale Beschreibungsverfahren



RW2011
Galway City, Ireland
August 23 - August 27



OWL Reasoning with Tableaux

```
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (
  [ rdf:type owl:Restriction ; owl:onProperty ex:owns ; owl:someValuesFrom ex:Cat ],
  [ rdf:type owl:Restriction ; owl:onProperty ex:caresFor ; owl:allValuesFrom ex:Healthy ] ) ] .
ex:schrödinger rdf:type ex:HappyCatOwner .
```

Knowledge Base

Tableau



OWL Reasoning with Tableaux

Knowledge Base

```
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .  
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .  
ex:owns rdfs:subPropertyOf ex:caresFor .  
ex:HappyCatOwner rdfs:subClassOf ([owl:intersectionOf (  
  [rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],  
  [rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:allValuesFrom ex:Healthy])]) .  
ex:schrödinger rdf:type ex:HappyCatOwner .
```

Tableau


ex:HappyCatOwner



OWL Reasoning with Tableaux

Knowledge Base

```
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (
  [rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],
  [rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:allValuesFrom ex:Healthy]) ] .
ex:schrödinger rdf:type ex:HappyCatOwner .
```



Tableau



ex:HappyCatOwner
[owl:intersectionOf (■, ■)]

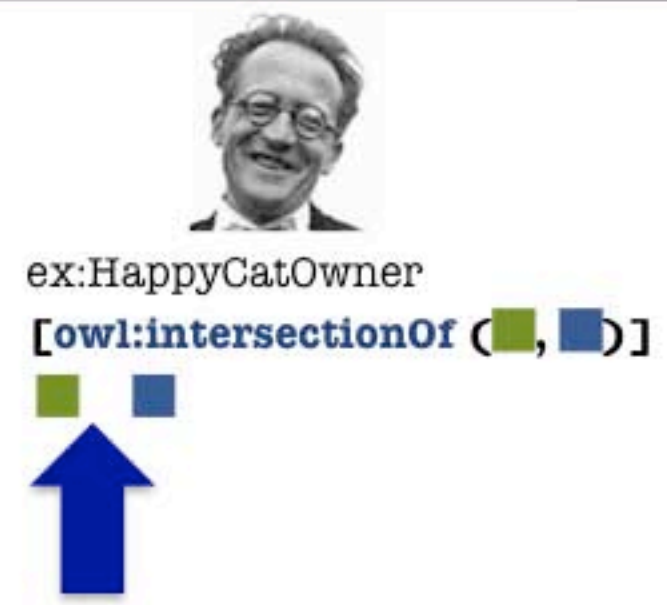




OWL Reasoning with Tableaux

Knowledge Base

```
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (
  [rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],
  [rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:allValuesFrom ex:Healthy])] .
ex:schrödinger rdf:type ex:HappyCatOwner .
```



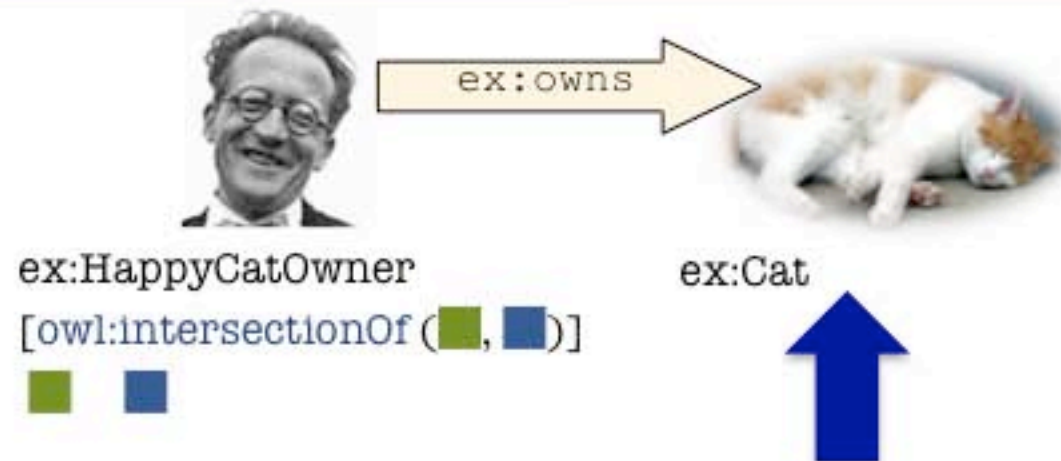
Tableau

OWL Reasoning with Tableaux

Knowledge Base

```
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf ([rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],
[rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:allValuesFrom ex:Healthy]) .
ex:schrödinger rdf:type ex:HappyCatOwner .
```

Tableau

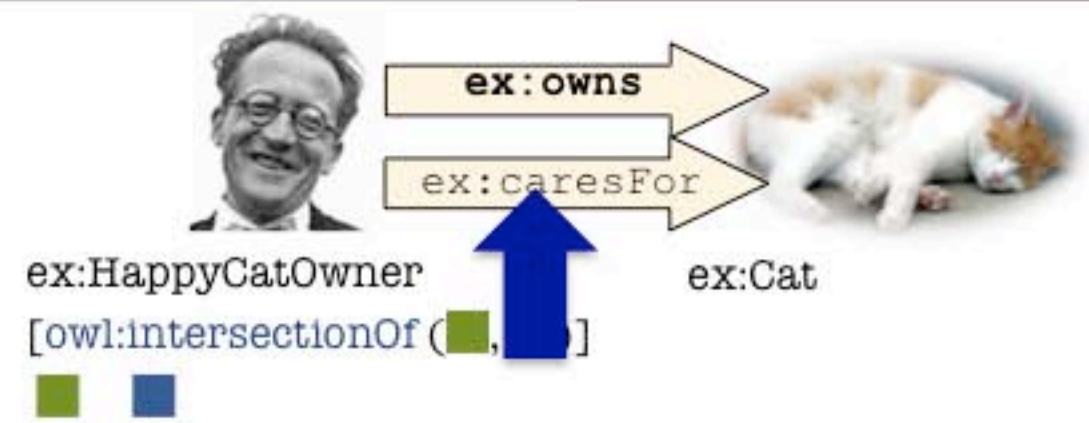


OWL Reasoning with Tableaux

Knowledge Base

```
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (
  [rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],
  [rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:allValuesFrom ex:Healthy])] .
ex:schrödinger rdf:type ex:HappyCatOwner .
```

Tableau



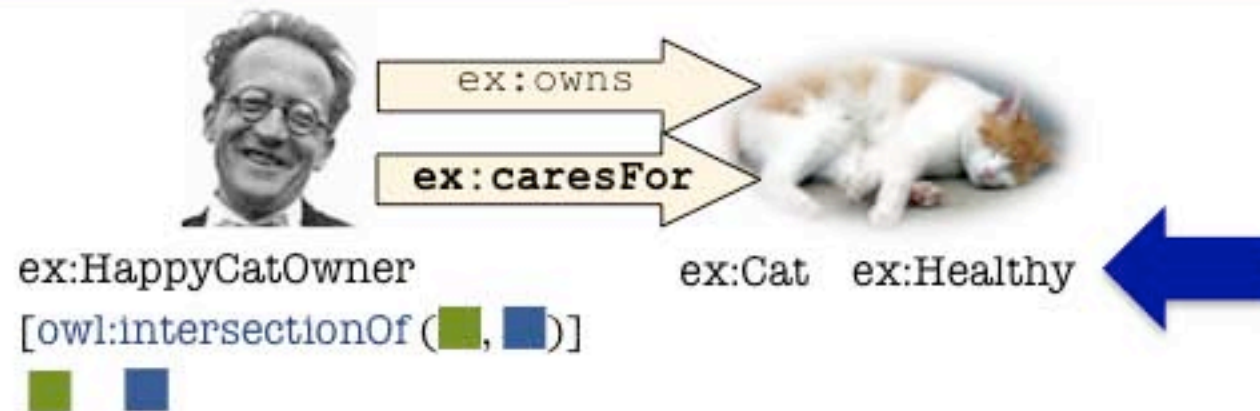
OWL Reasoning with Tableaux

Knowledge Base

```
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (
  [rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],
  [rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:allValuesFrom ex:Healthy]) ] .
ex:schrödinger rdf:type ex:HappyCatOwner .
```



Tableau





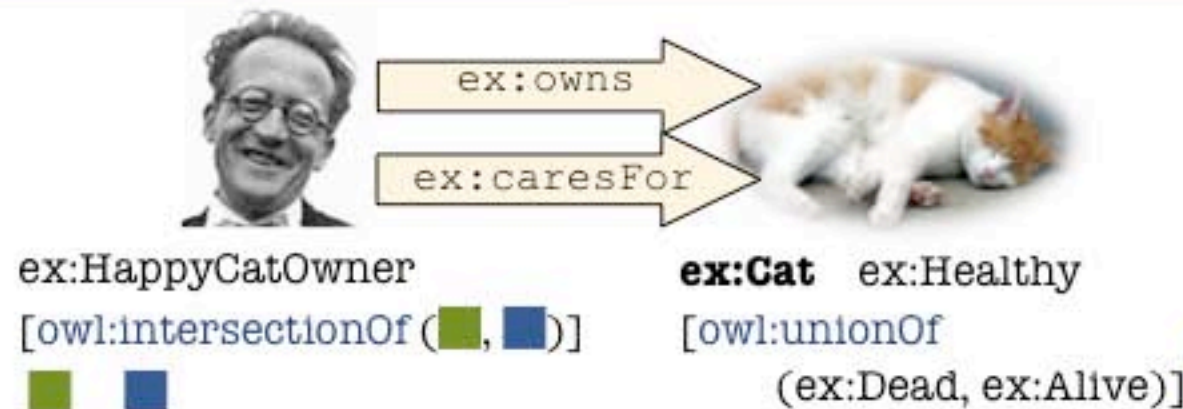
OWL Reasoning with Tableaux

Knowledge Base

```
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (
  [rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],
  [rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:allValuesFrom ex:Healthy])] .
ex:schrödinger rdf:type ex:HappyCatOwner .
```



Tableau

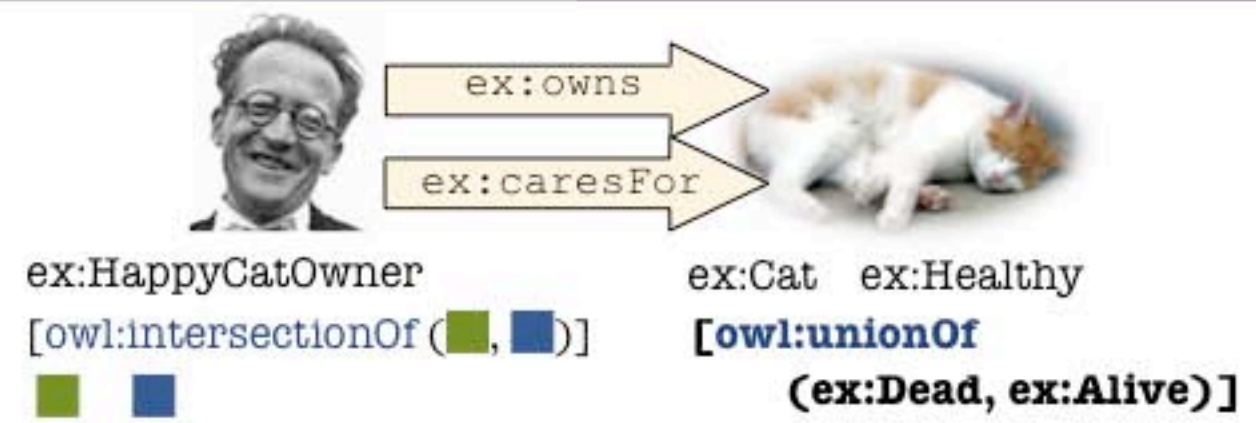


OWL Reasoning with Tableaux

Knowledge Base

```
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (
  [rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],
  [rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:allValuesFrom ex:Healthy])] .
ex:schrödinger rdf:type ex:HappyCatOwner .
```

Tableau





OWL Reasoning with Tableaux

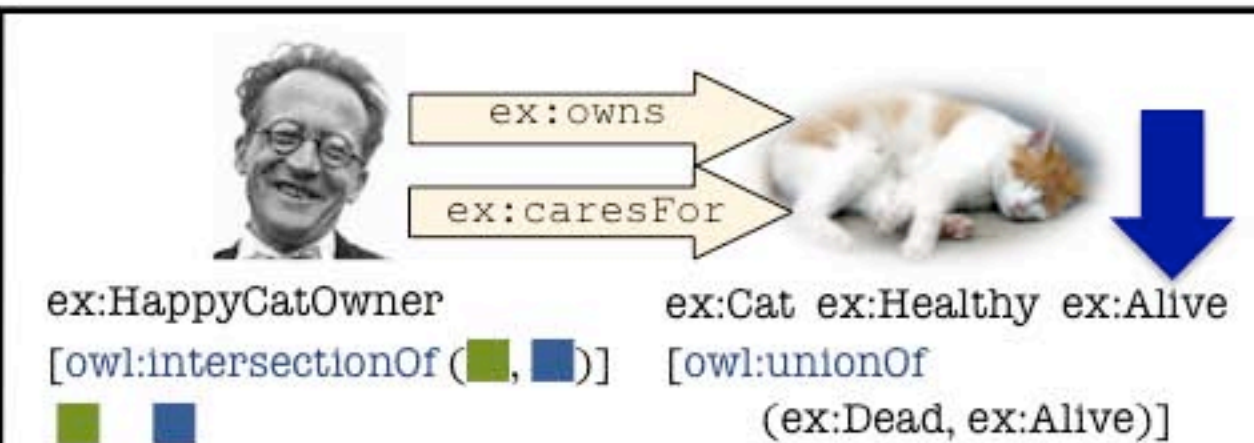
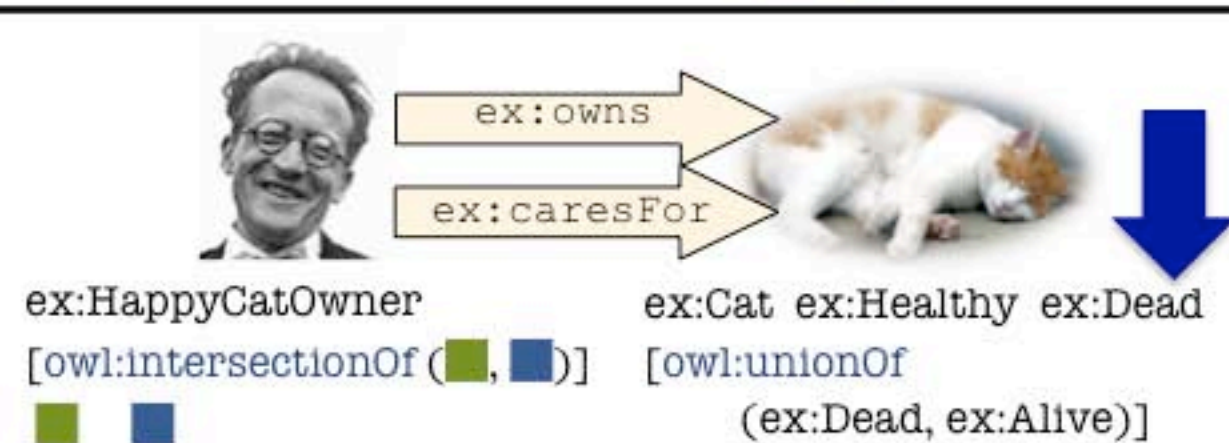
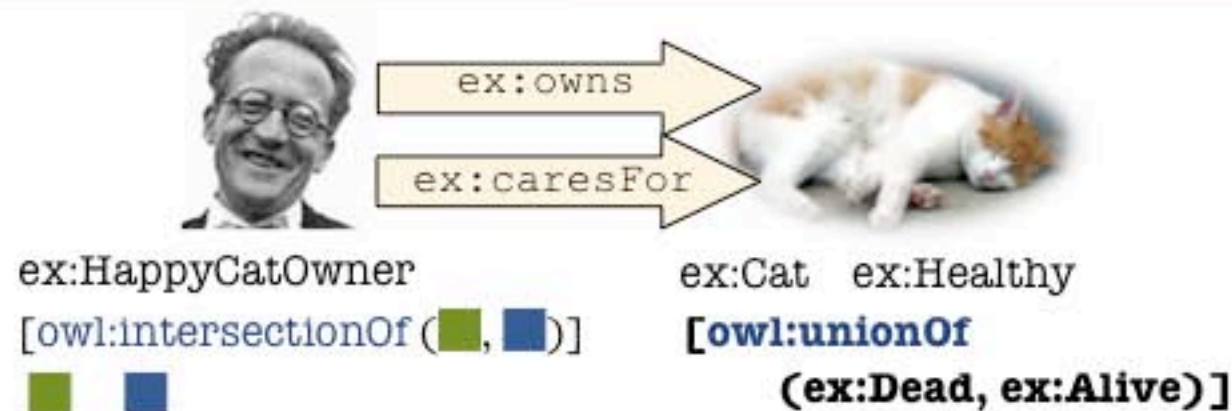
Knowledge Base

```

ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (
  [rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],
  [rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:allValuesFrom ex:Healthy])] .
ex:schrödinger rdf:type ex:HappyCatOwner .

```

Tableau





OWL Reasoning with Tableaux

Knowledge Base

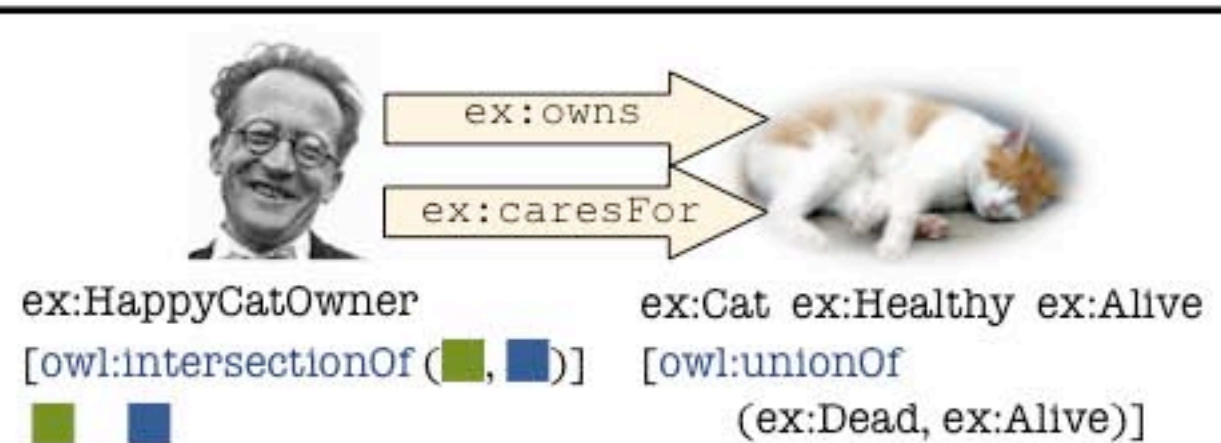
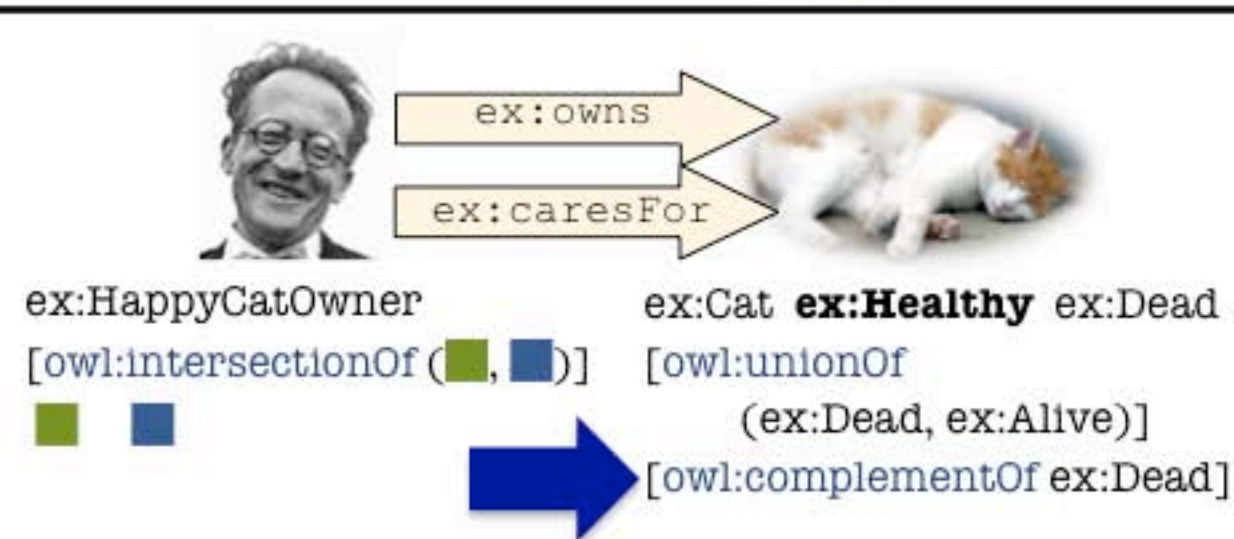
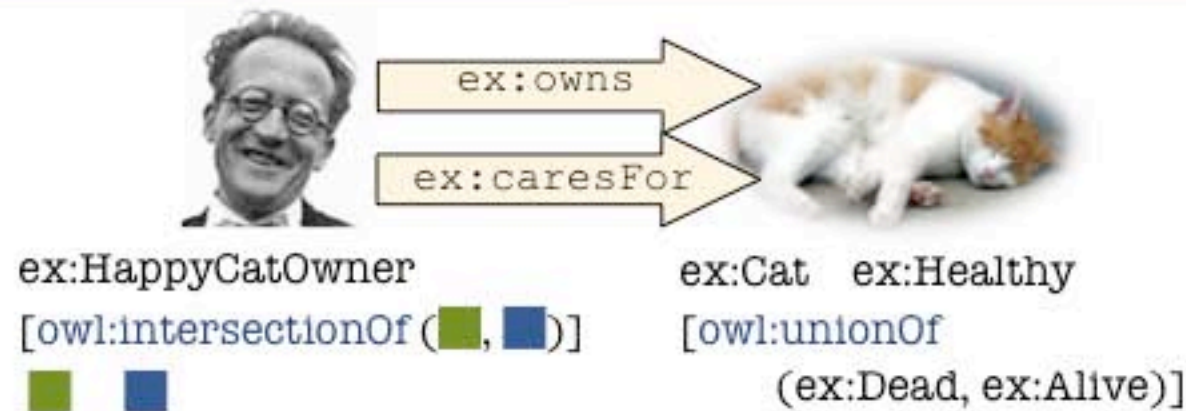


```

ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (
  [rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],
  [rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:allValuesFrom ex:Healthy])] .
ex:schrödinger rdf:type ex:HappyCatOwner .

```

Tableau





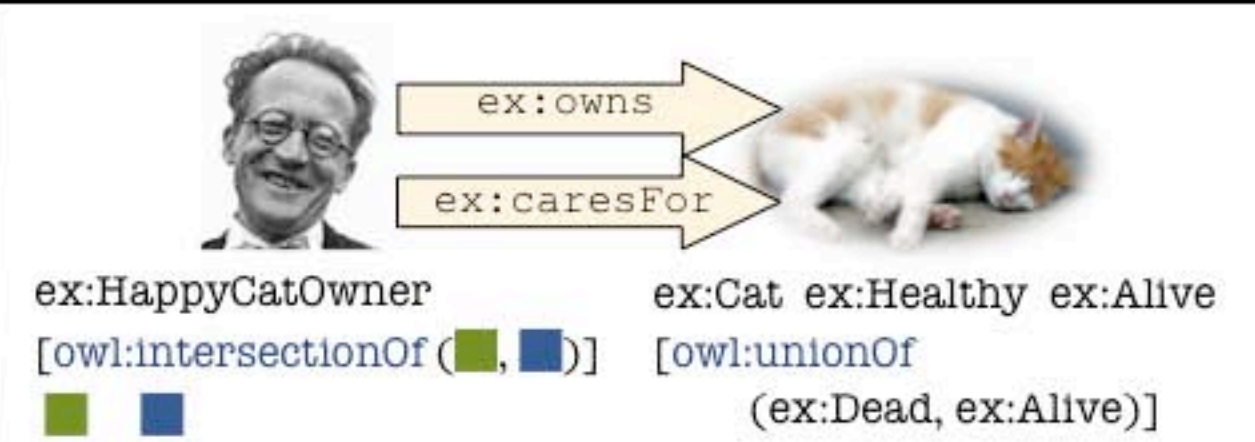
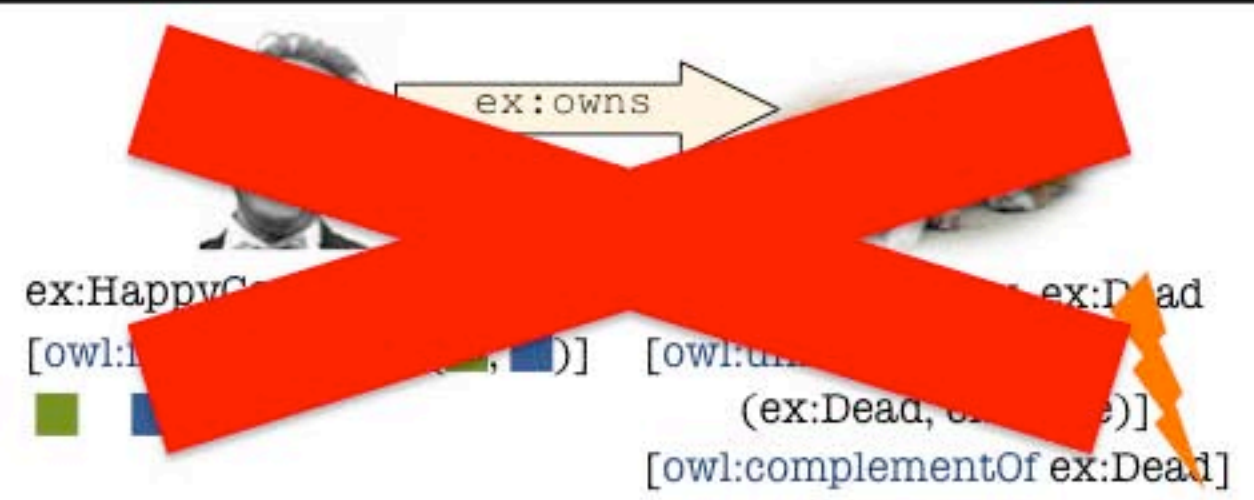
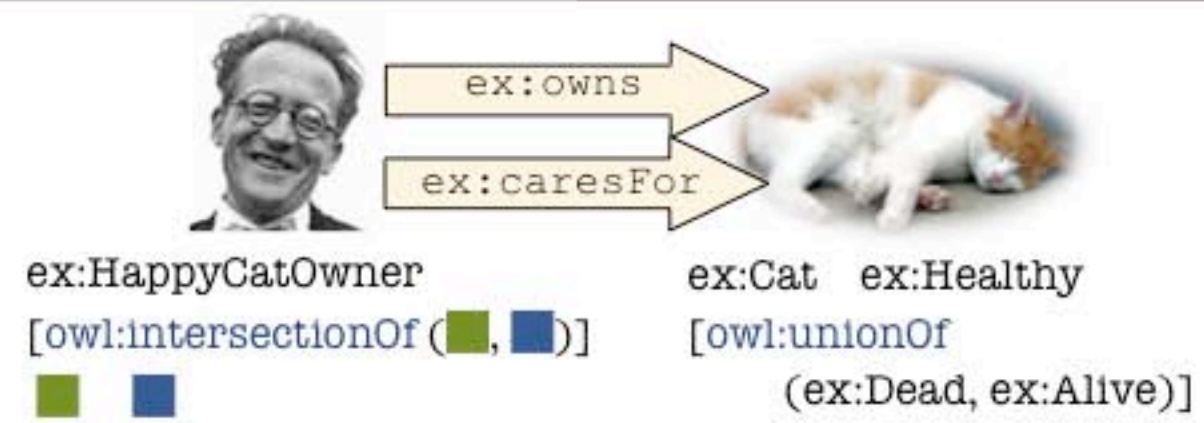
OWL Reasoning with Tableaux

Knowledge Base

```

ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (
  [rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],
  [rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:allValuesFrom ex:Healthy])] .
ex:schrödinger rdf:type ex:HappyCatOwner .
  
```

Tableau





OWL Reasoning with Tableaux

Knowledge Base

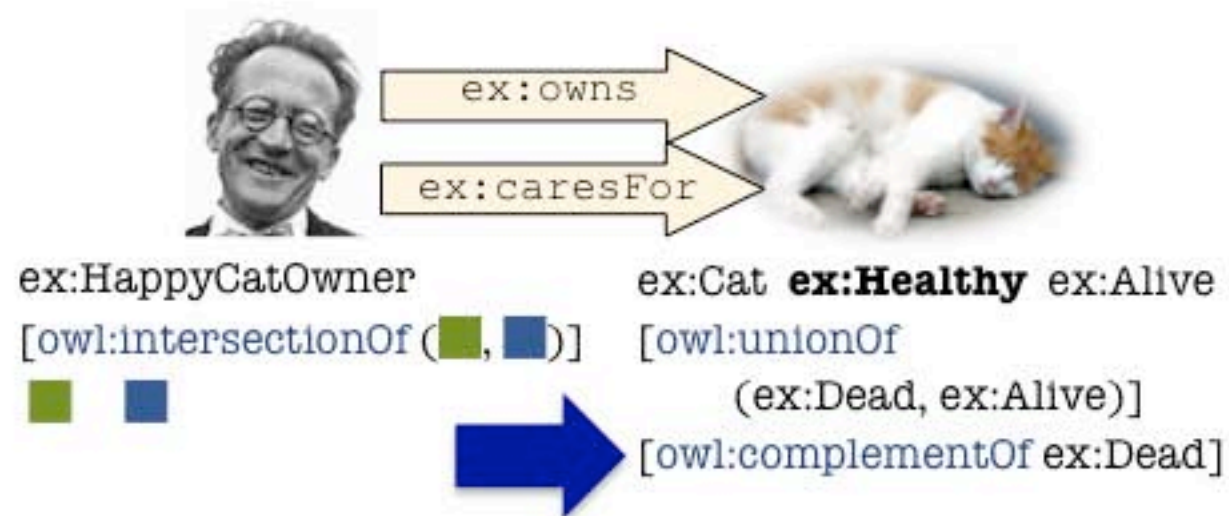
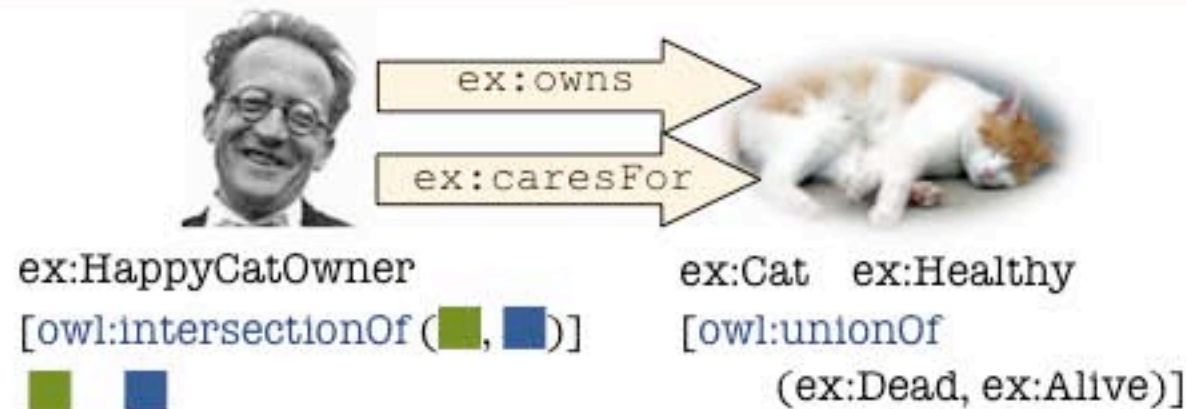


```

ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (
  [rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],
  [rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:allValuesFrom ex:Healthy])] .
ex:schrödinger rdf:type ex:HappyCatOwner .

```

Tableau





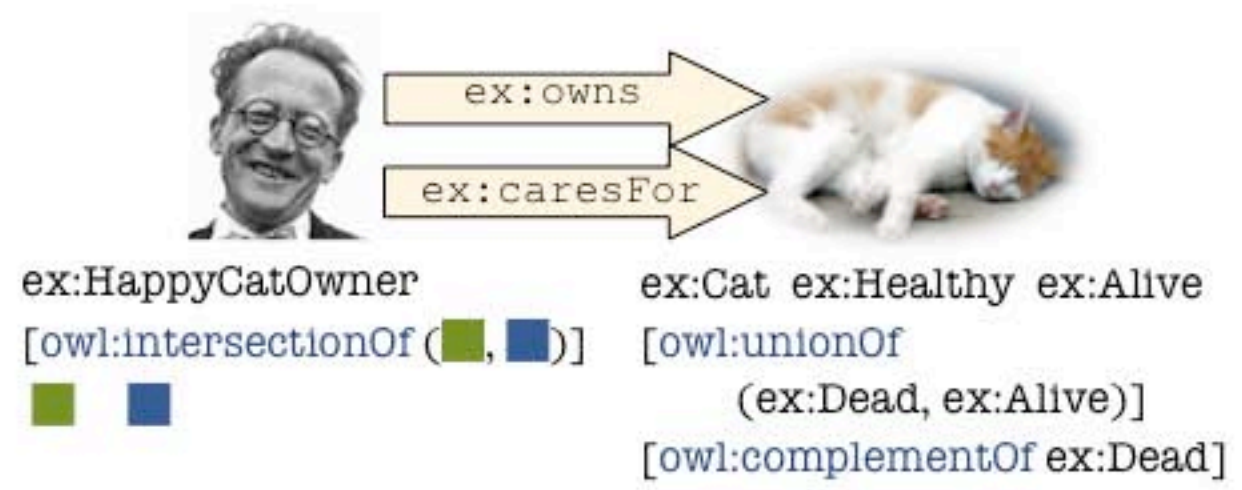
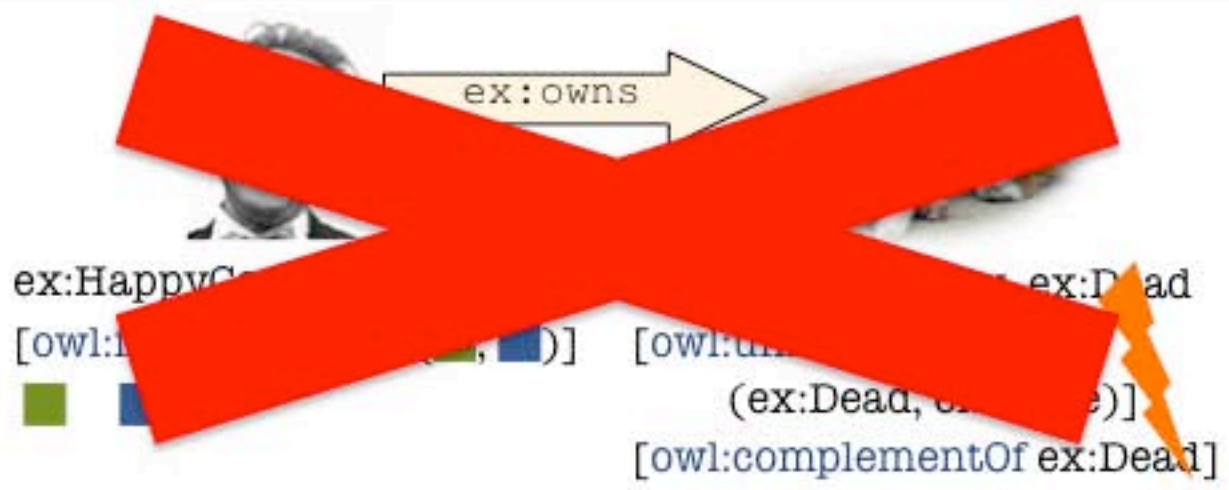
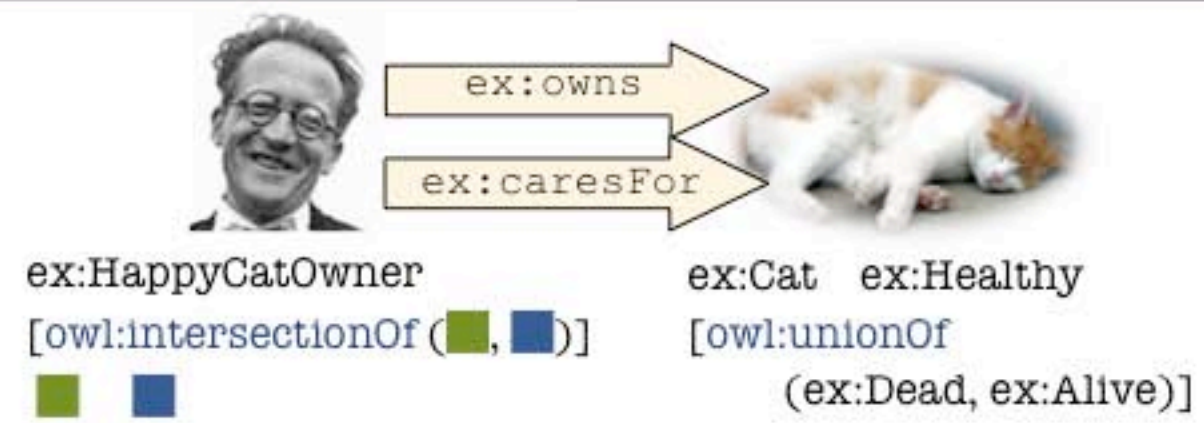
OWL Reasoning with Tableaux

Knowledge Base

```

ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (
  [rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],
  [rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:allValuesFrom ex:Healthy])] .
ex:schrödinger rdf:type ex:HappyCatOwner .
  
```

Tableau





OWL Reasoning with Tableaux

Knowledge Base

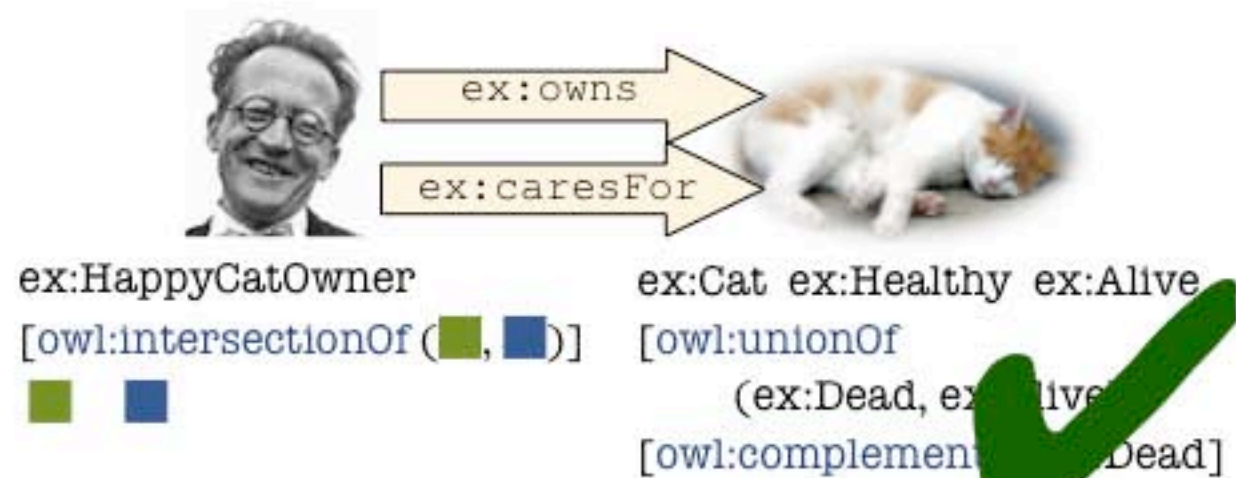
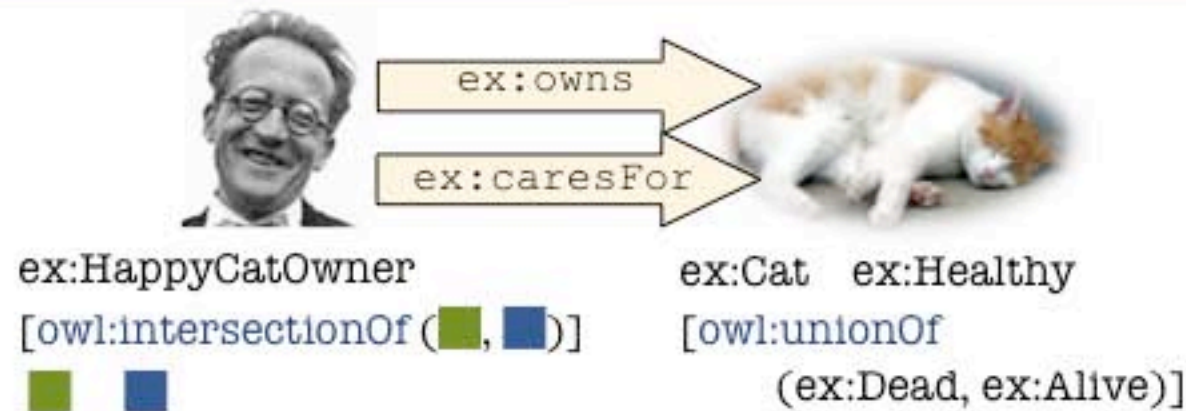
satisfiable

```

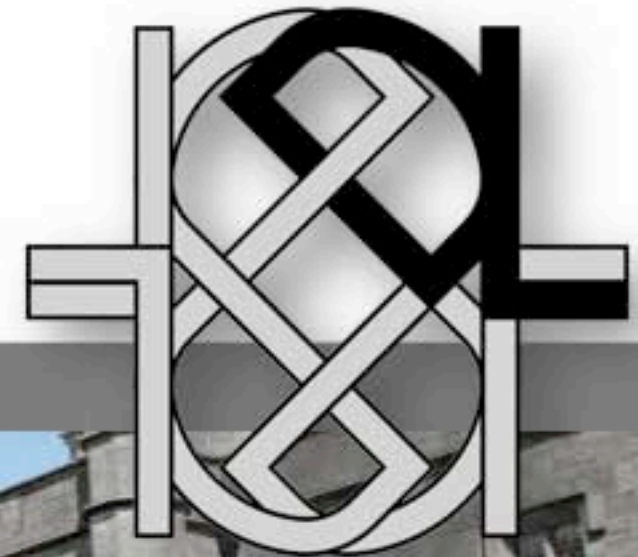
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (
  [rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],
  [rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:allValuesFrom ex:Healthy])] .
ex:schrödinger rdf:type ex:HappyCatOwner .

```

Tableau



Some Exercises



Institut für Angewandte Informatik und Formale Beschreibungsverfahren



RW2011
Galway City, Ireland
August 23 - August 27

Modeling

Exercise 22. *Come up with an \mathcal{ALC} GCI that expresses the following statement: “If an academic supervises a project, then he is a project leader and the project is a research project.” Use the role name `supervises` as well as the concept names `Academic`, `Project`, `ProjectLeader`, and `ResearchProject`.*

A Little Bit of Model Theory

Exercise 24. *Prove that the knowledge base*

$$(\forall \text{succ}^-. \top)(\text{zero}) \quad \top \sqsubseteq \exists \text{succ}. \top \quad \top \sqsubseteq \leq 1. \text{succ}^-. \top$$

cannot have a finite model.

Exercise 29. *As we have seen, $SR\mathcal{OIQ}$ allows to enforce that the domain size (i.e. the number of its elements) is at most n for any given $n \in \mathbb{N}$. Contemplate whether there is a knowledge base $\mathcal{KB}_{\text{fin}}$ that emulates finite models, i.e., for every knowledge base \mathcal{KB} not using vocabulary from $\mathcal{KB}_{\text{fin}}$ the models of $\mathcal{KB} \cup \mathcal{KB}_{\text{fin}}$ are exactly those models of \mathcal{KB} with finite domain, if one abstracts from the vocabulary of $\mathcal{KB}_{\text{fin}}$.*

Exercise 30. *Is it possible to create a $SH\mathcal{IQ}$ knowledge base \mathcal{KB} such that every model contains one individual which is connected via a role \mathbf{r} to infinitely many other individuals? Can the same be achieved in $ALCH\mathcal{OIQ}$? What about $ALCH\mathcal{IQ}$? For each of the cases either provide such a knowledge base or argue why this is not possible.*

Exercise 34. *To get a feeling for the relatedness between automata and DL reasoning, try to design an \mathcal{ALC} knowledge base \mathcal{KB} with the property that for any $r_1, r_2, \dots, r_n \in \mathbf{N}_R$ we have that $\mathcal{KB} \models A \sqsubseteq \exists r_1 \exists r_2 \dots \exists r_n . B$ exactly if the word $r_1 r_2 \dots r_n$ matches the regular expression $s^* (rs|srr)^*$.*