OWL 2 – Theory and Practice

Bernardo Cuenca Grau
University of Oxford
UK

Pascal Hitzler
Kno.e.sis Center
Wright State University
Dayton, OH, USA

Birte Glimm
University of Oxford
UK

Hector Perez-Urbina
Clark & Parsia, LLC

Today's Schedule

09:00 - 10:00 OWL introduction (Pascal)

10:00 - 10:30 coffee break

10:30 - 12:30 OWL introduction (Pascal)

12:30 - 14:00 lunch

14:00 - 16:00 hands-on session (Birte)

16:00 - 16:30 coffee break

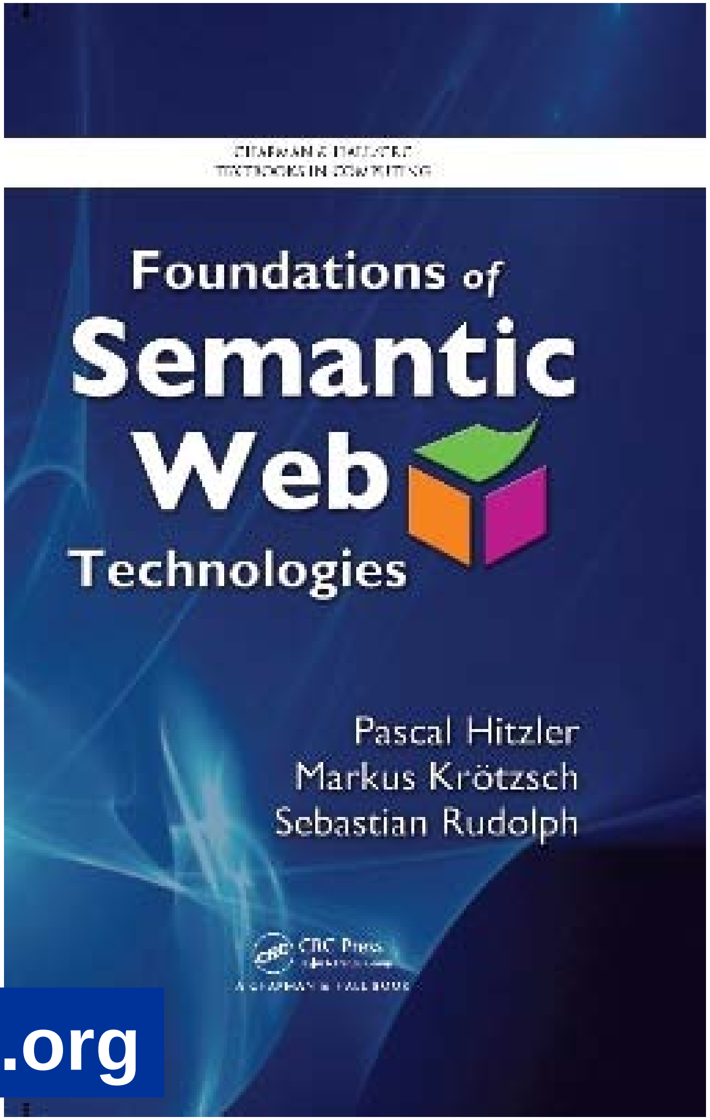
16:30 - 18:30 applications (Bernardo)

Textbook

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph

Foundations of Semantic Web Technologies
Chapman & Hall/CRC, 2009

Grab a flyer!



http://www.semantic-web-book.org

Textbook (Chinese translation)

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph

语义Web技术基础

Tsinghua University Press (清华大学出版社），2011, to appear

Translators:

Yong Yu, Haofeng Wang, Guilin Qi (俞勇，王昊奋，漆桂林)

http://www.semantic-web-book.org

Slides

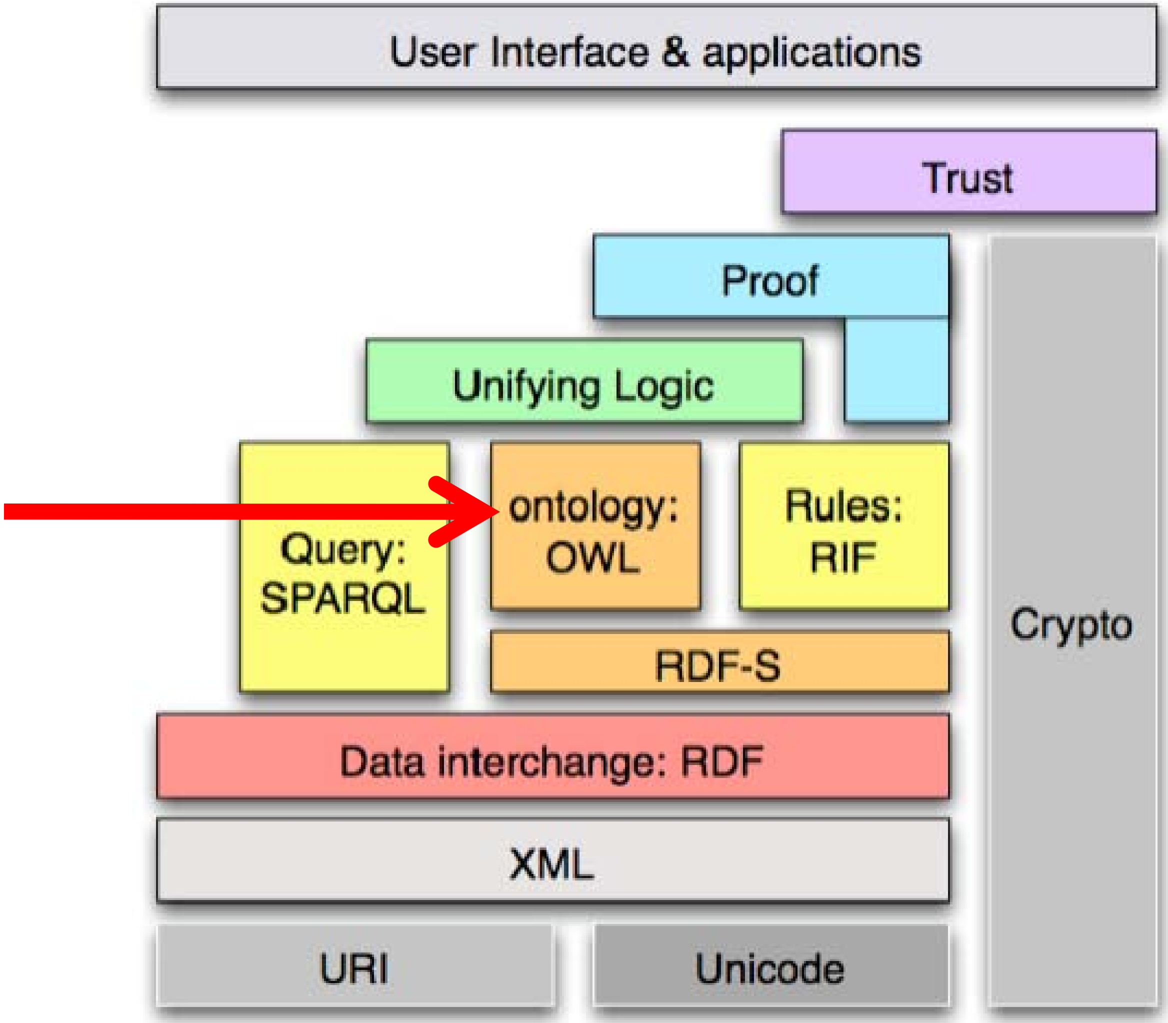
Available from

http://www.semantic-web-book.org/page/ISWC2010_Tutorial

Part 1

OWL 2 – Syntax, Semantics, Reasoning

OWL



User Interface & applications
Trust
Proof
Unifying Logic
Query: SPARQL
ontology: OWL
Rules: RIF
RDF-S
Crypto
Data interchange: RDF
XML
URI
Unicode

Main References Part 1

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, Foundations of Semantic Web Technologies, Chapman & Hall/CRC, 2009

OWL 2 Document Overview: http://www.w3.org/TR/owl2-overview/

Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, Sebastian Rudolph, OWL 2 Web Ontology Language: Primer. W3C Recommendation, 27 October 2009. http://www.w3.org/TR/owl2-primer/

OWL – Overview

- Web Ontology Language
  - W3C Recommendation for the Semantic Web, 2004
  - OWL 2 (revised W3C Recommendation), 2009

- Semantic Web KR language based on description logics (DLs)
  - OWL DL is essentially DL SROIQ(D)
  - KR for web resources, using URIs.
  - Using web-enabled syntaxes, e.g. based on XML or RDF. We present
    - DL syntax (used in research – not part of the W3C recommendation)
    - (some) RDF Turtle syntax

Contents

Contents

Rationale behind OWL

- Open World Assumption
- Favourable trade-off between expressivity and scalability
- Integrates with RDFS
- Purely declarative semantics

Features:

- Fragment of first-order predicate logic (FOL)
- Decidable
- Known complexity classes (N2ExpTime for OWL 2 DL)
- Reasonably efficient for real KBs

OWL Building Blocks

- individuals (written as URIs)
  - also: constants (FOL), resources (RDF)
  - http://example.org/sebastianRudolph
  - http://www.semantic-web-book.org
  - we write these lowercase and abbreviated, e.g. "sebastianRudolph"
- classes (also written as URIs!)
  - also: concepts, unary predicates (FOL)
  - we write these uppercase, e.g. "Father"
- properties (also written as URIs!)
  - also: roles (DL), binary predicates (FOL)
  - we write these lowercase, e.g. "hasDaughter"

DL syntax FOL syntax

ABox statements

- Person(mary)
- Person(mary)

- Woman $\sqsubseteq$ Person
  - Person ≡ HumanBeing
- $\forall x$ (Woman(x) $\rightarrow$ Person(x))

- hasWife(john,mary)
- hasWife(john,mary)

- hasWife $\sqsubseteq$ hasSpouse
  - hasSpouse ≡ marriedWith
- $\forall x \forall y$ (hasWife(x,y)$\rightarrow$ hasSpouse(x,y))

TBox statements

DL syntax FOL syntax

DL syntax	FOL syntax
Person(mary)	:mary rdf:type :Person .
Woman ⊑ Person – Person ≡ HumanBeing	:Woman rdfs:subClassOf :Person .
hasWife(john,mary)	:john :hasWife :mary .
hasWife ⊑ hasSpouse – hasSpouse ≡ marriedWith	:hasWife rdfs:subPropertyOf :hasSpouse .

Special classes and properties

- owl:Thing (RDF syntax)
  - DL-syntax: $\top$
  - contains everything
- owl:Nothing (RDF syntax)
  - DL-syntax: $\bot$
  - empty class
- owl:topProperty (RDF syntax)
  - DL-syntax: U
  - every pair is in U
- owl:bottomProperty (RDF syntax)
  - empty property

Class constructors

- **conjunction**
  - **Mother ≡ Woman ⊓ Parent**
  - **:Mother owl:equivalentClass _:x .**
    _:x rdf:type owl:Class .
    _:x owl:intersectionOf (:Woman :Parent) .

$\forall x\ (\mathrm{Mother}(x) \leftrightarrow \mathrm{Woman}(x) \wedge \mathrm{Parent}(x))$

- **disjunction**
  - **Parent ≡ Mother ⊔ Father**
  - **:Parent owl:equivalentClass _:x .**
    _:x rdf:type owl:Class .
    _:x owl:unionOf (:Mother :Father) .

$\forall x\ (\mathrm{Parent}(x) \leftrightarrow \mathrm{Mother}(x) \vee \mathrm{Father}(x))$

- **negation**
  - **ChildlessPerson ≡ Person ⊓ ¬Parent**
  - **:ChildlessPerson owl:equivalentClass _:x .**
    _:x rdf:type owl:Class .
    _:x owl:intersectionOf (:Person _:y) .
    _:y owl:complementOf :Parent .

$\forall x\ (\mathrm{ChildlessPerson}(x) \leftrightarrow \mathrm{Person}(x) \wedge \neg\mathrm{Parent}(x))$

Class constructors

- **existential quantification**
  - **only to be used with a role – also called a *property restriction***
  - **Parent $\equiv \exists$hasChild.Person**
  - **:Parent owl:equivalentClass _:x .**
    _:x rdf:type owl:Restriction .
    _:x owl:onProperty :hasChild .
    _:x owl:someValuesFrom :Person .

$$\forall x\, (\mathrm{Parent}(x) \leftrightarrow \exists y\, (\mathrm{hasChild}(x,y) \wedge \mathrm{Person}(y)))$$

- **universal quantification**
  - **only to be used with a role – also called a *property restriction***
  - **Person $\sqcap$ Happy $\equiv \forall$hasChild.Happy**
  - **_:x rdf:type owl:Class .**
    _:x owl:intersectionOf (:Person :Happy) .
    _:x owl:equivalentClass _:y .
    _:y rdf:type owl:Restriction .
    _:y owl:onProperty :hasChild .
    _:y owl:allValuesFrom :Happy .

$$\forall x\, (\mathrm{Person}(x) \wedge \mathrm{Happy}(x) \leftrightarrow \forall y\, (\mathrm{hasChild}(x,y) \rightarrow \mathrm{Happy}(y)))$$

- **Class constructors can be nested arbitrarily**

Contents

- OWL – Basic Ideas
- OWL as the Description Logic SROIQ(D)
- Different Perspectives on OWL
- OWL Semantics
- OWL Profiles
- Proof Theory
- Tools

Understanding SROIQ(D)

The description logic ALC

Complexity: ExpTime

- **ABox expressions:**
  Individual assignments **Father(john)**
  Property assignments **hasWife(john,mary)**

- **TBox expressions**
  subclass relationships $\sqsubseteq$

  conjunction $\sqcap$
  disjunction $\sqcup$
  negation $\neg$

  property restrictions $\forall$
  $\exists$

Also: $\top$, $\bot$

Understanding SROIQ(D)

ALC + role chains = SR

- **hasParent o hasBrother $\sqsubseteq$ hasUncle**

  $\forall x\ \forall y\ (\exists z\ ((hasParent(x,z) \wedge hasBrother(z,y)) \rightarrow hasUncle(x,y)))$

  - **includes top property and bottom property**

- **includes S = ALC + transitivity**
  - **hasAncestor o hasAncestor $\sqsubseteq$ hasAncestor**

- **includes SH = S + role hierarchies**
  - **hasFather $\sqsubseteq$ hasParent**

Understanding SROIQ(D)

- O – nominals (closed classes)
  - MyBirthdayGuests $\equiv$ {bill,john,mary}
  - Note the difference to
    MyBirthdayGuests(bill)
    MyBirthdayGuests(john)
    MyBirthdayGuests(mary)

- Individual equality and inequality (no unique name assumption!)
  - bill = john
    - $\{bill\} \equiv \{john\}$
  - bill $\neq$ john
    - $\{bill\} \sqcap \{john\} \equiv \bot$

Understanding SROIQ(D)

- I – inverse roles
  - $\text{hasParent} \equiv \text{hasChild}^{-}$
  - $\text{Orphan} \equiv \forall \text{hasChild}^{-}.\text{Dead}$

- Q – qualified cardinality restrictions
  - $\leq 4\ \text{hasChild}.\text{Parent}(\text{john})$
  - $\text{HappyFather} \equiv\ \geq 2\ \text{hasChild}.\text{Female}$
  - $\text{Car} \sqsubseteq\ =4 \text{hasTyre}.\top$

- Complexity SHIQ, SHOQ, SHIO: ExpTime.
  Complexity SHOIQ: NExpTime
  Complexity SROIQ: N2ExpTime

Understanding SROIQ(D)

Properties can be declared to be

• Transitive	hasAncestor
• Symmetric	hasSpouse
• Asymmetric	hasChild
• Reflexive	hasRelative
• Irreflexive	parentOf
• Functional	hasHusband
• InverseFunctional	hasHusband

called *property characteristics*

Understanding SROIQ(D)

(D) – datatypes

- **so far, we have only seen properties with individuals in second argument, called *object properties* or *abstract roles* (DL)**

- **properties with datatype literals in second argument are called *data properties* or *concrete roles* (DL)**

- **allowed are many XML Schema datatypes, including xsd:integer, xsd:string, xsd:float, xsd:booelan, xsd:anyURI, xsd:dateTime**

  and also e.g. owl:real

Understanding SROIQ(D)

(D) – datatypes

- **hasAge(john, "51"^^xsd:integer)**
- **additional use of *constraining facets* (from XML Schema)**
  - **e.g. Teenager $\equiv$ Person $\sqcap$ $\exists$hasAge.(xsd:integer: $\geq$12 and $\leq$19)**
    note: this is not standard DL notation!

Understanding SROIQ(D)

further expressive features

- Self
  - PersonCommittingSuicide $\equiv$ $\exists$kills.Self
- Keys (not really in SROIQ(D), but in OWL)
  - set of (object or data) properties whose values uniquely identify an object
- disjoint properties
  - Disjoint(hasParent,hasChild)
- explicit anonymous individuals
  - as in RDF: can be used instead of named individuals

SROIQ(D) constructors – overview

- ABox assignments of individuals to classes or properties
- ALC: $\sqsubseteq$, $\equiv$ for classes
  $\sqcap$, $\sqcup$, $\neg$, $\exists$, $\forall$
  $\top$, $\bot$
- SR: + property chains, property characteristics, role hierarchies $\sqsubseteq$
- SRO: + nominals {o}
- SROI: + inverse properties
- SROIQ: + qualified cardinality constraints
- SROIQ(D): + datatypes (including facets)

- + top and bottom roles (for objects and datatypes)
- + disjoint properties
- + Self
- + Keys (not in SROIQ(D), but in OWL)

Some Syntactic Sugar in OWL

This applies to the non-DL syntaxes (e.g. RDF syntax).

- **disjoint classes**
  - **Apple $\sqcap$ Pear $\sqsubseteq \bot$**
- **disjoint union**
  - **Parent $\equiv$ Mother $\sqcup$ Father**
    Mother $\sqcap$ Father $\sqsubseteq \bot$
- **negative property assignments (also for datatypes)**
  - **¬hasAge(jack,"53"^^xsd:integer)**

Contents

OWL – Extralogical Features

- **OWL ontologies have URIs and can be referenced by others via**
  - **import statements**
- **Namespace declarations**
- **Entity declarations (must be done)**
- **Versioning information etc.**

- **Annotations**
  - **Entities and axioms (statements) can be endowed with annotations, e.g. using rdfs:comment.**
  - **OWL syntax provides *annotation properties* for this purpose.**

The modal logic perspective

- Description logics can be understood from a modal logic perspective.
- Each pair of ∀R and ∃R statements give rise to a pair of modalities.
- Essentially, some description logics are multi-modal logics.
- See e.g. Baader et al., The Description Logic Handbook, Cambridge University Press, 2007.

The RDFS perspective

RDFS semantics is weaker

- :mary rdf:type :Person .
- :Mother rdfs:subClassOf :Woman .
- :john :hasWife :Mary .
- :hasWife rdfs:subPropertyOf :hasSpouse

- Person(mary)
- Mother $\sqsubseteq$ Woman
- hasWife(john,mary)
- hasWife $\sqsubseteq$ hasSpouse

- :hasWife rdfs:range :Woman .
- :hasWife rdfs:domain :Man .

- $\top \sqsubseteq \forall$hasWife.Woman
- $\top \sqsubseteq \forall$hasWife$^-$.Man or $\exists$hasWife.$\top \sqsubseteq$ Man

RDFS also allows to

- make statements about statements
  $\rightarrow$ only possible through annotations in OWL
- mix class names, individual names, property names (they are all URIs)
  $\rightarrow$ *punning* in OWL

Punning

- **Description logics impose *type separation*, i.e. names of individuals, classes, and properties must be disjoint.**
- **In OWL 2 Full, type separation does not apply.**
- **In OWL 2 DL, type separation is relaxed, but a class X and an individual X are interpreted semantically as if they were different.**
- **Father(john)**
  SocialRole(Father)
- **See further below on the two different semantics for OWL.**

Contents

- OWL – Basic Ideas
- OWL As the Description Logic SROIQ(D)
- Different Perspectives on OWL
- OWL Semantics
- OWL Profiles
- Proof Theory
- Tools

OWL Semantics

- There are two semantics for OWL.

1. Description Logic Semantics
   also: Direct Semantics; FOL Semantics
   Can be obtained by translation to FOL.
   Syntax restrictions apply! (see next slide)

2. RDF-based Semantics
   No syntax restrictions apply.
   Extends the direct semantics with RDFS-reasoning features.

In the following, we will deal with the direct semantics only.

OWL Direct Semantics

To obtain decidability, syntactic restrictions apply.

- Type separation / punning
- No cycles in property chains.
- No transitive properties in cardinality restrictions.

OWL Direct Semantics: Restrictions

- arbitrary property chain axioms lead to undecidability
- restriction: set of property chain axioms has to be *regular*
  - there must be a strict linear order $\prec$ on the properties
  - every property chain axiom has to have one of the following forms:

    $R \circ R \sqsubseteq R$ $\quad$ $S^- \sqsubseteq R$ $\quad$ $S_1 \circ S_2 \circ \dots \circ S_n \sqsubseteq R$

    $R \circ S_1 \circ S_2 \circ \dots \circ S_n \sqsubseteq R$ $\quad$ $S_1 \circ S_2 \circ \dots \circ S_n \circ R \sqsubseteq R$
  - thereby, $S_i \prec R$ for all i= 1, 2, . . . , *n.*

- Example 1: $R \circ S \sqsubseteq R$ $\quad$ $S \circ S \sqsubseteq S$ $\quad$ $R \circ S \circ R \sqsubseteq T$

  → regular with order $S \prec R \prec T$
- Example 2: $R \circ T \circ S \sqsubseteq T$

  → not regular because form not admissible
- Example 3: $R \circ S \sqsubseteq S$ $\quad$ $S \circ R \sqsubseteq R$

  → not regular because no adequate order exists

OWL Direct Semantics: Restrictions

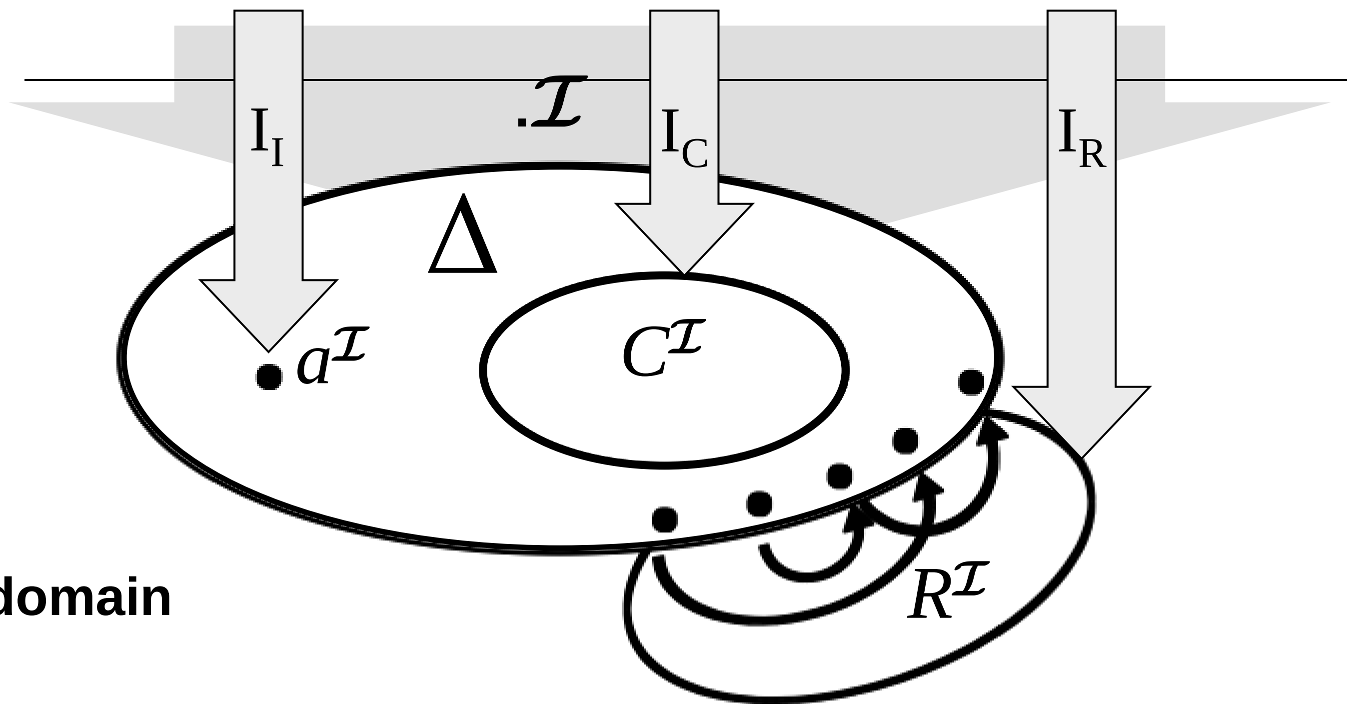
- **combining property chain axioms and cardinality constraints may lead to undecidability**
- **restriction: use only *simple* properties in cardinality expressions (i.e. those which cannot be – directly or indirectly – inferred from property chains)**
- **technically:**
  - **for any property chain axiom $S_1 \circ S_2 \circ_{\dots} \circ S_n \sqsubseteq R$ with n>1, R is non-simple**
  - **for any subproperty axiom $S \sqsubseteq R$ with S non-simple, R is non-simple**
  - **all other properties are simple**
- **Example: $Q \circ P \sqsubseteq R$ $R \circ P \sqsubseteq R$ $R \sqsubseteq S$ $P \sqsubseteq R$ $Q \sqsubseteq S$**

  non-simple: R, S simple: P, Q

OWL Direct Semantics

- **model-theoretic semantics**
- **starts with interpretations**
- **an interpretation $\mathcal{I}$ maps**

  individual names, class names and property names...



...into a domain

Interpretation Example

If we consider, for example, the knowledge base consisting of the axioms

$$\begin{aligned}&\texttt{Professor} \sqsubseteq \texttt{FacultyMember}\\&\texttt{Professor}(\texttt{rudiStuder})\\&\texttt{hasAffiliation}(\texttt{rudiStuder}, \texttt{aifb})\end{aligned}$$

then we could set

$$\begin{aligned}\Delta &= \{a, b, \mathrm{Ian}\}\\ \mathrm{I}_{\mathbf{I}}(\texttt{rudiStuder}) &= \mathrm{Ian}\\ \mathrm{I}_{\mathbf{I}}(\texttt{aifb}) &= b\\ \mathrm{I}_{\mathbf{C}}(\texttt{Professor}) &= \{a\}\\ \mathrm{I}_{\mathbf{C}}(\texttt{FacultyMember}) &= \{a, b\}\\ \mathrm{I}_{\mathbf{R}}(\texttt{hasAffiliation}) &= \{(a, b), (b, \mathrm{Ian})\}\end{aligned}$$

Intuitively, these settings are nonsense, but they nevertheless determine a valid interpretation.

OWL Direct Semantics

- mapping is extended to complex class expressions:
  - $\top^I = \Delta^I$ $\quad\quad\quad$ $\bot^I = \emptyset$
  - $(C \sqcap D)^I = C^I \cap D^I$ $\quad\quad$ $(C \sqcup D)^I = C^I \cup D^I$ $\quad\quad$ $(\neg C)^I = \Delta^I \setminus C^I$
  - $(\forall R.C)^I = \{ x \mid \text{for all } (x,y) \in R^I \text{ we have } y \in C^I \}$
    $(\exists R.C)^I = \{ x \mid \text{there is } (x,y) \in R^I \text{ with } y \in C^I \}$
  - $(\geq nR.C)^I = \{ x \mid \#\{ y \mid (x,y) \in R^I \text{ and } y \in C^I \} \geq n \}$
  - $(\leq nR.C)^I = \{ x \mid \#\{ y \mid (x,y) \in R^I \text{ and } y \in C^I \} \leq n \}$
- ...and to role expressions:
  - $U^I = \Delta^I \times \Delta^I$ $\quad\quad\quad$ $(R^-)^I = \{ (y,x) \mid (x,y) \in R^I \}$
- ...and to axioms:
  - $C(a)$ holds, if $a^I \in C^I$ $\quad\quad$ $R(a,b)$ holds, if $(a^I,b^I) \in R^I$
  - $C \sqsubseteq D$ holds, if $C^I \subseteq D^I$ $\quad\quad$ $R \sqsubseteq S$ holds, if $R^I \subseteq S^I$
  - Disjoint(R,S) holds if $R^I \cap S^I = \emptyset$
  - $S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R$ holds if $S_1^I \circ S_2^I \circ \ldots \circ S_n^I \subseteq R^I$

OWL Direct Semantics

- **what's below gives us a notion of *model*:**

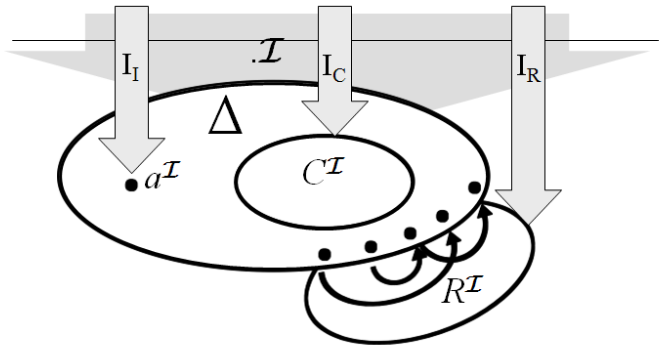
  An interpretation is a model of a set of axioms if all the axioms hold (are evaluated to true) in the interpretation.

- **Notion of *logical consequence* obtained via models (below).**

- **...and to axioms:**
  - $C(a)$ holds, if $a^I \in C^I$ — $R(a,b)$ holds, if $(a^I,b^I) \in R^I$
  - $C \sqsubseteq D$ holds, if $C^I \subseteq D^I$ — $R \sqsubseteq S$ holds, if $R^I \subseteq S^I$
  - $\mathrm{Disjoint}(R,S)$ holds if $R^I \cap S^I = \emptyset$
  - $S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R$ holds if $S_1^I \circ S_2^I \circ \ldots \circ S_n^I \subseteq R^I$

Logical Consequence

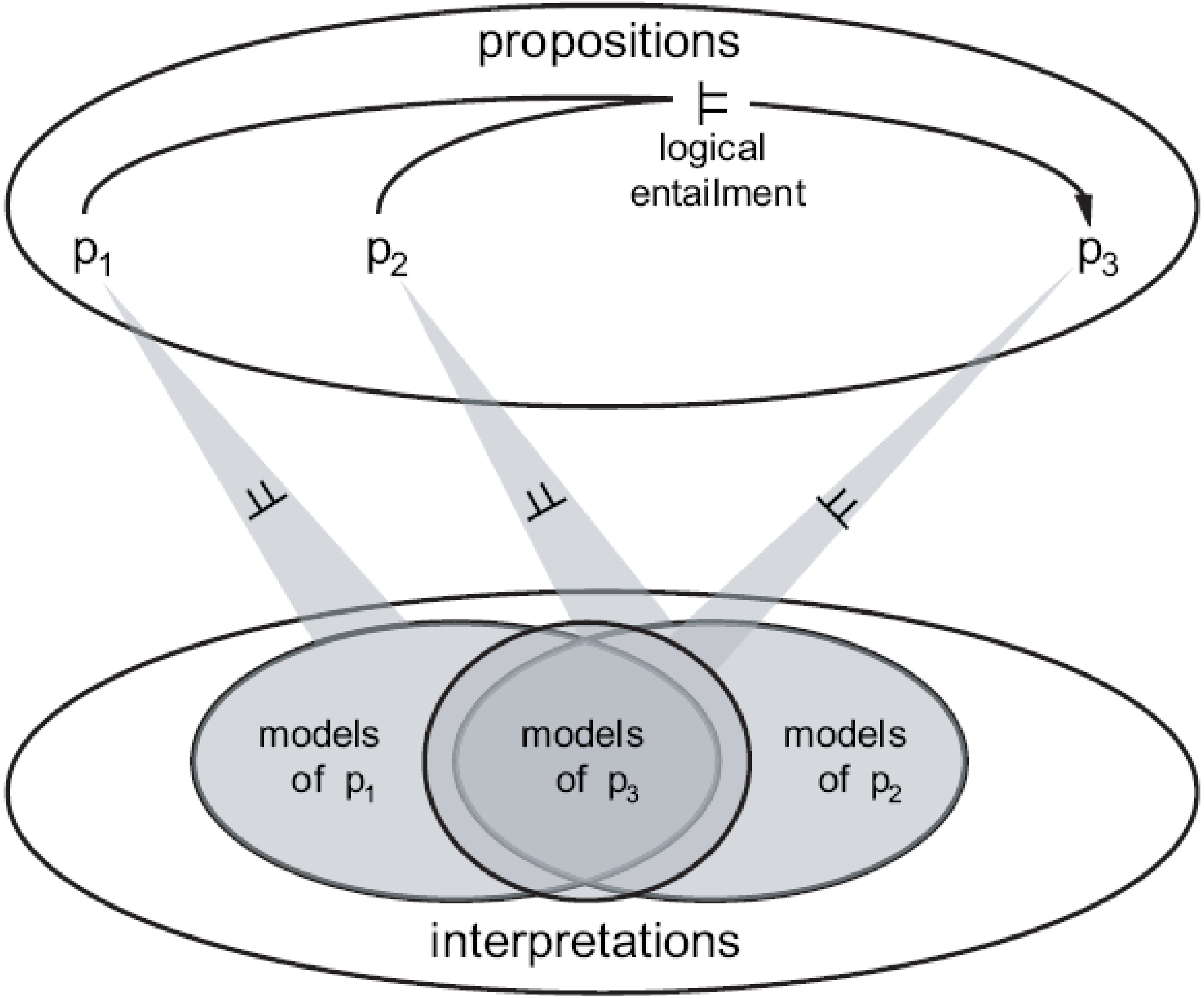
A *model* for an OWL KB is such a mapping I which satisfies all axioms in the KB.



An axiom α is a *logical consequence* of a KB if every model of the KB is also a model of α.

The logical consequences of a KB are all those things which are *necessarily the case in all „realities“* in which the KB is the case.

Notion of logical consequence



Not a model!

If we consider, for example, the knowledge base consisting of the axioms

$$\begin{aligned}\mathtt{Professor} &\sqsubseteq \mathtt{FacultyMember}\\ &\mathtt{Professor}(\mathtt{rudiStuder})\\ &\mathtt{hasAffiliation}(\mathtt{rudiStuder}, \mathtt{aifb})\end{aligned}$$

then we could set

$$\begin{aligned}\Delta &= \{a, b, \mathrm{Ian}\}\\ \mathrm{I}_{\mathbf{I}}(\mathtt{rudiStuder}) &= \mathrm{Ian}\\ \mathrm{I}_{\mathbf{I}}(\mathtt{aifb}) &= b\\ \mathrm{I}_{\mathbf{C}}(\mathtt{Professor}) &= \{a\}\\ \mathrm{I}_{\mathbf{C}}(\mathtt{FacultyMember}) &= \{a, b\}\\ \mathrm{I}_{\mathbf{R}}(\mathtt{hasAffiliation}) &= \{(a, b), (b, \mathrm{Ian})\}\end{aligned}$$

Intuitively, these settings are nonsense, but they nevertheless determine a valid interpretation.

A model

$$\mathtt{Professor} \sqsubseteq \mathtt{FacultyMember}$$
$$\mathtt{Professor}(\mathtt{rudiStuder})$$
$$\mathtt{hasAffiliation}(\mathtt{rudiStuder}, \mathtt{aifb})$$

$$\begin{aligned}
\Delta &= \{a, r, s\} \\
\mathrm{I}_{\mathbf{I}}(\mathtt{rudiStuder}) &= r \\
\mathrm{I}_{\mathbf{I}}(\mathtt{aifb}) &= a \\
\mathrm{I}_{\mathbf{C}}(\mathtt{Professor}) &= \{r\} \\
\mathrm{I}_{\mathbf{C}}(\mathtt{FacultyMember}) &= \{r, s\} \\
\mathrm{I}_{\mathbf{R}}(\mathtt{hasAffiliation}) &= \{(r, a)\}
\end{aligned}$$

Models

$$\texttt{Professor} \sqsubseteq \texttt{FacultyMember}$$
$$\texttt{Professor}(\texttt{rudiStuder})$$
$$\texttt{hasAffiliation}(\texttt{rudiStuder}, \texttt{aifb})$$

	Model 1	Model 2	Model 3
Δ	$\{a, r, s\}$	$\{1, 2\}$	$\{\spadesuit\}$
$\mathrm{I_I}(\texttt{rudiStuder})$	r	1	$\spadesuit$
$\mathrm{I_I}(\texttt{aifb})$	a	2	$\spadesuit$
$\mathrm{I_C}(\texttt{Professor})$	$\{r\}$	$\{1\}$	$\{\spadesuit\}$
$\mathrm{I_C}(\texttt{FacultyMember})$	$\{a, r, s\}$	$\{1, 2\}$	$\{\spadesuit\}$
$\mathrm{I_R}(\texttt{hasAffiliation})$	$\{(r, a)\}$	$\{(1, 1), (1, 2)\}$	$\{(\spadesuit, \spadesuit)\}$

Is FacultyMember(aifb) a logical consequence?

Counterexample

Returning to our running example knowledge base, let us show formally that $\texttt{FacultyMember}(\texttt{aifb})$ is not a logical consequence. This can be done by giving a model M of the knowledge base where $\texttt{aifb}^M \notin \texttt{FacultyMember}^M$. The following determines such a model.

$$
\begin{aligned}
\Delta &= \{a, r\} \\
I_{\mathbf{I}}(\texttt{rudiStuder}) &= r \\
I_{\mathbf{I}}(\texttt{aifb}) &= a \\
I_{\mathbf{C}}(\texttt{Professor}) &= \{r\} \\
I_{\mathbf{C}}(\texttt{FacultyMember}) &= \{r\} \\
I_{\mathbf{R}}(\texttt{hasAffiliation}) &= \{(r, a)\}
\end{aligned}
$$

Logical Consequence

$$\texttt{Professor} \sqsubseteq \texttt{FacultyMember}$$
$$\texttt{Professor}(\texttt{rudiStuder})$$
$$\texttt{hasAffiliation}(\texttt{rudiStuder}, \texttt{aifb})$$

	Model 1	Model 2	Model 3
Δ	$\{a, r, s\}$	$\{1, 2\}$	$\{\spadesuit\}$
$\mathrm{I}_\mathrm{I}(\texttt{rudiStuder})$	r	1	$\spadesuit$
$\mathrm{I}_\mathrm{I}(\texttt{aifb})$	a	2	$\spadesuit$
$\mathrm{I}_\mathrm{C}(\texttt{Professor})$	$\{r\}$	$\{1\}$	$\{\spadesuit\}$
$\mathrm{I}_\mathrm{C}(\texttt{FacultyMember})$	$\{a, r, s\}$	$\{1, 2\}$	$\{\spadesuit\}$
$\mathrm{I}_\mathrm{R}(\texttt{hasAffiliation})$	$\{(r, a)\}$	$\{(1, 1), (1, 2)\}$	$\{(\spadesuit, \spadesuit)\}$

Is FacultyMember(rudiStuder) a logical consequence?

OWL Direct Semantics via FOL

- **but often OWL 2 DL is said to be a fragment of first-order predicate logic (FOL) [with equality]...**
- **yes, there is a translation of OWL 2 DL into FOL**

$$\pi(C \sqsubseteq D) = (\forall x)(\pi_x(C) \to \pi_x(D))$$
$$\pi_x(A) = A(x)$$
$$\pi_x(\neg C) = \neg \pi_x(C)$$
$$\pi_x(C \sqcap D) = \pi_x(C) \wedge \pi_x(D)$$
$$\pi_x(C \sqcup D) = \pi_x(C) \vee \pi_x(D)$$
$$\pi_x(\forall R.C) = (\forall x_1)(R(x, x_1) \to \pi_{x_1}(C))$$
$$\pi_x(\exists R.C) = (\exists x_1)(R(x, x_1) \wedge \pi_{x_1}(C))$$
$$\pi_x(\geq nS.C) = (\exists x_1)\dots(\exists x_n)\left(\bigwedge_{i \neq j}(x_i \neq x_j) \wedge \bigwedge_i (S(x, x_i) \wedge \pi_{x_i}(C))\right)$$
$$\pi_x(\leq nS.C) = \neg(\exists x_1)\dots(\exists x_{n+1})\left(\bigwedge_{i \neq j}(x_i \neq x_j) \wedge \bigwedge_i (S(x, x_i) \wedge \pi_{x_i}(C))\right)$$
$$\pi_x(\{a\}) = (x = a)$$
$$\pi_x(\exists S.\mathrm{Self}) = S(x, x)$$

$$\pi(R_1 \sqsubseteq R_2) = (\forall x)(\forall y)(\pi_{x,y}(R_1) \to \pi_{x,y}(R_2))$$
$$\pi_{x,y}(S) = S(x, y)$$
$$\pi_{x,y}(R^-) = \pi_{y,x}(R)$$
$$\pi_{x,y}(R_1 \circ \dots \circ R_n) = (\exists x_1)\dots(\exists x_{n-1})\left(\pi_{x,x_1}(R_1) \wedge \bigwedge_{i=1}^{n-2} \pi_{x_i,x_{i+1}}(R_{i+1}) \wedge \pi_{x_{n-1},y}(R_n)\right)$$
$$\pi(\mathrm{Ref}(R)) = (\forall x)\pi_{x,x}(R)$$
$$\pi(\mathrm{Asy}(R)) = (\forall x)(\forall y)(\pi_{x,y}(R) \to \neg \pi_{y,x}(R)$$
$$\pi(\mathrm{Dis}(R_1, R_2)) = \neg(\exists x)(\exists y)(\pi_{x,y}(R_1) \wedge \pi_{x,y}(R_2))$$

- **...which (interpreted under FOL semantics) coincides with the definition just given.**

Inconsistency and Satisfiability

- A set of axioms (knowledge base) is satisfiable (or consistent) if it has a model.
- It is unsatisfiable (inconsistent) if it does not have a model.

- Inconsistency is often caused by modeling errors.

- Unicorn(beauty)
  Unicorn $\sqsubseteq$ Fictitious
  Unicorn $\sqsubseteq$ Animal
  Animal $\sqsubseteq$ $\neg$Fictitious

Inconsistency and Satisfiability

- **A knowledge base is incoherent if a named class is equivalent to $\bot$.**
- **It usually also points to a modeling error.**

$$\begin{aligned}\mathtt{Unicorn} &\sqsubseteq \mathtt{Fictitious}\\ \mathtt{Unicorn} &\sqsubseteq \mathtt{Animal}\\ \mathtt{Fictitious} \sqcap \mathtt{Animal} &\sqsubseteq \bot\end{aligned}$$

A Semantic Puzzle

From Horridge, Parsia, Sattler, From Justifications to Proofs for Entailments in OWL. In: Proceedings OWLED2009.
http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-529/

$$\mathsf{Person} \sqsubseteq \neg\mathsf{Movie}$$

$$\mathsf{RRated} \sqsubseteq \mathsf{CatMovie}$$

$$\mathsf{CatMovie} \sqsubseteq \mathsf{Movie}$$

$$\mathsf{RRated} \equiv (\exists \mathsf{hasScript}.\mathsf{ThrillerScript}) \sqcup (\forall \mathsf{hasViolenceLevel}.\mathsf{High})$$

$$\mathsf{Domain}(\mathsf{hasViolenceLevel},\ \mathsf{Movie})$$

Fig. 1. A justification for $\mathsf{Person} \sqsubseteq \bot$

What Semantics Is Good For

- **Opinions Differ. Here's my take.**

- **Semantic Web requires a shareable, declarative and *computable* semantics.**
- **I.e., the semantics must be a formal entity which is clearly defined and automatically computable.**

- **Ontology languages provide this by means of their formal semantics.**
- **Semantic Web Semantics is given by a relation – the *logical consequence* relation.**

- **Note: This is considerably more than saying that the semantics of an ontology is the set of its logical consequences!**

In other words

We capture the meaning of information

not by specifying its meaning (which is impossible)
but by specifying

how information interacts with other information.

We describe the meaning indirectly through its effects.

Simple Logical Reasoning

If I ask for soccer team members, I also want to get the goalkeepers listed ...

If I ask for cities, I also want all capitals listed ...

inheritance reasoning

Less Simple Reasoning

What was again the name of that russian researcher who worked on resolution-based calculi for EL?

answering requires merging of knowledge from many websites and using background knowledge.

Are lobsters spiders?

What is "Käuzchen" in english?

SNOMED CT

- **SNOMED CT: commercial ontology, medical domain ca. 300,000 axioms**

- **InjuryOfFinger $\equiv$ Injury $\sqcap$ $\exists$site.Finger$_S$**
  InjuryOfHand $\equiv$ Injury $\sqcap$ $\exists$site.Hand$_S$
  Finger$_S$ $\sqsubseteq$ Hand$_P$
  Hand$_P$ $\sqsubseteq$ Hand$_S$ $\sqcap$ $\exists$part.Hand$_E$

- **Reasoning has been used e.g. for**
  - **classification (computing the hidden taxonomy) e.g., InjuryOfFinger $\sqsubseteq$ InjuryOfHand**
  - **bug finding**

Contents

- OWL – Basic Ideas
- OWL As the Description Logic SROIQ(D)
- Different Perspectives on OWL
- OWL Semantics
- OWL Profiles
- Proof Theory
- Tools

OWL Profiles

- **OWL Full – using the RDFS-based semantics**
- **OWL DL – using the FOL semantics**

The OWL 2 documents describe further profiles, which are of polynomial complexity:

- **OWL EL (EL++)**
- **OWL QL (DL Lite$_R$)**
- **OWL RL (DLP)**

OWL 2 EL

- **allowed:**
  - **subclass axioms with intersection, existential quantification, top, bottom**
    - **closed classs must have only one member**
  - **property chain axioms, range restrictions (under certain conditions)**
- **disallowed:**
  - **negation, disjunction, arbitrary universal quantification, role inverses**

$$\sqcap \exists \top \bot \sqsubseteq \sqcap \exists \top \bot$$

- **Examples:** $\text{Human} \sqsubseteq \exists \text{hasParent}.\text{Person}$
  $\exists \text{married}.\top \sqcap \text{CatholicPriest} \sqsubseteq \bot$;
  $\text{hasParent} \circ \text{hasParent} \sqsubseteq \text{hasGrandparent}$

OWL 2 RL

- **Motivated by the question: what fraction of OWL 2 DL can be expressed naively by rules (with equality)?**
- **Examples:**
  - **$\exists$parentOf.$\exists$parentOf.$\top \sqsubseteq$ Grandfather**
    rule version: parentOf(x,y) parentOf(y,z) $\rightarrow$ Grandfather(x)
  - **Orphan $\sqsubseteq$ $\forall$hasParent.Dead**
    rule version: Orphan(x) hasParent(x,y) $\rightarrow$ Dead(y)
  - **Monogamous $\sqsubseteq$ $\leq$1married.Alive**
    rule version:
    Monogamous(x) married(x,y) Alive(y) married(x,z) Alive(z)$\rightarrow$ y=z
  - **childOf $\circ$ childOf $\sqsubseteq$ grandchildOf**
    rule version: childOf(x,y) childOf(y,z) $\rightarrow$ grandchildOf(x,z)
  - **Disj(childOf,parentOf)**
    rule version: childOf(x,y) parentOf(x,y) $\rightarrow$

OWL 2 RL

- **Syntactic characterization:**
  - **essentially, all axiom types are allowed**
  - **disallow certain constructors on lhs and rhs of subclass statements**

$$\cancel{\forall}\ \cancel{\neg}\ \sqsubseteq\ \cancel{\exists}\ \cancel{\sqcup}$$

  - **cardinality restrictions: only on rhs and only ≤1 and ≤0 allowed**
  - **closed classes: only with one member**

- **Reasoner conformance requires only soundness.**

OWL 2 QL

- **Motivated by the question: what fraction of OWL 2 DL can be captured by standard database technology?**
- **Formally: query answering LOGSPACE w.r.t. data (via translation into SQL)**
- **Allowed:**
  - **subproperties, domain, range**
  - **subclass statements with**
    - **left hand side: class name or expression of type $\exists r.\top$**
    - **right hand side: intersection of class names, expressions of type $\exists r.C$ and negations of lhs expressions**
    - **no closed classes!**
- **Example:**

$$\exists \mathbf{married}.\top \sqsubseteq \neg \mathbf{Free} \sqcap \exists \mathbf{has}.\mathbf{Sorrow}$$

Contents

- OWL – Basic Ideas
- OWL As the Description Logic SROIQ(D)
- Different Perspectives on OWL
- OWL Semantics
- OWL Profiles
- Proof Theory
- Tools

A Reasoning Problem

A is a logical consequence of K

written K $\models$ A

if and only if

every model of K is a model of A.

- **To show an entailment, we need to check all models?**
- **But that‘s infinitely many!!!**

A Reasoning Problem

We need algorithms which do not apply the model-based definition of logical consequence in a naive manner.

These algorithms should be syntax-based.
(Computers can only do syntax manipulations.)

Luckily, such algorithms exist!

However, their correctness (soundness and completeness) needs to be proven formally.
Which is often a non-trivial problem requiring substantial mathematical build-up.

We won‘t do the proofs here.

Proof Theory

We will show the Tableaux Method – implemented, e.g., in Pellet and Racer.

Alternatives:

- Transformation to disjunctive datalog using basic superposition done for SHIQ
- Naive mapping to Datalog for OWL RL
- Mapping to SQL for OWL QL
- Special-purpose algorithms for OWL EL e.g. transformation to Datalog

Proof theory Via Tableaux

- **Adaptation of FOL tableaux algorithms.**
- **Problem: OWL is decidable, but FOL tableaux algorithms do not guarantee termination.**
- **Solution: *blocking*.**

Contents

- Important inference problems
- Tableaux algorithm for ALC
- Tableaux algorithm for SHIQ

Important Inference Problems

- Global consistency of a knowledge base. KB $\models$ false?
  - Is the knowledge base meaningful?
- Class consistency $C \equiv \bot$?
  - Is C necessarily empty?
- Class inclusion (Subsumption) $C \sqsubseteq D$?
  - Structuring knowledge bases
- Class equivalence $C \equiv D$?
  - Are two classes in fact the same class?
- Class disjointness $C \sqcap D = \bot$?
  - Do they have common members?
- Class membership $C(a)$?
  - Is a contained in C?
- Instance Retrieval „find all x with C(x)“
  - Find all (known!) individuals belonging to a given class.

Reduction to Unsatisfiability

- **Global consistency of a knowledge base.** — **KB unsatisfiable**
  - Failure to find a model.
- **Class consistency** — $C \equiv \bot$?
  - KB $\cup$ {C(a)} unsatisfiable
- **Class inclusion (Subsumption)** — $C \sqsubseteq D$?
  - KB $\cup$ {$C \sqcap \neg D(a)$} unsatisfiable (a new)
- **Class equivalence** — $C \equiv D$?
  - $C \sqsubseteq D$ und $D \sqsubseteq C$
- **Class disjointness** — $C \sqcap D = \bot$?
  - KB $\cup$ {$(C \sqcap D)(a)$} unsatisfiable (a new)
- **Class membership** — C(a)?
  - KB $\cup$ {$\neg$C(a)} unsatisfiable
- **Instance Retrieval** — „find all x with C(x)“
  - Check class membership for all individuals.

Reduction to Satisfiability

- We will present so-called tableaux algorithms.

- They attempt to construct a model of the knowledge base in a „general, abstract“ manner.
  - If the construction fails, then (provably) there is no model – i.e. the knowledge base is unsatisfiable.
  - If the construction works, then it is satisfiable.

$\rightarrow$ Hence the reduction of all inference problems to the checking of unsatisfiability of the knowledge base!

Contents

- Important inference problems
- Tableaux algorithm for ALC
- Tableaux algorithm for SHIQ

ALC tableaux: contents

- Transformation to negation normal form
- Naive tableaux algorithm
- Tableaux algorithm with blocking

Transform. to negation normal form

Given a knowledge base K.

- Replace $C \equiv D$ by $C \sqsubseteq D$ and $D \sqsubseteq C$.
- Replace $C \sqsubseteq D$ by $\neg C \sqcup D$.
- Apply the equations on the next slide exhaustively.

Resulting knowledge base: NNF(K)

Negation normal form of K.

Negation occurs only directly in front of atomic classes.

$$
\begin{aligned}
\mathrm{NNF}(C) &= C \qquad \text{if } C \text{ is a class name}\\
\mathrm{NNF}(\neg C) &= \neg C \qquad \text{if } C \text{ is a class name}\\
\mathrm{NNF}(\neg\neg C) &= \mathrm{NNF}(C)\\
\mathrm{NNF}(C \sqcup D) &= \mathrm{NNF}(C) \sqcup \mathrm{NNF}(D)\\
\mathrm{NNF}(C \sqcap D) &= \mathrm{NNF}(C) \sqcap \mathrm{NNF}(D)\\
\mathrm{NNF}(\neg(C \sqcup D)) &= \mathrm{NNF}(\neg C) \sqcap \mathrm{NNF}(\neg D)\\
\mathrm{NNF}(\neg(C \sqcap D)) &= \mathrm{NNF}(\neg C) \sqcup \mathrm{NNF}(\neg D)\\
\mathrm{NNF}(\forall R.C) &= \forall R.\mathrm{NNF}(C)\\
\mathrm{NNF}(\exists R.C) &= \exists R.\mathrm{NNF}(C)\\
\mathrm{NNF}(\neg\forall R.C) &= \exists R.\mathrm{NNF}(\neg C)\\
\mathrm{NNF}(\neg\exists R.C) &= \forall R.\mathrm{NNF}(\neg C)
\end{aligned}
$$

K and NNF(K) have the same models (are *logically equivalent*).

Example

$P \sqsubseteq (E \sqcap U) \sqcup \neg(\neg E \sqcup D).$

In negation normal form:

$\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D).$

ALC tableaux: contents

- Transformation to negation normal form
- Naive tableaux algorithm
- Tableaux algorithm with blocking

Naive tableaux algorithm

Reduction to (un)satisfiability.

Idea:

- **Given knowledge base K**
- **Attempt construction of a tree (called *Tableau*), which represents a model of K.
(It‘s actually rather a *Forest*.)**
- **If attempt fails, K is unsatisfiable.**

The Tableau

- **Nodes represent elements of the domain of the model**
  $\rightarrow$ Every node x is labeled with a set L(x) of class expressions.
  C $\in$ L(x) means: "x is in the extension of C"

- **Edges stand for role relationships:**
  $\rightarrow$ Every edge <x,y> is labeled with a set L(<x,y>) of role names.
  R $\in$ L(<x,y>) means: "(x,y) is in the extension of R"

Simple example

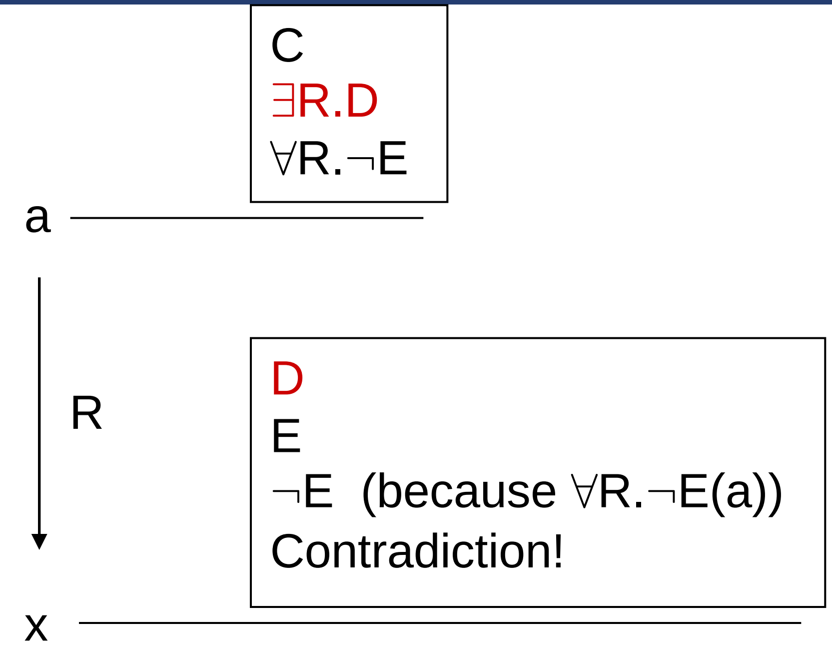
$C(a)$

$C \sqsubseteq \exists R.D$

$D \sqsubseteq E$

Does this entail $(\exists R.E)(a)$?

(add $\forall R.\neg E(a)$ and show unsatisfiability)



Another example

$C(a)$

$C \sqsubseteq \exists R.D$

$D \sqsubseteq E \sqcup F$

$F \sqsubseteq E$

Does this entail $(\exists R.E)(a)$?

(add $\forall R.\neg E(a)$ and show unsatisfiability)

C
$\exists R.D$
$\forall R.\neg E$

a

R

x

D
$\neg E$ (because $\forall R.\neg E(a)$)
choice: ($D \sqsubseteq E \sqcup F$):
1. E (contradiction!)
2. F
   E (contradiction!)

Formal Definition

- **Input: K=TBox + ABox (in NNF)**
- **Output: Whether or not K is satisfiable.**

- **A tableau is a directed labeled graph**
  - **nodes are individuals or (new) variable names**
  - **nodes x are labeled with sets L(x) of classes**
  - **edges <x,y> are labeled with sets L(<x,y>) of role names**

Initialisation

- Make a node for every individual in the ABox.
- Every node is labeled with the corresponding class names from the ABox.
- There is an edge, labeled with R, between a and b, if R(a,b) is in the ABox.

- (If there is no ABox, the initial tableau consists of a node x with empty label.)

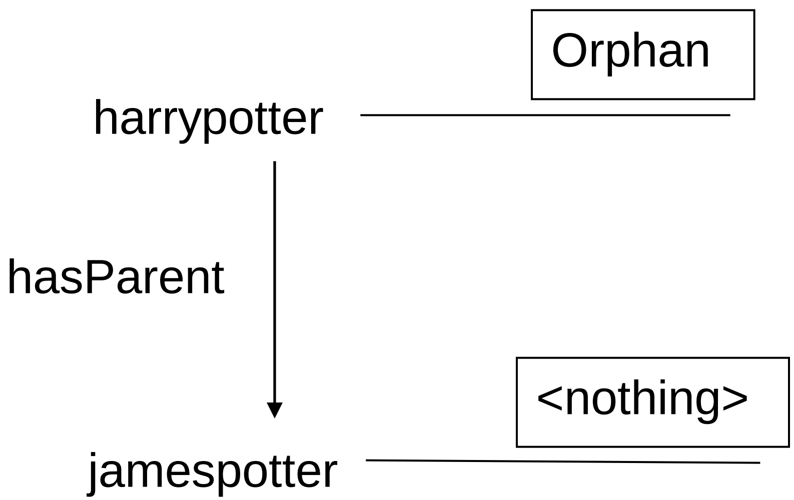
Example initialisation

Human $\sqsubseteq \exists$hasParent.Human

Orphan $\sqsubseteq$ Human $\sqcap \neg\exists$hasParent.Alive

Orphan(harrypotter)

hasParent(harrypotter,jamespotter)



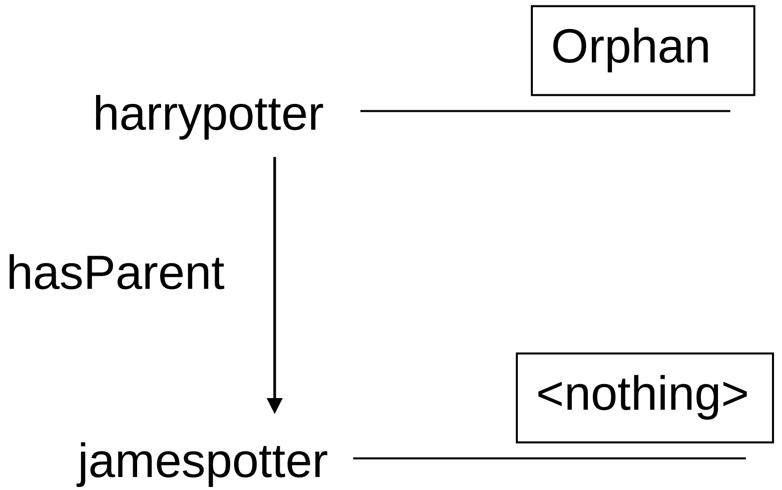
Careful: need NNF!

$\neg\text{Human} \sqcup \exists\text{hasParent}.\text{Human}$

$\neg\text{Orphan} \sqcup (\text{Human} \sqcap \forall\text{hasParent}.\neg\text{Alive})$

Orphan(harrypotter)

hasParent(harrypotter,jamespotter)



Constructing the tableau

- **Non-deterministically extend the tableau using the rules on the next slide.**

- **Terminate, if**
  - **there is a contradiction in a node label (i.e., it contains classes C and ¬C, or it contains ⊥), or**
  - **none of the rules is applicable.**

- **If the tableau does not contain a contradiction, then the knowledge base is satisfiable.**
  Or more precisely: If you can make a choice of rule applications such that no contradiction occurs and the process terminates, then the knowledge base is satisfiable.

Naive ALC tableaux rules

$\sqcap$-rule: If $C \sqcap D \in \mathcal{L}(x)$ and $\{C, D\} \not\subseteq \mathcal{L}(x)$, then set $\mathcal{L}(x) \leftarrow \{C, D\}$.

$\sqcup$-rule: If $C \sqcup D \in \mathcal{L}(x)$ and $\{C, D\} \cap \mathcal{L}(x) = \emptyset$, then set $\mathcal{L}(x) \leftarrow C$ or $\mathcal{L}(x) \leftarrow D$.

$\exists$-rule: If $\exists R.C \in \mathcal{L}(x)$ and there is no y with $R \in L(x, y)$ and $C \in \mathcal{L}(y)$, then

1. add a new node with label y (where y is a new node label),
2. set $\mathcal{L}(x, y) = \{R\}$, and
3. set $\mathcal{L}(y) = \{C\}$.

$\forall$-rule: If $\forall R.C \in \mathcal{L}(x)$ and there is a node y with $R \in \mathcal{L}(x, y)$ and $C \notin \mathcal{L}(y)$, then set $\mathcal{L}(y) \leftarrow C$.

TBox-rule: If C is a TBox statement and $C \notin \mathcal{L}(x)$, then set $\mathcal{L}(x) \leftarrow C$.

Example

¬Alive(jamespotter)
i.e. add: Alive(jamespotter)
and search for contradiction

$\neg$Human $\sqcup$ $\exists$hasParent.Human

$\neg$Orphan $\sqcup$ (Human $\sqcap$ $\forall$hasParent.$\neg$Alive)

Orphan(harrypotter)

hasParent(harrypotter,jamespotter)

harrypotter

Orphan
$\neg$Orphan $\sqcup$ (Human $\sqcap$ $\forall$hasParent.$\neg$Alive)
1. $\neg$Orphan (contradiction)
2. Human $\sqcap$ $\forall$hasParent.$\neg$Alive
   Human
   $\forall$hasParent.$\neg$Alive

hasParent

Alive

2. $\neg$Alive (contradiction)

jamespotter

ALC tableaux: contents

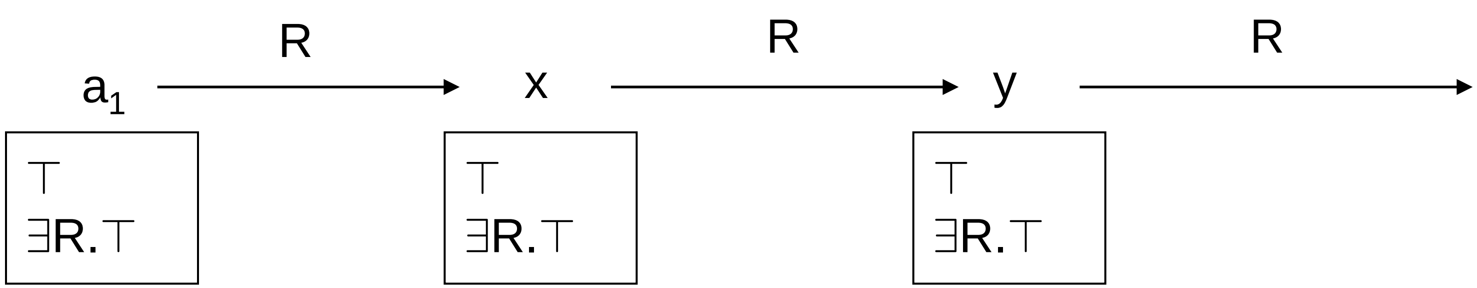
- **Transformation to negation normal form**
- **Naive tableaux algorithm**
- **Tableaux algorithm with blocking**

There‘s a termination problem

TBox: $\exists R.\top$

ABox: $\top(a_1)$

- **Obviously satisfiable:**
  Model M with domain elements $a_1^M, a_2^M, \ldots$
  and $R^M(a_i^M, a_{i+1}^M)$ for all $i \geq 1$
- **but tableaux algorithm does not terminate!**

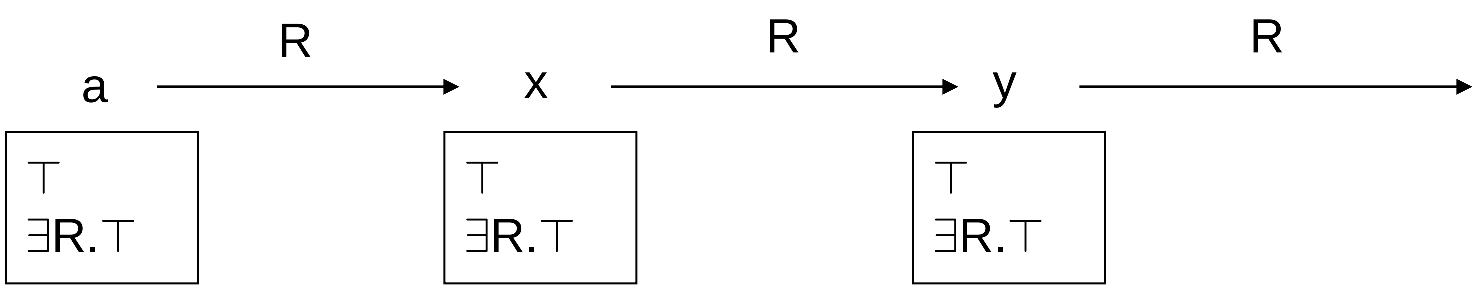


Solution?

Actually, things repeat!

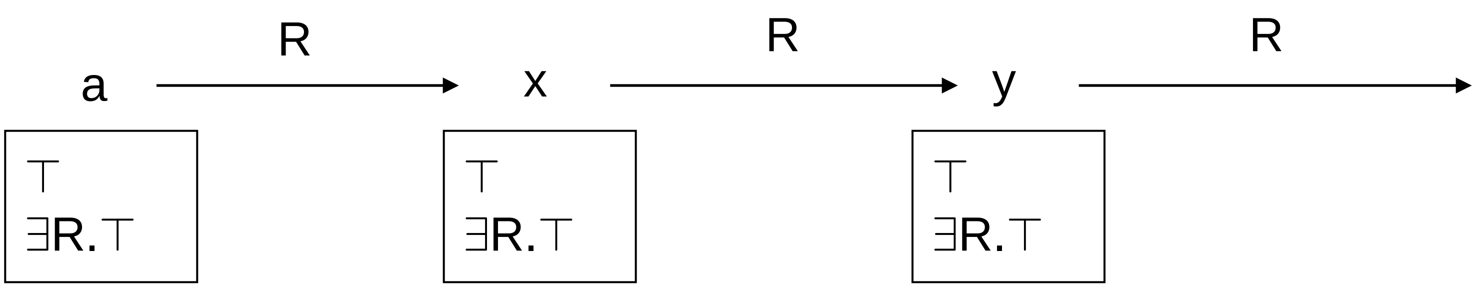
Idea: it is not necessary to expand x, since it‘s simply a copy of a.

$\Rightarrow$ **Blocking**



Blocking

- x is *blocked* (by y) if
  - x is not an individual (but a variable)
  - y is a predecessor of x and L(x) $\subseteq$ L(y)
  - or a predecessor of x is blocked



Here, x is blocked by a.

Constructing the tableau

- **Non-deterministically extend the tableau using the rules on the next slide, but only apply a rule if x is not blocked!**

- **Terminate, if**
  - **there is a contradiction in a node label (i.e., it contains classes C and ¬C), or**
  - **none of the rules is applicable.**

- **If the tableau does not contain a contradiction, then the knowledge base is satisfiable.**
  Or more precisely: If you can make a choice of rule applications such that no contradiction occurs and the process terminates, then the knowledge base is satisfiable.

Naive ALC tableaux rules

$\sqcap$-rule: If $C \sqcap D \in \mathcal{L}(x)$ and $\{C, D\} \not\subseteq \mathcal{L}(x)$, then set $\mathcal{L}(x) \leftarrow \{C, D\}$.

$\sqcup$-rule: If $C \sqcup D \in \mathcal{L}(x)$ and $\{C, D\} \cap \mathcal{L}(x) = \emptyset$, then set $\mathcal{L}(x) \leftarrow C$ or $\mathcal{L}(x) \leftarrow D$.

$\exists$-rule: If $\exists R.C \in \mathcal{L}(x)$ and there is no y with $R \in L(x, y)$ and $C \in \mathcal{L}(y)$, then

1. add a new node with label y (where y is a new node label),
2. set $\mathcal{L}(x, y) = \{R\}$, and
3. set $\mathcal{L}(y) = \{C\}$.

$\forall$-rule: If $\forall R.C \in \mathcal{L}(x)$ and there is a node y with $R \in \mathcal{L}(x, y)$ and $C \notin \mathcal{L}(y)$, then set $\mathcal{L}(y) \leftarrow C$.

TBox-rule: If C is a TBox statement and $C \notin \mathcal{L}(x)$, then set $\mathcal{L}(x) \leftarrow C$.

Apply only if x is not blocked!

Example (0)

- **Knowledge base {Human $\sqsubseteq$ $\exists$hasParent.Human, Bird(tweety)}**
- **We want to show that Human(tweety) does *not* hold,
  i.e. that ¬Human(tweety) is entailed.**
- **We will not be able to show this.
  I.e. Human(tweety) is *possible*.**

- **Shorter notation:**
  $H \sqsubseteq \exists p.H$
  B(t)

  ¬H(t) entailed?

Example (0)

Knowledge base {¬H ⊔ ∃p.H, B(t), H(t)}

$t \xrightarrow{p} x$

Box under t:

H
B
¬H ⊔ ∃p.H
1. ¬H (contradiction)
2. ∃p.H

Box under x:

2.:
H
blocked by t!

expansion stops. Cannot find contradiction!

Example (0) the other case

Knowledge base {$\neg H \sqcup \exists p.H$, $B(t)$, $\neg H(t)$}

$$t \xrightarrow{p} x \xrightarrow{p} y$$

$\neg H$
B
$\neg H \sqcup \exists p.H$
1. $\neg H$ cannot be added. no expansion in this part
2. $\exists p.H$

2.:
H
$\neg H \sqcup \exists p.H$
2.1: $\neg H$ (contradiction)
2.2: $\exists p.H$

2.2:
H
blocked by x

no further expansion possible – knowledge base is satisfiable!

Example(1)

Show, that

Professor $\sqsubseteq$ (Person $\sqcap$ Unversitymember)
$\sqcup$ (Person $\sqcap$ $\neg$PhDstudent)

entails that every Professor is a Person.

Find contradiction in:
$\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg S)$
$P \sqcap \neg E(x)$

x

$P \sqcap \neg E$
P
$\neg E$
$\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg S)$

1. $\neg P$ (contradiction)
2. $(E \sqcap U) \sqcup (E \sqcap \neg S)$
    1. $E \sqcap U$
       E (contradiction)
    2. $E \sqcap \neg S$
       E (contradiction)

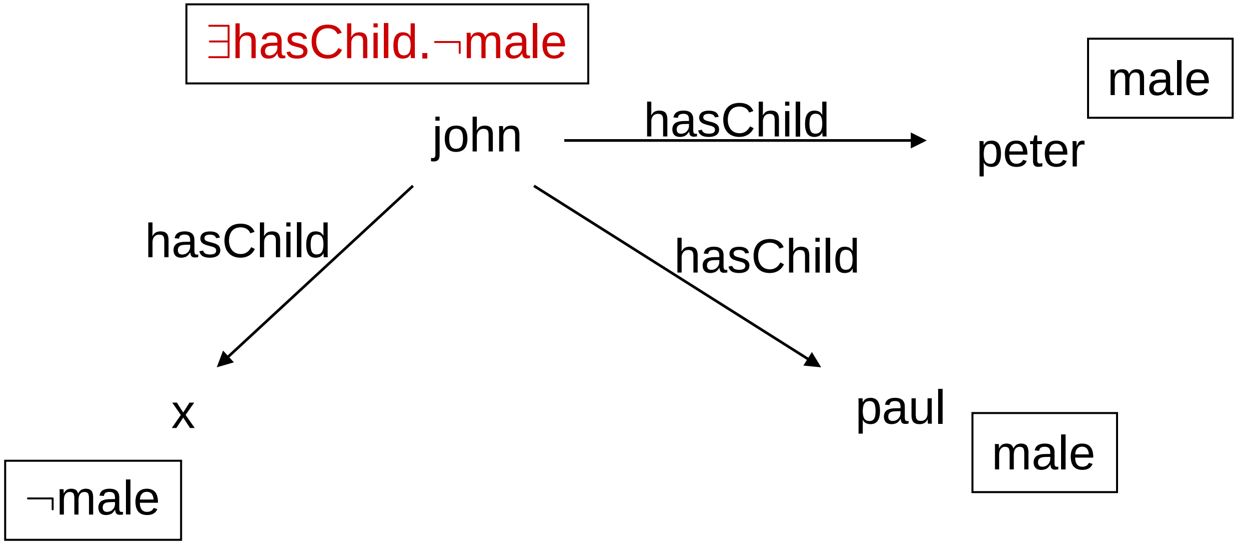
Example (2)

Show that

> **hasChild(john, peter)**
> **hasChild(john, paul)**
> **male(peter)**
> **male(paul)**

does *not* entail ∀hasChild.male(john).

$\neg\forall hasChild.male \equiv \exists hasChild.\neg male$



Example (3)

Show that the knowledge base

Bird $\sqsubseteq$ Flies

Penguin $\sqsubseteq$ Bird

Penguin $\sqcap$ Flies $\sqsubseteq \bot$

Penguin(tweety)

is unsatisfiable.

TBox:
$\neg B \sqcup F$
$\neg P \sqcup B$
$\neg P \sqcup \neg F \sqcup \bot$

tweety

P
$\neg P \sqcup B$
$\neg B \sqcup F$
$\neg P \sqcup \neg F$

1. $\neg P$ (contradiction)
2. B
   1. $\neg B$ (contradiction)
   2. F
      1. $\neg P$ (contradiction)
      2. $\neg F$ (contradiction)

Example (4)

Show that the knowledge base

$C(a)$

$C(c)$

$R(a,b)$

$R(a,c)$

$S(a,a)$

$S(c,b)$

$C \sqsubseteq \forall S.A$

$A \sqsubseteq \exists R.\exists S.A$

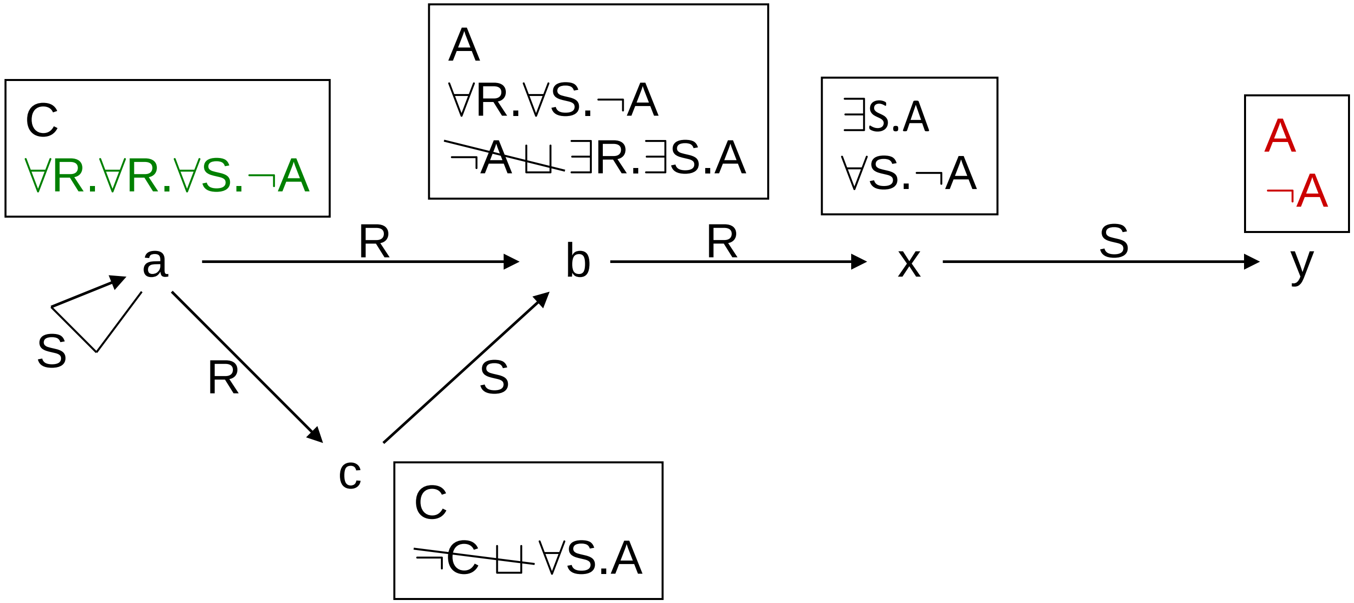
$A \sqsubseteq \exists R.C$

entails $\exists R.\exists R.\exists S.A(a)$.

Example (4)

$\neg\exists R.\exists R.\exists S.A \equiv \forall R.\forall R.\forall S.\neg A$

TBox:
$\neg C \sqcup \forall S.A$
$\neg A \sqcup \exists R.\exists S.A$
$\neg A \sqcup \exists R.C$



Contents

- Important inference problems
- Tableaux algorithm for ALC
- Tableaux algorithm for SHIQ

Tableaux Algorithm for SHIQ

- Basic idea is the same.
- Blocking rule is more complicated
- Other modifictions are also needed.

Transform. to negation normal form

Given a knowledge base K.

- **Replace $C \equiv D$ by $C \sqsubseteq D$ and $D \sqsubseteq C$.**
- **Replace $C \sqsubseteq D$ by $\neg C \sqcup D$.**
- **Apply the equations on the next slide exhaustively.**

Resulting knowledge base: NNF(K)

***Negation normal form* of K.**

Negation occurs only directly in front of atomic classes.

$$
\begin{aligned}
\mathrm{NNF}(C) &= C \qquad \text{if } C \text{ is a class name}\\
\mathrm{NNF}(\neg C) &= \neg C \qquad \text{if } C \text{ is a class name}\\
\mathrm{NNF}(\neg\neg C) &= \mathrm{NNF}(C)\\
\mathrm{NNF}(C \sqcup D) &= \mathrm{NNF}(C) \sqcup \mathrm{NNF}(D)\\
\mathrm{NNF}(C \sqcap D) &= \mathrm{NNF}(C) \sqcap \mathrm{NNF}(D)\\
\mathrm{NNF}(\neg(C \sqcup D)) &= \mathrm{NNF}(\neg C) \sqcap \mathrm{NNF}(\neg D)\\
\mathrm{NNF}(\neg(C \sqcap D)) &= \mathrm{NNF}(\neg C) \sqcup \mathrm{NNF}(\neg D)\\
\mathrm{NNF}(\forall R.C) &= \forall R.\mathrm{NNF}(C)\\
\mathrm{NNF}(\exists R.C) &= \exists R.\mathrm{NNF}(C)\\
\mathrm{NNF}(\neg\forall R.C) &= \exists R.\mathrm{NNF}(\neg C)\\
\mathrm{NNF}(\neg\exists R.C) &= \forall R.\mathrm{NNF}(\neg C)
\end{aligned}
$$

NNF($\leq$n R.C) = $\leq$n R.NNF(C)

NNF($\geq$n R.C) = $\geq$n R.NNF(C)

NNF($\neg$ $\leq$n R.C) = $\geq$(n+1)R.NNF(C)

NNF($\neg$ $\geq$n R.C) = $\leq$(n-1)R.NNF(C), where $\leq$(-1)R.C = $\perp$

K and NNF(K) have the same models (are *logically equivalent*).

Formal Definition

- A tableau is a directed labeled graph
  - nodes are individuals or (new) variable names
  - nodes x are labeled with sets L(x) of classes
  - edges <x,y> are labeled
    - either with sets L(<x,y>) of role names or inverse role names
    - or with the symbol = (for equality)
    - or with the symbol ≠ (for inequality)

Initialisation

- Make a node for every individual in the ABox. These nodes are called *root nodes*.
- Every node is labeled with the corresponding class names from the ABox.
- There is an edge, labeled with R, between a and b, if R(a,b) is in the ABox.
- There is an edge, labeled ≠, between a and b if a ≠ b is in the ABox.
- There are no = relations (yet).

Notions

- We write S^{--} as S.
- If R $\in$ L(<x,y>) and R $\sqsubseteq$ S (where R,S can be inverse roles), then
  - y is an S-successor of x and
  - x is an S-predecessor of y.
- If y is an S-successor or an S^{-}-predecessor of x, then y is an *neighbor* of x.
- *Ancestor* is the transitive closure of *Predecessor.*

Blocking for SHIQ

- x is *blocked* by y if x,y are not root nodes and
  - the following hold: ["x is directly blocked"]
    - no ancestor of x is blocked
    - there are predecessors y', x' of x
    - y is a successor of y' and x is a successor of x'
    - L(x) = L(y) and L(x') = L(y')
    - L(<x',x>) = L(<y',y>)
  - or the following holds: ["x is indirectly blocked"]
    - an ancestor of x is blocked or
    - x is successor of some y with L(<y,x>) = $\emptyset$

Constructing the tableau

- **Non-deterministically extend the tableau using the rules on the next slide.**

- **Terminate, if**
  - **there is a contradiction in a node label, i.e.,**
    - **it contains $\bot$ or classes C and $\neg$C or**
    - **it contains a class $\leq$ nR.C and x also has (n+1) R-successors y_i and $y_i \neq y_j$ (for all $i \neq j$)**
  - **or none of the rules is applicable.**

- **If the tableau does not contain a contradiction, then the knowledge base is satisfiable.**
  Or more precisely: If you can make a choice of rule applications such that no contradiction occurs and the process terminates, then the knowledge base is satisfiable.

SHIQ Tableaux Rules

$\sqcap$-rule: If x is not indirectly blocked, $C \sqcap D \in \mathcal{L}(x)$, and $\{C, D\} \not\subseteq \mathcal{L}(x)$, then set $\mathcal{L}(x) \leftarrow \{C, D\}$.

$\sqcup$-rule: If x is not indirectly blocked, $C \sqcup D \in \mathcal{L}(x)$ and $\{C, D\} \sqcap \mathcal{L}(x) = \emptyset$, then set $\mathcal{L}(x) \leftarrow C$ or $\mathcal{L}(x) \leftarrow D$.

$\exists$-rule: If x is not blocked, $\exists R.C \in \mathcal{L}(x)$, and there is no y with $R \in \mathcal{L}(x, y)$ and $C \in \mathcal{L}(y)$, then

1. add a new node with label y (where y is a new node label),
2. set $\mathcal{L}(x, y) = \{R\}$ and $\mathcal{L}(y) = \{C\}$.

$\forall$-rule: If x is not indirectly blocked, $\forall R.C \in \mathcal{L}(x)$, and there is a node y with $R \in \mathcal{L}(x, y)$ and $C \notin \mathcal{L}(y)$, then set $\mathcal{L}(y) \leftarrow C$.

TBox-rule: If x is not indirectly blocked, C is a TBox statement, and $C \notin \mathcal{L}(x)$, then set $\mathcal{L}(x) \leftarrow C$.

SHIQ Tableaux Rules

trans-rule: If x is not indirectly blocked, $\forall S.C \in \mathcal{L}(x)$, S has a transitive subrole R, and x has an R-neighbor y with $\forall R.C \notin \mathcal{L}(y)$, then set $\mathcal{L}(y) \leftarrow \forall R.C$.

choose-rule: If x is not indirectly blocked, $\leq nS.C \in \mathcal{L}(x)$ or $\geq nS.C \in \mathcal{L}(x)$, and there is an S-neighbor y of x with $\{C, \mathrm{NNF}(\neg C)\} \cap \mathcal{L}(y) = \emptyset$, then set $\mathcal{L}(y) \leftarrow C$ or $\mathcal{L}(y) \leftarrow \mathrm{NNF}(\neg C)$.

$\geq$-rule: If x is not blocked, $\geq nS.C \in \mathcal{L}(x)$, and there are no n S-neighbors $y_1, \ldots, y_n$ of x with $C \in \mathcal{L}(y_i)$ and $y_i \not\approx y_j$ for $i, j \in \{1, \ldots, n\}$ and $i \neq j$, then

1. create n new nodes with labels $y_1, \ldots, y_n$ (where the labels are new),
2. set $\mathcal{L}(x, y_i) = \{S\}$, $\mathcal{L}(y_i) = \{C\}$, and $y_i \not\approx y_j$ for all $i, j \in \{1, \ldots, n\}$ with $i \neq j$.

$\leq$-rule: If x is not indirectly blocked, $\leq nS.C \in \mathcal{L}(x)$, there are more than n S-neighbors y_i of x with $C \in \mathcal{L}(y_i)$, and x has two S-neighbors y, z such that y is neither a root node nor an ancestor of z, $y \not\approx z$ does not hold, and $C \in \mathcal{L}(y) \cap \mathcal{L}(z)$, then

1. set $\mathcal{L}(z) \leftarrow \mathcal{L}(y)$,
2. if z is an ancestor of x, then $\mathcal{L}(z, x) \leftarrow \{\mathrm{Inv}(R) \mid R \in \mathcal{L}(x, y)\}$,
3. if z is not an ancestor of x, then $\mathcal{L}(x, z) \leftarrow \mathcal{L}(x, y)$,
4. set $\mathcal{L}(x, y) = \emptyset$, and
5. set $u \not\approx z$ for all u with $u \not\approx y$.

$\leq$-root-rule: If $\leq nS.C \in \mathcal{L}(x)$, there are more than n S-neighbors y_i of x with $C \in \mathcal{L}(y_i)$, and x has two S-neighbors y, z which are both root nodes, $y \not\approx z$ does not hold, and $C \in \mathcal{L}(y) \cap \mathcal{L}(z)$, then

1. set $\mathcal{L}(z) \leftarrow \mathcal{L}(y)$,
2. for all directed edges from y to some w, set $\mathcal{L}(z, w) \leftarrow \mathcal{L}(y, w)$,
3. for all directed edges from some w to y, set $\mathcal{L}(w, z) \leftarrow \mathcal{L}(w, y)$,
4. set $\mathcal{L}(y) = \mathcal{L}(w, y) = \mathcal{L}(y, w) = \emptyset$ for all w,
5. set $u \not\approx z$ for all u with $u \not\approx y$, and
6. set $y \approx z$.

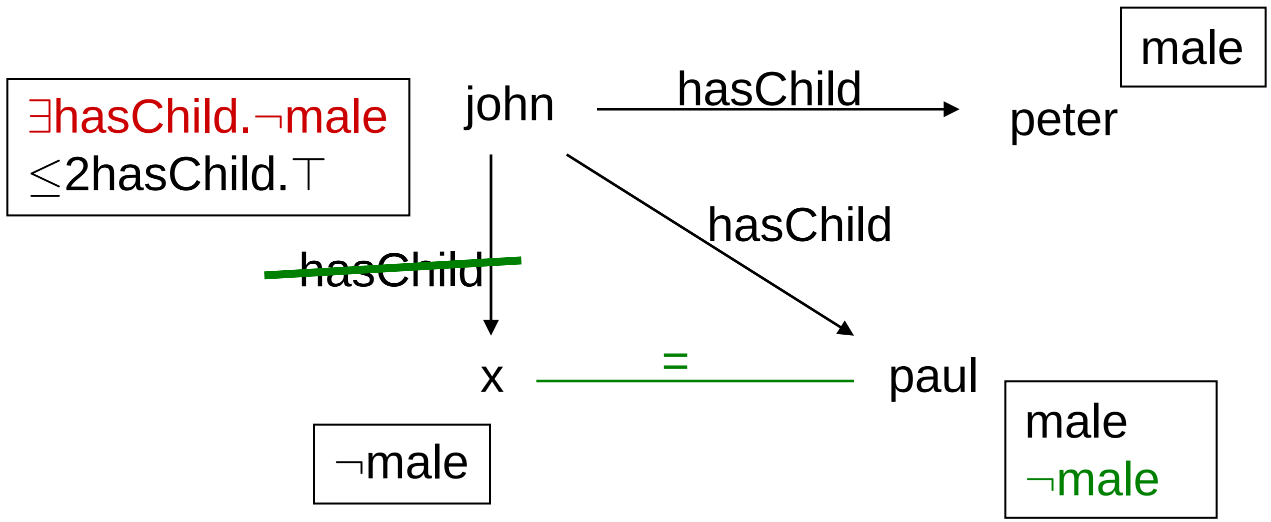
Example (1): cardinalities

Show, that

hasChild(john, peter)
hasChild(john, paul)
male(peter)
male(paul)
$\leq$2hasChild.$\top$(john)

does *not* entail $\forall$hasChild.male(john).

$\neg\forall$hasChild.male $\equiv$ $\exists$hasChild.$\neg$male



now apply $\leq$

Example (1): cardinalities

Show, that

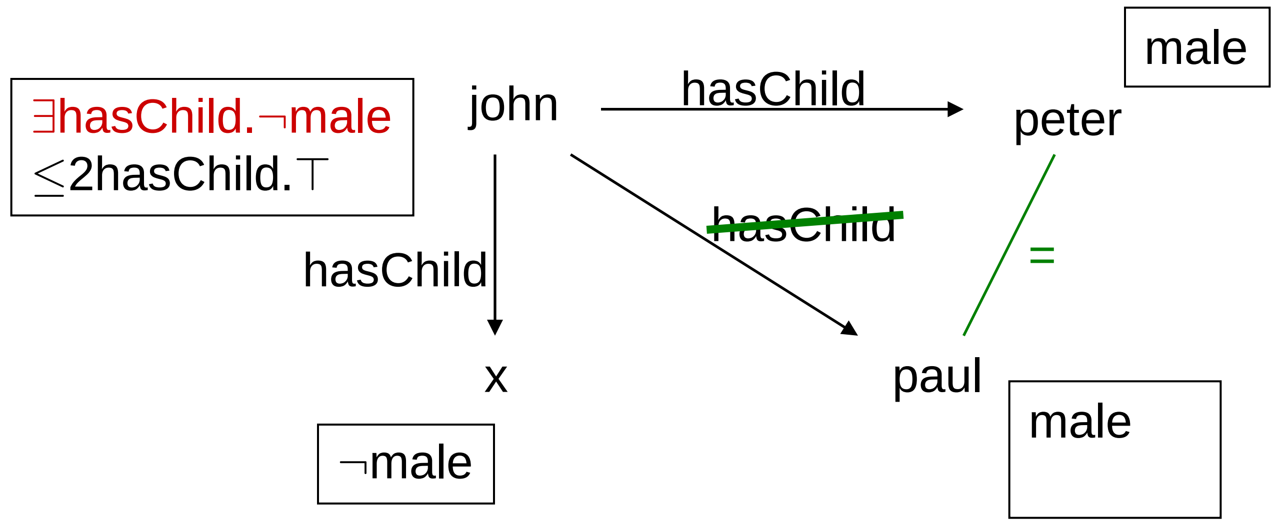
hasChild(john, peter)
hasChild(john, paul)
male(peter)
male(paul)
$\leq$2hasChild.$\top$(john)

does *not* entail $\forall$hasChild.male(john).

$\neg\forall$hasChild.male $\equiv$ $\exists$hasChild.$\neg$male

backtracking!

now apply $\leq$



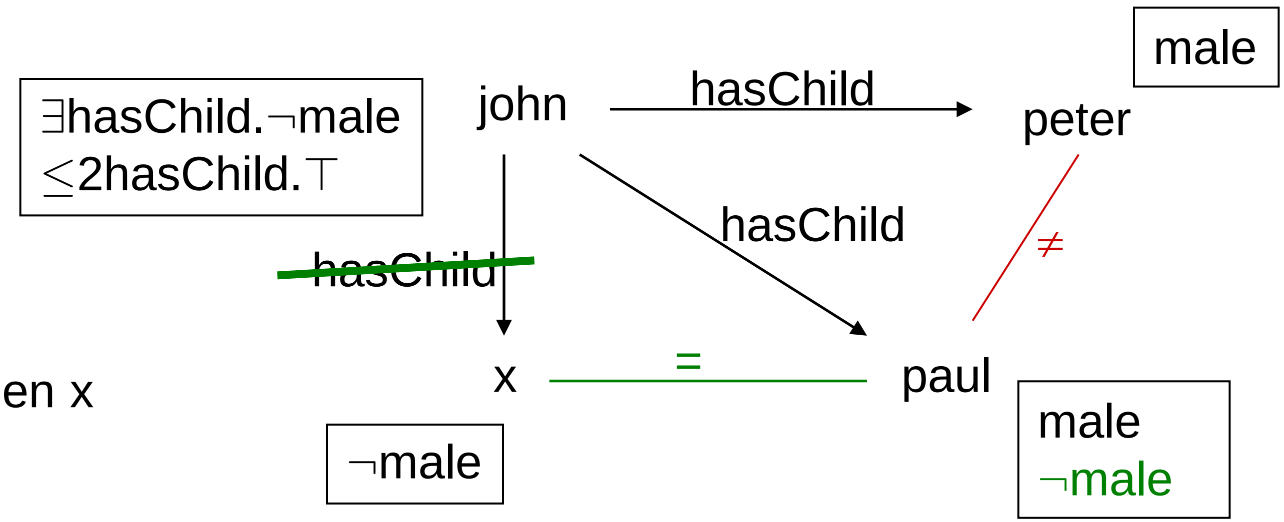
Example (1): cardinalities – again

Show, that

hasChild(john, peter)
hasChild(john, paul)
male(peter)
male(paul)
$\leq$2hasChild.$\top$(john) and peter $\neq$ paul

does *not* entail $\forall$hasChild.male(john).

$\neg\forall$hasChild.male $\equiv$ $\exists$hasChild.$\neg$male



now apply $\leq$

can backtrack only between x and peter – also leads to contradiction

Example (2): cardinalities

$\neg{\geq}2\text{hasSon}.\top \equiv {\leq}1\text{hasChild}.\top$

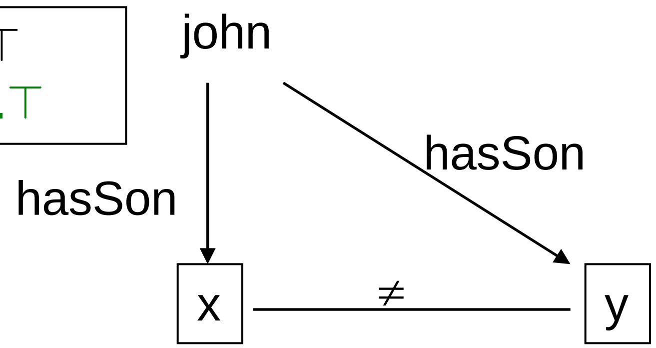
hasSon $\sqsubseteq$ hasChild

Show, that

$\geq 2\text{hasSon}.\top(\text{john})$

entails $\geq 2\text{hasChild}.\top(\text{john})$.

$\geq 2\text{hasSon}.\top$
$\leq 1\text{hasChild}.\top$



hasSon-neighbors are also hasChild-neighbors,
tableau terminates with contradiction

Example (3): choose

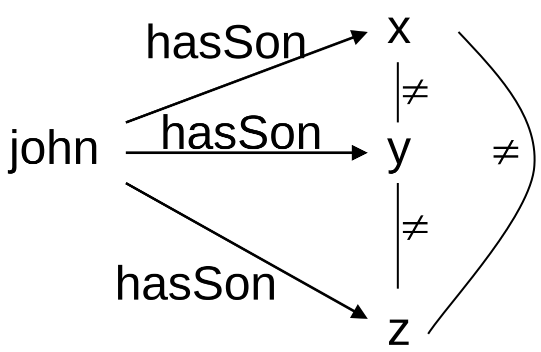
$\geq$3hasSon(john)

$\leq$2hasSon.male(john)

Is this contradictory?

No, because the following tableau is complete.

$\geq$3hasSon
$\leq$2hasSon.male



Example (4): inverse roles

$\exists$hasChild.human(john)

human $\sqsubseteq$ $\forall$hasParent.human

hasChild $\sqsubseteq$ hasParent^{-}

zu zeigen: human(john)

$\exists$hasChild.human
$\neg$human
human

john $\xrightarrow{\text{hasChild}}$ x

human
$\cancel{\neg\text{human} \sqcup}$ $\forall$hasParent.human

john is hP^{-}-predecessor of x, hence hP-neighbor of x

Example (5): Transitivity and Blocking

human $\sqsubseteq$ $\exists$hasFather.$\top$

human $\sqsubseteq$ $\forall$hasAncestor.human

hasFather $\sqsubseteq$ hasAncestor **Trans(hasAncestor)**

human(john)

Does this entail $\leq$1hasFather.$\top$(john)?

Negation: $\geq$2hasFather.$\top$(john)

Example (5): Transitivity and Blocking

human $\sqsubseteq$ $\exists$hasFather.$\top$

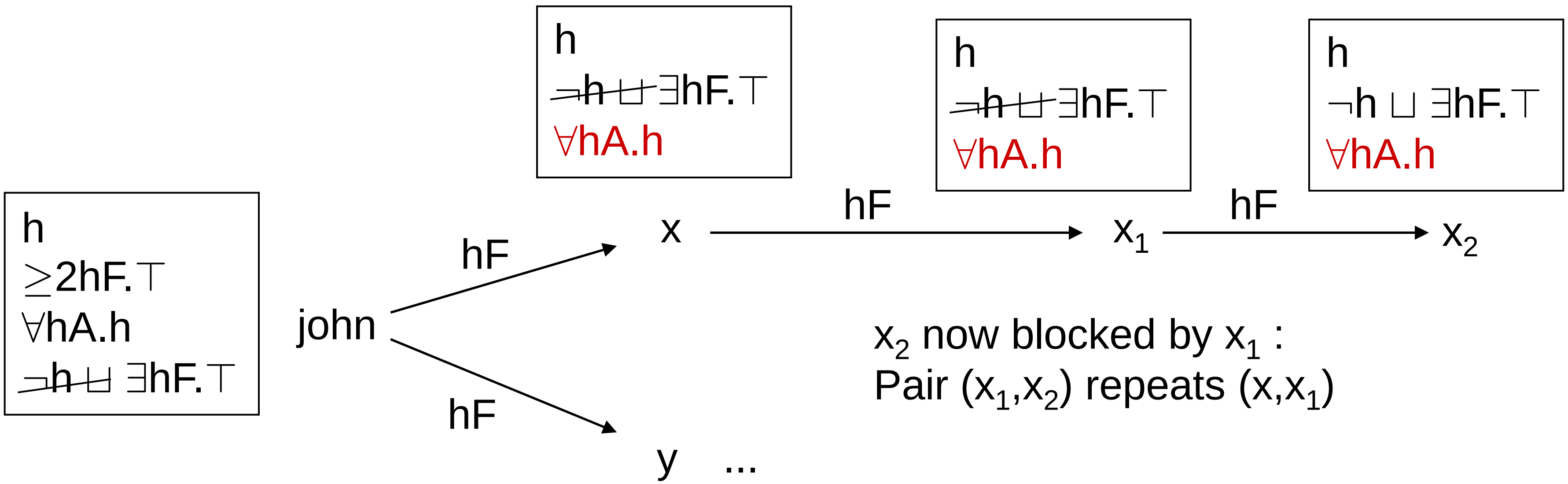
hasFather $\sqsubseteq$ hasAncestor

Trans(hasAncestor)

$\forall$hasAncestor.human(john)

human(john)

$\geq$2hasFather.$\top$(john)



same as branch above

Example (6): Pairwise Blocking

$\neg C \sqcap (\leq 1F) \sqcap \exists F^{-}.D \sqcap \forall R^{-}.(\exists F^{-}.D)$, where
$D = C \sqcap (\leq 1F) \sqcap \exists F.\neg C$, Trans(R), and $F \sqsubseteq R$,
is not satisfiable.

$$x \xrightarrow{F^{-}} y \xrightarrow{F^{-}} z$$

x	y	z
$\neg C$ $\leq 1F$ $\exists F^{-}.D$ $\forall R^{-}.(\exists F^{-}.D)$	D $\exists F^{-}.D$ $\forall R^{-}.(\exists F^{-}.D)$ C $\leq 1F$ $\exists F.\neg C$	D $\exists F^{-}.D$ $\forall R^{-}.(\exists F^{-}.D)$ C $\leq 1F$ $\exists F.\neg C$

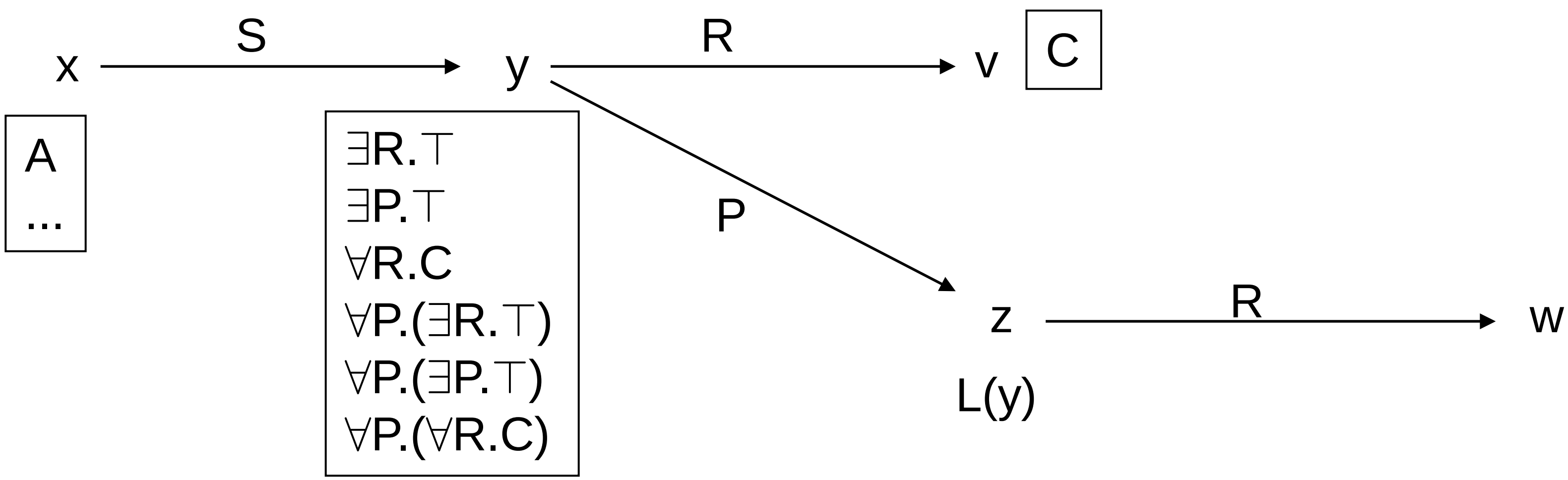
Without pairwise blocking, z would be blocked, which shouldn‘t happen:
Expansion of $\exists F.\neg C$ yields $\neg C$ for node y as required.

Example (7): Dynamic Blocking

$A \sqcap \exists S.(\exists R.\top \sqcap \exists P.\top \sqcap \forall R.C \sqcap \forall P.(\exists R.\top) \sqcap \forall P.(\forall R.C) \sqcap \forall P.(\exists P.\top))$

with $C = \forall R^{-}.(\forall P^{-}.(\forall S^{-}.\neg A))$ and Trans(P), is not satisfiable.

Part of the tableau:



At this stage, z would be blocked by y (assuming the presence of another pair). However, when C from v is expanded, z becomes unblocked, which is necessary in order to label w with C which in turn labels x with $\neg$A, yielding the required contradiction.

Tableaux Reasoners

- **Fact++**
  - **http://owl.man.ac.uk/factplusplus/**
- **Pellet**
  - **http://www.mindswap.org/2003/pellet/index.shtml**
- **RacerPro**
  - **http://www.sts.tu-harburg.de/~r.f.moeller/racer/**

Contents

OWL tools (incomplete listing)

Reasoner:

- OWL 2 DL:
  - Pellet http://clarkparsia.com/pellet/
  - HermiT http://www.hermit-reasoner.com/
- OWL 2 EL:
  - CEL http://code.google.com/p/cel/
- OWL 2 RL:
  - essentially any rule engine
- OWL 2 QL:
  - essentially any SQL engine (with a bit of query rewriting on top)

Editors:

- Protégé
- NeOn Toolkit
- TopBraid Composer

Main References

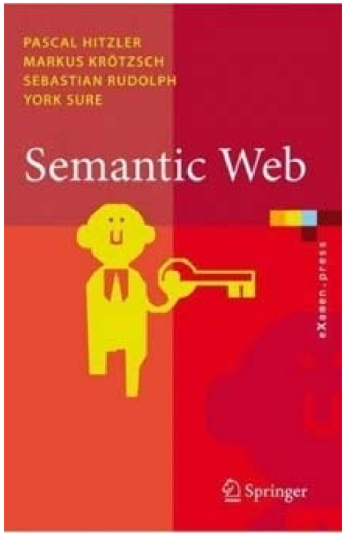
- **W3C OWL Working Group, OWL 2 Web Ontology Language: Document Overview. http://www.w3.org/TR/owl2-overview/**

- **Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter Patel-Schneider, Sebastian Rudolph, OWL 2 Web Ontology Language: Primer. http://www.w3.org/TR/owl2-primer/**

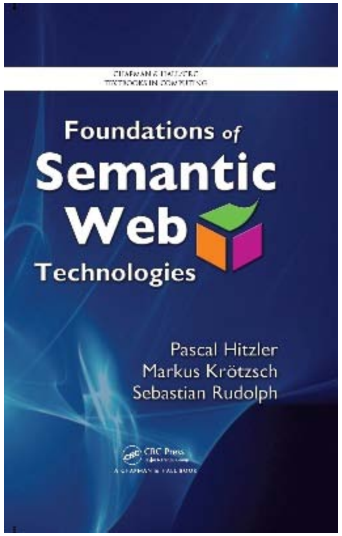
- **Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, Peter F. Patel-Schneider, The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2nd edition, 2007.**

Main References – Textbooks

- **Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, York Sure, Semantic Web – Grundlagen. Springer, 2008. http://www.semantic-web-grundlagen.de/ (In German.) (Does not cover OWL 2.)**



- **Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, Foundations of Semantic Web Technologies. Chapman & Hall/CRC, 2009. http://www.semantic-web-book.org/wiki/FOST (Ask for a flyer from us.)**



Further References

- DL complexity calculator: http://www.cs.man.ac.uk/~ezolin/dl/

- Markus Krötzsch, Sebastian Rudolph, Pascal Hitzler, Description Logic Rules. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, Nikos Avouris, eds.: Proceedings of the 18th European Conference on Artificial Intelligence (ECAI-08), pp. 80–84. IOS Press 2008.

- Markus Krötzsch, Sebastian Rudolph, Pascal Hitzler, ELP: Tractable Rules for OWL 2. In: Amit Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy Finin, Krishnaprasad Thirunarayan (eds.), The Semantic Web - ISWC 2008, 7th International Semantic Web Conference. Springer Lecture Notes in Computer Science Vol. 5318, 2008, pp. 649-664.

Thanks!

http://www.semantic-web-book.org/page/ISWC2010_Tutorial

OWL 2 and Rules

–

Optional Part, If Enough Time

Main References Optional Part

Kno.e.sis

Main References:

- Markus Krötzsch, Sebastian Rudolph, Pascal Hitzler, Description Logic Rules. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, Nikos Avouris, eds.: Proceedings of the 18th European Conference on Artificial Intelligence (**ECAI-08**), pp. 80–84. IOS Press 2008.
- Markus Krötzsch, Sebastian Rudolph, Pascal Hitzler, ELP: Tractable Rules for OWL 2. In Amit Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy Finin, Krishnaprasad Thirunarayan, eds.: Proceedings of the 7th International Semantic Web Conference (**ISWC-08**), pp. 649–664. Springer 2008.

Contents

- Motivation: OWL and Rules
- Preliminaries: Datalog

Intro

- More rules than you ever need: SWRL
- Retaining decidability I: DL-safety
- Retaining decidability II: DL Rules

Extending OWL with Rules

- The rules hidden in OWL 2: SROIQ Rules
- Retaining tractability I: OWL 2 EL Rules
- Retaining tractability II: DLP 2

Rules inside OWL

- Retaining tractability III: ELP

putting it all together

Contents

- **Motivation: OWL and Rules**
- Preliminaries: Datalog

Intro

- More rules than you ever need: SWRL
- Retaining decidability I: DL-safety
- Retaining decidability II: DL Rules

Extending OWL with Rules

- The rules hidden in OWL 2: SROIQ Rules
- Retaining tractability I: OWL 2 EL Rules
- Retaining tractability II: DLP 2

Rules inside OWL

- Retaining tractability III: ELP

putting it all together

Motivation: OWL and Rules

- Rules (mainly, logic programming) as alternative ontology modelling paradigm.
- Similar tradition, and in use in practice (e.g. F-Logic)

- Ongoing: W3C RIF working group
  - Rule Interchange Format
  - based on Horn-logic
  - language standard forthcoming 2009

- Seek: Integration of rules paradigm with ontology paradigm
  - Here: Tight Integration in the tradition of OWL
  - Foundational obstacle: reasoning efficiency / decidability [naive combinations are undecidable]

Contents

- Motivation: OWL and Rules
- **Preliminaries: Datalog**

Intro

- More rules than you ever need: SWRL
- Retaining decidability I: DL-safety
- Retaining decidability II: DL Rules

Extending OWL with Rules

- The rules hidden in OWL 2: SROIQ Rules
- Retaining tractability I: OWL 2 EL Rules
- Retaining tractability II: DLP 2

Rules inside OWL

- Retaining tractability III: ELP

putting it all together

Preliminaries: Datalog

- Essentially Horn-rules without function symbols

  general form of the rules:

  $p_1(x_1,\ldots,x_n) \wedge \ldots \wedge p_m(y_1,\ldots,y_k) \rightarrow q(z_1,\ldots,z_j)$

  body $\rightarrow$ head

  semantics either as in predicate logic
  or as Herbrand semantics (see next slide)

- decidable
- polynomial data complexity (in number of facts)
- combined (overall) complexity: ExpTime
- combined complexity is P if the number of variables per rule is globally bounded

Datalog semantics example

- **Example:**
  $p(x) \rightarrow q(x)$
  $q(x) \rightarrow r(x)$
  $\rightarrow p(a)$

- **predicate logic semantics:**

  $(\forall x)\ (p(x) \rightarrow r(x))$
  and
  $(\forall x)\ (\neg r(x) \rightarrow \neg p(x))$
  are logical consequences

  q(a) and r(a)
  are logical consequences

- **Herbrand semantics**

  those on the left are not logical consequences

  q(a) and r(a)
  are logical consequences

  material implication:
  apply only to known constants

Contents

- Motivation: OWL and Rules
- Preliminaries: Datalog

Intro

- **More rules than you ever need: SWRL**
- Retaining decidability I: DL-safety
- Retaining decidability II: DL Rules

Extending OWL with Rules

- The rules hidden in OWL 2: SROIQ Rules
- Retaining tractability I: OWL 2 EL Rules
- Retaining tractability II: DLP 2

Rules inside OWL

- Retaining tractability III: ELP

putting it all together

More rules than you ever need: SWRL

- **Union of OWL DL with (binary) function-free Horn rules (with binary Datalog rules)**

- **undecidable**
- **no native tools available**

- **rather an overarching formalism**

- **see http://www.w3.org/Submission/SWRL/**

SWRL example (running example)

NutAllergic(sebastian)

NutProduct(peanutOil)

$\exists$orderedDish.ThaiCurry(sebastian)

ThaiCurry $\sqsubseteq$ $\exists$contains.{peanutOil}

$\top$ $\sqsubseteq$ $\forall$orderedDish.Dish

NutAllergic(x) $\wedge$ NutProduct(y) $\rightarrow$ dislikes(x,y)

dislikes(x,z) $\wedge$ Dish(y) $\wedge$ contains(y,z) $\rightarrow$ dislikes(x,y)

orderedDish(x,y) $\wedge$ dislikes(x,y) $\rightarrow$ Unhappy(x)

SWRL example (running example)

NutAllergic(sebastian)

NutProduct(peanutOil)

$\exists$orderedDish.ThaiCurry(sebastian)

ThaiCurry $\sqsubseteq$ $\exists$contains.{peanutOil}

$\top \sqsubseteq \forall$orderedDish.Dish

NutAllergic(x) $\wedge$ NutProduct(y) $\rightarrow$ dislikes(x,y)

dislikes(x,z) $\wedge$ Dish(y) $\wedge$ contains(y,z) $\rightarrow$ dislikes(x,y)

orderedDish(x,y) $\wedge$ dislikes(x,y) $\rightarrow$ Unhappy(x)

Conclusions:

dislikes(sebastian,peanutOil)

SWRL example (running example)

NutAllergic(sebastian)

NutProduct(peanutOil)

∃orderedDish.ThaiCurry(sebastian)

ThaiCurry $\sqsubseteq$ ∃contains.{peanutOil}

$\top \sqsubseteq$ ∀orderedDish.Dish

orderedDish rdfs:range Dish.

NutAllergic(x) $\wedge$ NutProduct(y) $\rightarrow$ dislikes(x,y)

dislikes(x,z) $\wedge$ Dish(y) $\wedge$ contains(y,z) $\rightarrow$ dislikes(x,y)

orderedDish(x,y) $\wedge$ dislikes(x,y) $\rightarrow$ Unhappy(x)

Conclusions:

dislikes(sebastian,peanutOil)

orderedDish(sebastian,y_s)

ThaiCurry(y_s)

Dish(y_s)

SWRL example (running example)

NutAllergic(sebastian)

NutProduct(peanutOil)

$\exists$orderedDish.ThaiCurry(sebastian)

ThaiCurry $\sqsubseteq$ $\exists$contains.{peanutOil}

$\top \sqsubseteq \forall$orderedDish.Dish

NutAllergic(x) $\wedge$ NutProduct(y) $\rightarrow$ dislikes(x,y)

dislikes(x,z) $\wedge$ Dish(y) $\wedge$ contains(y,z) $\rightarrow$ dislikes(x,y)

orderedDish(x,y) $\wedge$ dislikes(x,y) $\rightarrow$ Unhappy(x)

Conclusions:

dislikes(sebastian,peanutOil)

orderedDish(sebastian,y_s)

ThaiCurry(y_s)

Dish(y_s)

contains(y_s,peanutOil)

SWRL example (running example)

KNO.E.SIS

NutAllergic(sebastian)

NutProduct(peanutOil)

∃orderedDish.ThaiCurry(sebastian)

ThaiCurry ⊑ ∃contains.{peanutOil}

⊤ ⊑ ∀orderedDish.Dish

NutAllergic(x) ∧ NutProduct(y) → dislikes(x,y)

dislikes(x,z) ∧ Dish(y) ∧ contains(y,z) → dislikes(x,y)

orderedDish(x,y) ∧ dislikes(x,y) → Unhappy(x)

Conclusions:

dislikes(sebastian,peanutOil)

orderedDish(sebastian,y_s)

ThaiCurry(y_s)

Dish(y_s)

contains(y_s,peanutOil)

dislikes(sebastian,y_s)

SWRL example (running example)

NutAllergic(sebastian)
NutProduct(peanutOil)
$\exists$orderedDish.ThaiCurry(sebastian)

ThaiCurry $\sqsubseteq$ $\exists$contains.{peanutOil}
$\top$ $\sqsubseteq$ $\forall$orderedDish.Dish

NutAllergic(x) $\wedge$ NutProduct(y) $\rightarrow$ dislikes(x,y)
dislikes(x,z) $\wedge$ Dish(y) $\wedge$ contains(y,z) $\rightarrow$ dislikes(x,y)
orderedDish(x,y) $\wedge$ dislikes(x,y) $\rightarrow$ Unhappy(x)

Conclusions:
dislikes(sebastian,peanutOil)
orderedDish(sebastian,y_s)
ThaiCurry(y_s)
Dish(y_s)
contains(y_s,peanutOil)
dislikes(sebastian,y_s)
Unhappy(sebastian)

SWRL example (running example)

NutAllergic(sebastian)
NutProduct(peanutOil)
$\exists$orderedDish.ThaiCurry(sebastian)

ThaiCurry $\sqsubseteq$ $\exists$contains.{peanutOil}
$\top \sqsubseteq \forall$orderedDish.Dish

NutAllergic(x) $\wedge$ NutProduct(y) $\rightarrow$ dislikes(x,y)
dislikes(x,z) $\wedge$ Dish(y) $\wedge$ contains(y,z) $\rightarrow$ dislikes(x,y)
orderedDish(x,y) $\wedge$ dislikes(x,y) $\rightarrow$ Unhappy(x)

Conclusion: Unhappy(sebastian)

Contents

- Motivation: OWL and Rules
- Preliminaries: Datalog

Intro

- More rules than you ever need: SWRL
- **Retaining decidability I: DL-safety**
- Retaining decidability II: DL Rules

Extending OWL with Rules

- The rules hidden in OWL 2: SROIQ Rules
- Retaining tractability I: OWL 2 EL Rules
- Retaining tractability II: DLP 2

Rules inside OWL

- Retaining tractability III: ELP

putting it all together

Retaining decidability I: DL-safety

- **Reinterpret SWRL rules:**
  Rules apply only to individuals which are explicitly given in the knowledge base.
  – **Herbrand-style way of interpreting them**

- **OWL DL + DL-safe SWRL is decidable**
- **Native support e.g. by KAON2 and Pellet**

- See e.g. Boris Motik, Ulrike Sattler, and Rudi Studer. Query Answering for OWL-DL with Rules. Journal of Web Semantics 3(1):41–60, 2005.

DL-safe SWRL example

NutAllergic(sebastian)
NutProduct(peanutOil)
$\exists$orderedDish.ThaiCurry(sebastian)

ThaiCurry $\sqsubseteq$ $\exists$contains.{peanutOil}
$\top$ $\sqsubseteq$ $\forall$orderedDish.Dish

DL-safe $\left\{\begin{array}{l}\textbf{NutAllergic(x)} \wedge \textbf{NutProduct(y)} \rightarrow \textbf{dislikes(x,y)} \\ \textbf{dislikes(x,z)} \wedge \textbf{Dish(y)} \wedge \textbf{contains(y,z)} \rightarrow \textbf{dislikes(x,y)} \\ \textbf{orderedDish(x,y)} \wedge \textbf{dislikes(x,y)} \rightarrow \textbf{Unhappy(x)}\end{array}\right.$

Unhappy(sebastian) can*not* be concluded

DL-safe SWRL example

NutAllergic(sebastian)

NutProduct(peanutOil)

∃orderedDish.ThaiCurry(sebastian)

ThaiCurry ⊑ ∃contains.{peanutOil}

⊤ ⊑ ∀orderedDish.Dish

DL-safe $\left\{ \begin{array}{l} \text{NutAllergic(x)} \land \text{NutProduct(y)} \rightarrow \text{dislikes(x,y)} \\ \text{dislikes(x,z)} \land \text{Dish(y)} \land \text{contains(y,z)} \rightarrow \text{dislikes(x,y)} \\ \text{orderedDish(x,y)} \land \text{dislikes(x,y)} \rightarrow \text{Unhappy(x)} \end{array} \right.$

Conclusions:

dislikes(sebastian,peanutOil)

orderedDish(sebastian,y_s)

ThaiCurry(y_s)

Dish(y_s)

contains(y_s,peanutOil)

~~dislikes(sebastian,y_s)~~

Contents

- Motivation: OWL and Rules
- Preliminaries: Datalog

Intro

- More rules than you ever need: SWRL
- Retaining decidability I: DL-safety
- **Retaining decidability II: DL Rules**

Extending OWL with Rules

- The rules hidden in OWL 2: SROIQ Rules
- Retaining tractability I: OWL 2 EL Rules
- Retaining tractability II: DLP 2

Rules inside OWL

- Retaining tractability III: ELP

putting it all together

Retaining decidability II: DL Rules

- **General idea:**
  Find out which rules can be encoded in OWL (2 DL) anyway

- **Man(x) $\land$ hasBrother(x,y) $\land$ hasChild(y,z) $\rightarrow$ Uncle(x)**
  - **Man $\sqcap$ $\exists$hasBrother.$\exists$hasChild.$\top$ $\sqsubseteq$ Uncle**

- **ThaiCurry(x) $\rightarrow$ $\exists$contains.FishProduct(x)**
  - **ThaiCurry $\sqsubseteq$ $\exists$contains.FishProduct**

- **kills(x,x) $\rightarrow$ suicide(x)**
  - **$\exists$kills.Self $\sqsubseteq$ suicide**

  suicide(x) $\rightarrow$ kills(x,x)
  suicide $\sqsubseteq$ $\exists$kills.Self

Note: with these two axioms,
suicide* is basically the same as *kills

DL Rules: more examples

- NutAllergic(x) ∧ NutProduct(y) → dislikes(x,y)
  - NutAllergic ≡ ∃nutAllergic.Self
    NutProduct ≡ ∃nutProduct.Self
    nutAllergic o U o nutProduct ⊑ dislikes

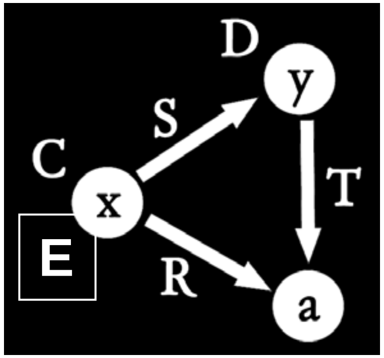
- dislikes(x,z) ∧ Dish(y) ∧ contains(y,z) → dislikes(x,y)
  - Dish ≡ ∃dish.Self
    dislikes o contains$^-$ o dish ⊑ dislikes

- worksAt(x,y) ∧ University(y) ∧ supervises(x,z) ∧ PhDStudent(z)
  → professorOf(x,z)
  - ∃worksAt.University ≡ ∃worksAtUniversity.Self
    PhDStudent ≡ ∃phDStudent.Self
    worksAtUniversity o supervises o phDStudent ⊑ professorOf

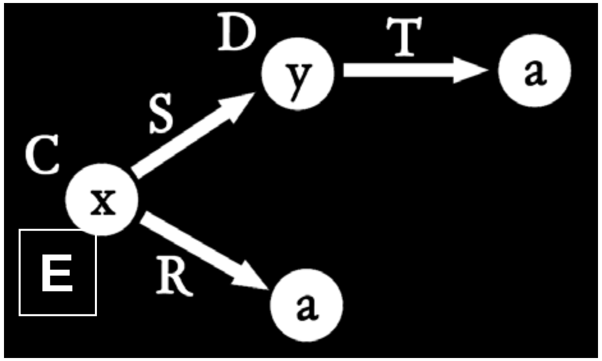
DL Rules: definition

- **Tree-shaped bodies**
- **First argument of the conclusion is the root**

- **$C(x) \wedge R(x,a) \wedge S(x,y) \wedge D(y) \wedge T(y,a) \rightarrow E(x)$**
  - **$C \sqcap \exists R.\{a\} \sqcap \exists S.(D \sqcap \exists T.\{a\}) \sqsubseteq E$**



duplicating nominals is ok

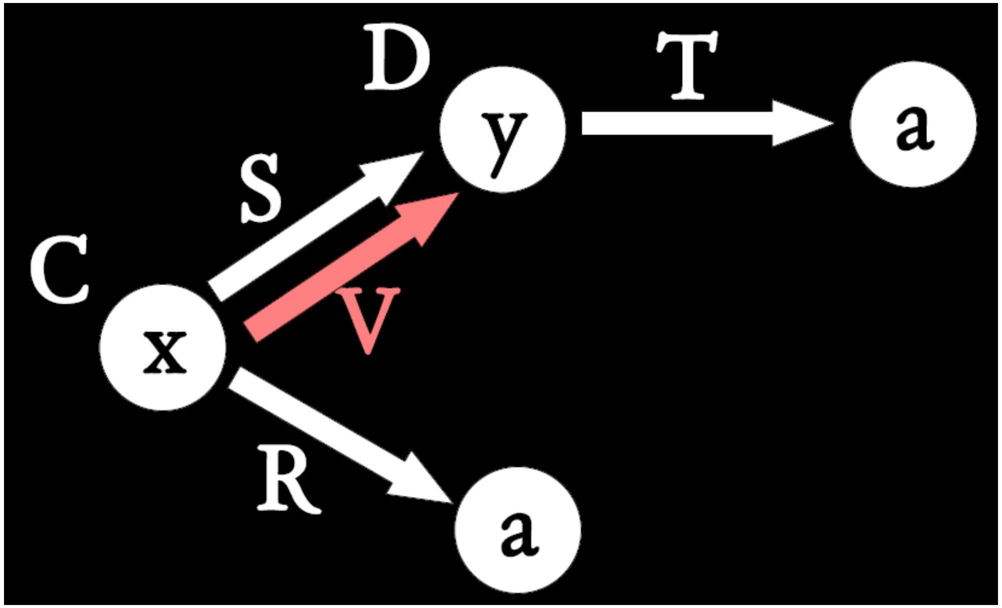


DL Rules: definition

- **Tree-shaped bodies**
- **First argument of the conclusion is the root**

- **$C(x) \wedge R(x,a) \wedge S(x,y) \wedge D(y) \wedge T(y,a) \rightarrow V(x,y)$**

$C \sqcap \exists R.\{a\} \sqsubseteq \exists R1.\mathrm{Self}$
$D \sqcap \exists T.\{a\} \sqsubseteq \exists R2.\mathrm{Self}$
$R1 \circ S \circ R2 \sqsubseteq V$



DL Rules: definition

- Tree-shaped bodies
- First argument of the conclusion is the root

- complex classes are allowed in the rules
  - Mouse(x) ∧ ∃hasNose.TrunkLike(y) → smallerThan(x,y)
  - ThaiCurry(x) → ∃contains.FishProduct(x)

    Note: This allows to reason with unknowns (unlike Datalog)

  - allowed class constructors depend on the chosen underlying description logic!

DL Rules: definition

Given a description logic $\mathcal{D}$,

the language $\mathcal{D}$ Rules consists of

- **all axioms expressible in $\mathcal{D}$,**
- **plus all rules with**
  - **tree-shaped bodies, where**
  - **the first argument of the conclusion is the root, and**
  - **complex classes from $\mathcal{D}$ are allowed in the rules.**
  - **<plus possibly some restrictions concerning e.g. the use of simple roles – depending on $\mathcal{D}$>**

Contents

- Motivation: OWL and Rules
- Preliminaries: Datalog

Intro

- More rules than you ever need: SWRL
- Retaining decidability I: DL-safety
- Retaining decidability II: DL Rules

Extending OWL with Rules

- **The rules hidden in OWL 2: SROIQ Rules**
- Retaining tractability I: OWL 2 EL Rules
- Retaining tractability II: DLP 2

Rules inside OWL

- Retaining tractability III: ELP

putting it all together

The rules hidden in OWL 2: SROIQ Rules

- N2ExpTime complete

- In fact, SROIQ Rules can be translated into SROIQ
  i.e. they don't add expressivity.

  Translation is polynomial.

- SROIQ Rules are essentially helpful syntactic sugar for OWL 2.

SROIQ Rules example

NutAllergic(sebastian)
NutProduct(peanutOil)
$\exists$orderedDish.ThaiCurry(sebastian)

ThaiCurry $\sqsubseteq$ $\exists$contains.{peanutOil}
$\top$ $\sqsubseteq$ $\forall$orderedDish.Dish

NutAllergic(x) $\wedge$ NutProduct(y) $\rightarrow$ dislikes(x,y)
dislikes(x,z) $\wedge$ Dish(y) $\wedge$ contains(y,z) $\rightarrow$ dislikes(x,y)
orderedDish(x,y) $\wedge$ dislikes(x,y) $\rightarrow$ Unhappy(x)

!not a SROIQ Rule!

SROIQ Rules normal form

- Each SROIQ Rule can be written ("linearised") such that
  - the body-tree is linear,
  - if the head is of the form R(x,y), then y is the leaf of the tree, and
  - if the head is of the form C(x), then the tree is only the root.

- $\text{worksAt}(x,y) \land \text{University}(y) \land \text{supervises}(x,z) \land \text{PhDStudent}(z) \rightarrow \text{professorOf}(x,z)$
  - $\exists\text{worksAt}.\text{University}(x) \land \text{supervises}(x,z) \land \text{PhDStudent}(z) \rightarrow \text{professorOf}(x,z)$

- $C(x) \land R(x,a) \land S(x,y) \land D(y) \land T(y,a) \rightarrow V(x,y)$
  - $(C \sqcap \exists R.\{a\})(x) \land S(x,y) \land (D \sqcap \exists T.\{a\})(y) \rightarrow V(x,y)$

Contents

- Motivation: OWL and Rules
- Preliminaries: Datalog

Intro

- More rules than you ever need: SWRL
- Retaining decidability I: DL-safety
- Retaining decidability II: DL Rules

Extending OWL with Rules

- The rules hidden in OWL 2: SROIQ Rules
- **Retaining tractability I: OWL 2 EL Rules**
- Retaining tractability II: DLP 2

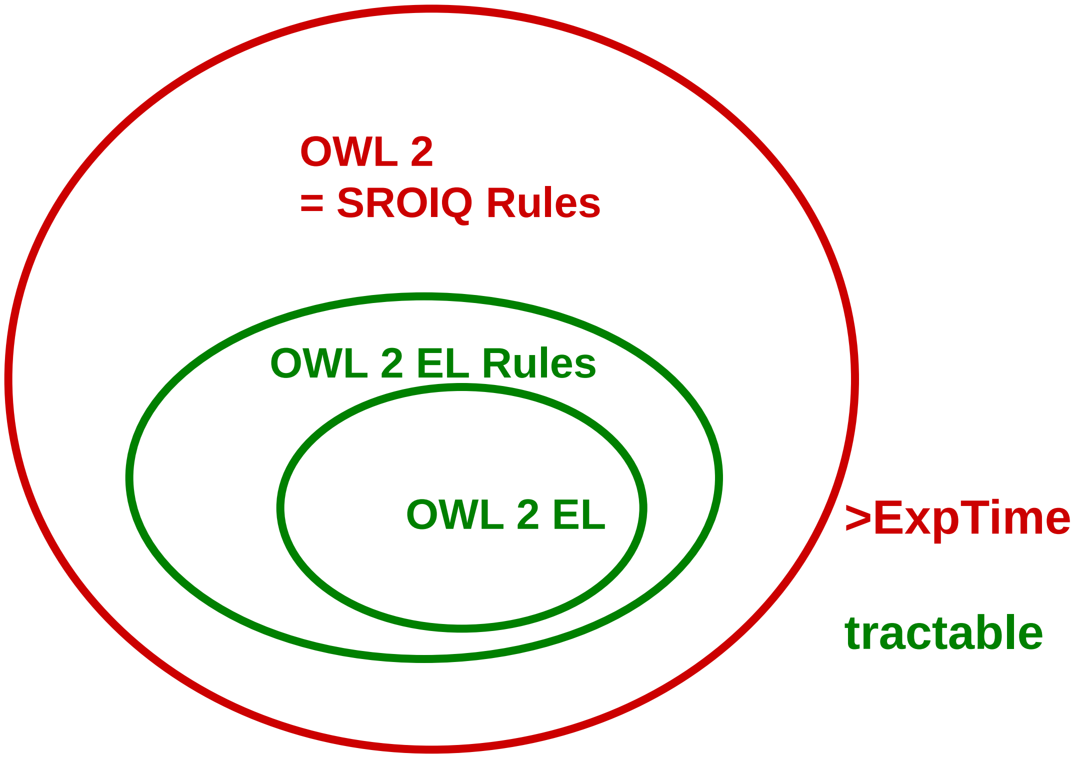
Rules inside OWL

- Retaining tractability III: ELP

putting it all together

Retaining tractability I: OWL 2 EL Rules

- **EL++ Rules are PTime complete**
- **EL++ Rules offer expressivity which is not readily available in EL++.**



OWL 2 EL Rules: normal form

- **Every EL++ Rule can be converted into a normal form,where**
  - **occurring classes in the rule body are either atomic or nominals,**
  - **all variables in a rule's head occur also in its body, and**
  - **rule heads can only be of one of the forms A(x), ∃R.A(x), R(x,y), where A is an atomic class or a nominal or ⊤ or ⊥.**
- **Translation is polynomial.**
- **∃worksAt.University(x) ∧ supervises(x,z) ∧ PhDStudent(z) → professorOf(x,z)**
  - **worksAt(x,y) ∧ University(y) ∧ supervises(x,z) ∧ PhDStudent(z) → professorOf(x,z)**
- **ThaiCurry(x) → ∃contains.FishProduct(x)**

OWL 2 EL Rules in a nutshell

Essentially, OWL 2 EL Rules is

- Binary Datalog with tree-shaped rule bodies,
- extended by
  - occurrence of nominals as atoms and
  - existential class expressions in the head.

- The existentials really make the difference.

- Arguably the better alternative to OWL 2 EL (aka EL++)?
  - (which is covered anyway)

Contents

- Motivation: OWL and Rules
- Preliminaries: Datalog

Intro

- More rules than you ever need: SWRL
- Retaining decidability I: DL-safety
- Retaining decidability II: DL Rules

Extending OWL with Rules

- The rules hidden in OWL 2: SROIQ Rules
- Retaining tractability I: OWL 2 EL Rules
- **Retaining tractability II: DLP 2**

Rules inside OWL

- Retaining tractability III: ELP

putting it all together

Retaining tractability II: DLP 2

- DLP 2 is
  - DLP (aka OWL 2 RL) extended with
  - DL rules, which use
    - left-hand-side class expressions in the bodies and
    - right-hand-side class expressions in the head.

- Polynomial transformation into 5-variable Horn rules.

- PTime.

- Quite a bit more expressive than DLP / OWL 2 RL ...

Contents

- Motivation: OWL and Rules
- Preliminaries: Datalog

Intro

- More rules than you ever need: SWRL
- Retaining decidability I: DL-safety
- Retaining decidability II: DL Rules

Extending OWL with Rules

- The rules hidden in OWL 2: SROIQ Rules
- Retaining tractability I: OWL 2 EL Rules
- Retaining tractability II: DLP 2

Rules inside OWL

- **Retaining tractability III: ELP**

putting it all together

Retaining tractability III: ELP

Putting it all together:

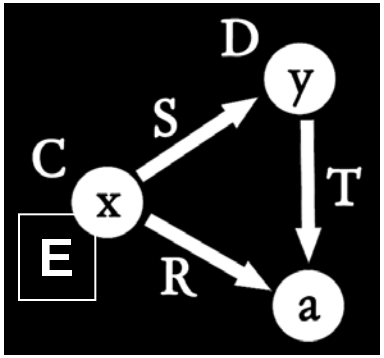
- **ELP is**
  - **OWL 2 EL Rules +**
  - **a generalisation of DL-safety +**
  - **variable-restricted DL-safe Datalog +**
  - **role conjunctions (for simple roles).**

- **PTime complete.**
- **Contains OWL 2 EL and OWL 2 RL.**
- **Covers variable-restricted Datalog.**

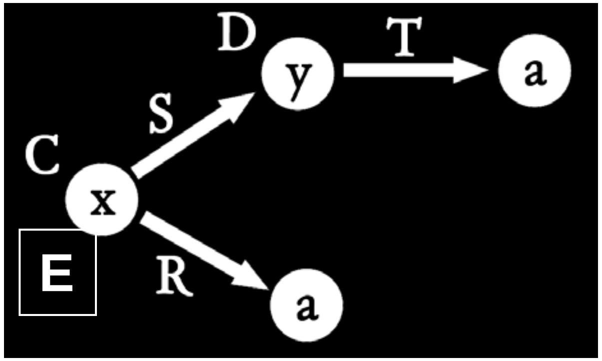
DL-safe variables

- A generalisation of DL-safety.
- DL-safe variables are special variables which bind only to named individuals (like in DL-safe rules).
- DL-safe variables can replace individuals in EL++ rules.

- $C(x) \wedge R(x,x_s) \wedge S(x,y) \wedge D(y) \wedge T(y,x_s) \rightarrow E(x)$
  with x_s a safe variable is allowed, because
  $C(x) \wedge R(x,a) \wedge S(x,y) \wedge D(y) \wedge T(y,a) \rightarrow E(x)$
  is an EL++ rule.



duplicating nominals is ok



Variable-restricted DL-safe Datalog

- n-Datalog is Datalog, where the number of variables occurring in rules is globally bounded by n.

- complexity of n-Datalog is PTime (for fixed n)
  - (but exponential in n)

- in a sense, this is cheating.
- in another sense, this means that using a few DL-safe Datalog rules together with an EL++ rules knowledge base shouldn't really be a problem in terms of reasoning performance.

Role conjunctions

- **orderedDish(x,y) $\wedge$ dislikes(x,y) $\rightarrow$ Unhappy(x)**

- **In fact, role conjunctions can also be added to OWL 2 DL without increase in complexity.**

- Sebastian Rudolph, Markus Krötzsch, Pascal Hitzler, Cheap Boolean Role Constructors for Description Logics. In: Steffen Hölldobler and Carsten Lutz and Heinrich Wansing (eds.), Proceedings of 11th European Conference on Logics in Artificial Intelligence (JELIA), volume 5293 of LNAI, pp. 362-374. Springer, September 2008.

Retaining tractability III: ELP

- ELP_n is
  - OWL 2 EL Rules generalised by DL-safe variables +
  - DL-safe Datalog rules with at most n variables +
  - role conjunctions (for simple roles).
- PTime complete (for fixed n).
  - exponential in n
- Contains OWL 2 EL and OWL 2 RL.
- Covers all Datalog rules with at most n variables. (!)

ELP example

NutAllergic(sebastian)

NutProduct(peanutOil)

$\exists$orderedDish.ThaiCurry(sebastian)

ThaiCurry $\sqsubseteq$ $\exists$contains.{peanutOil}

$\top$ $\sqsubseteq$ $\forall$orderedDish.Dish

[okay] **NutAllergic(x) $\wedge$ NutProduct(y) $\rightarrow$ dislikes(x,y)**

dislikes(x,z) $\wedge$ Dish(y) $\wedge$ contains(y,z) $\rightarrow$ dislikes(x,y)

orderedDish(x,y) $\wedge$ dislikes(x,y) $\rightarrow$ Unhappy(x)

[okay – role conjunction]

not an EL++ rule

ELP example

- **dislikes(x,z) $\wedge$ Dish(y) $\wedge$ contains(y,z) $\rightarrow$ dislikes(x,y)**
  as SROIQ rule translates to

  Dish $\equiv$ $\exists$dish.Self
  dislikes o contains^{-} o dish $\sqsubseteq$ dislikes

  but we don't have inverse roles in ELP!

- **solution: make z a DL-safe variable:**

  dislikes(x,!z) $\wedge$ Dish(y) $\wedge$ contains(y,!z) $\rightarrow$ dislikes(x,y)

  this is fine ☺

DL-safe SWRL example

NutAllergic(sebastian)
NutProduct(peanutOil)
$\exists$orderedDish.ThaiCurry(sebastian)

ThaiCurry $\sqsubseteq$ $\exists$contains.{peanutOil}
$\top$ $\sqsubseteq$ $\forall$orderedDish.Dish

NutAllergic(x) $\land$ NutProduct(y) $\rightarrow$ dislikes(x,y)
dislikes(x,!z) $\land$ Dish(y) $\land$ contains(y,!z) $\rightarrow$ dislikes(x,y)
orderedDish(x,y) $\land$ dislikes(x,y) $\rightarrow$ Unhappy(x)

Conclusions:
dislikes(sebastian,peanutOil)
orderedDish(sebastian,y_s)
ThaiCurry(y_s)
Dish(y_s)
contains(y_s,peanutOil)
dislikes(sebastian,y_s)

ELP example

NutAllergic(sebastian)

NutProduct(peanutOil)

$\exists$orderedDish.ThaiCurry(sebastian)

ThaiCurry $\sqsubseteq$ $\exists$contains.{peanutOil}

$\top \sqsubseteq \forall$orderedDish.Dish

NutAllergic(x) $\wedge$ NutProduct(y) $\rightarrow$ dislikes(x,y)

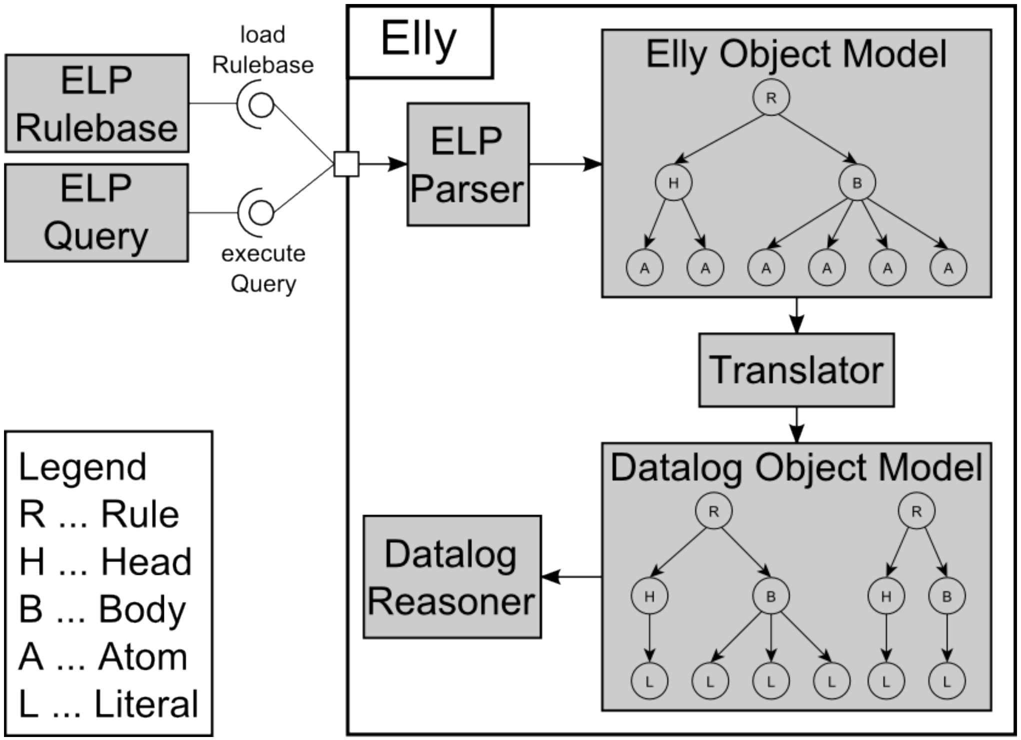
dislikes(x,!z) $\wedge$ Dish(y) $\wedge$ contains(y,!z) $\rightarrow$ dislikes(x,y)

orderedDish(x,y) $\wedge$ dislikes(x,y) $\rightarrow$ Unhappy(x)

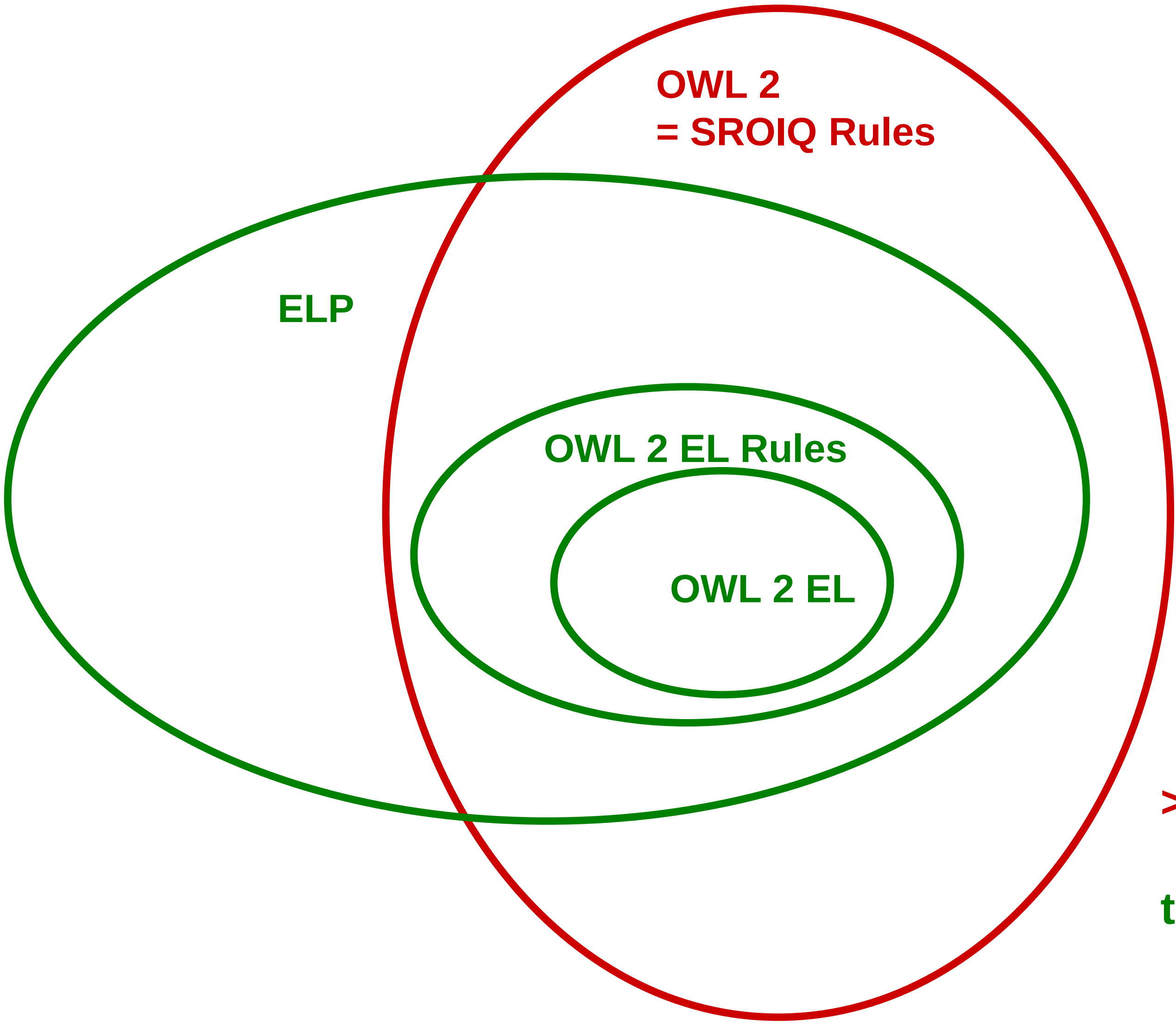
Conclusion: Unhappy(sebastian)

ELP Reasoner ELLY

- **Implementation currently being finalised.**
- **Based on IRIS Datalog reasoner.**
- **In cooperation with STI Innsbruck (Barry Bishop, Daniel Winkler, Gulay Unel).**



The Big Picture



>ExpTime

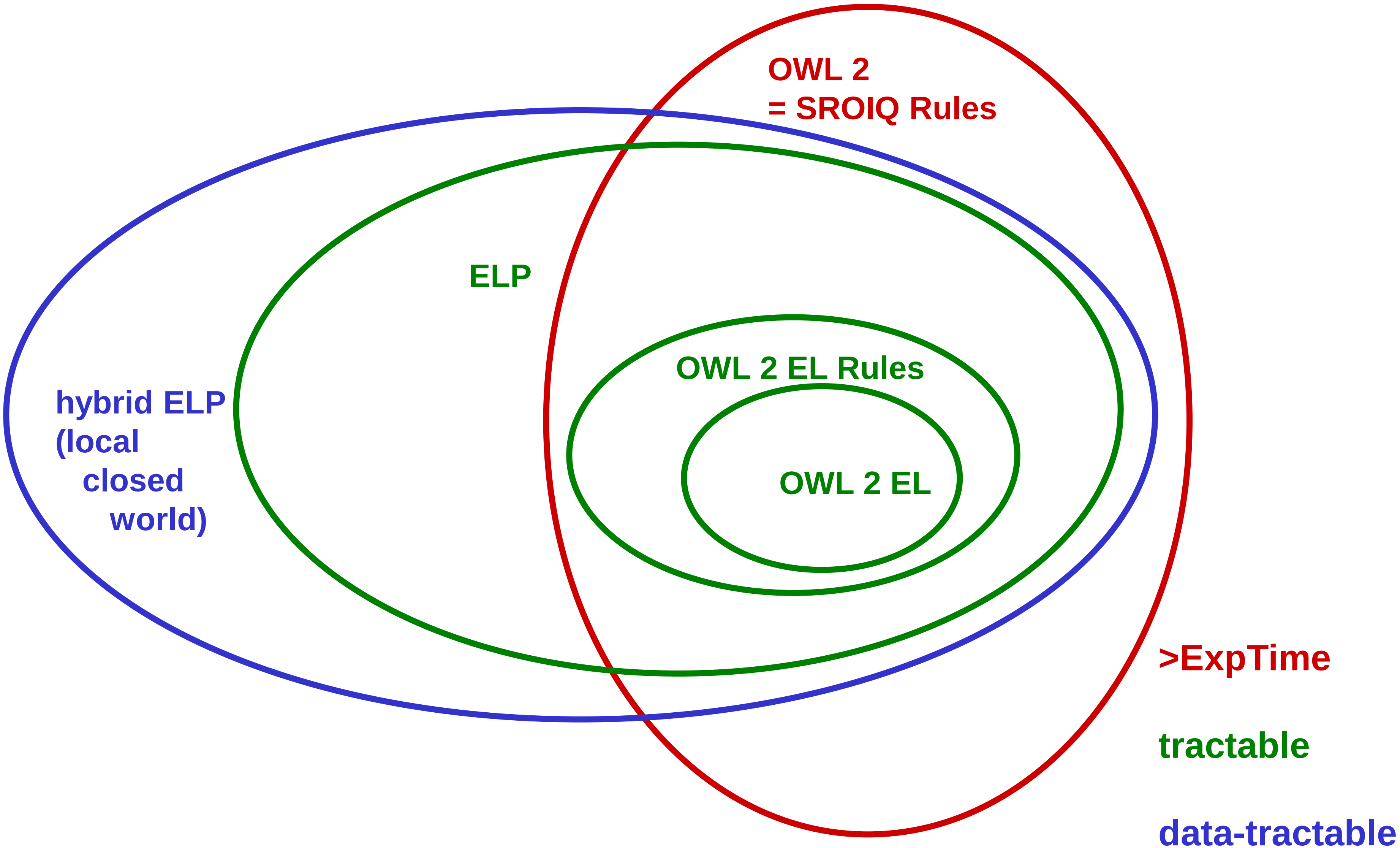
tractable

Closed World and ELP

- **There's an extension of ELP using (non-monotonic) closed-world reasoning – based on a well-founded semantics for hybrid MKNF knowledge bases.**

- Matthias Knorr, Jose Julio Alferes, Pascal Hitzler, A Coherent Well-founded model for Hybrid MKNF knowledge bases. In: Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, Nikos Avouris (eds.), Proceedings of the 18th European Conference on Artificial Intelligence, ECAI2008, Patras, Greece, July 2008. IOS Press, 2008, pp. 99-103.

The Big Picture II



Thanks!

http://www.semantic-web-book.org/page/ISWC2010_Tutorial

References OWL and Rules

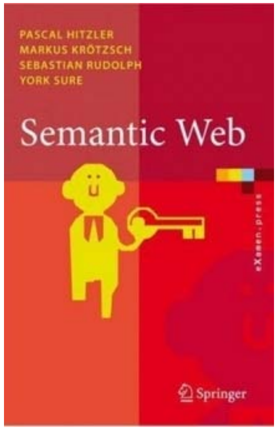
- Markus Krötzsch, Sebastian Rudolph, Pascal Hitzler, Description Logic Rules. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, Nikos Avouris, eds.: Proceedings of the 18th European Conference on Artificial Intelligence (ECAI-08), pp. 80–84. IOS Press 2008.
- Markus Krötzsch, Sebastian Rudolph, Pascal Hitzler, ELP: Tractable Rules for OWL 2. In Amit Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy Finin, Krishnaprasad Thirunarayan, eds.: Proceedings of the 7th International Semantic Web Conference (ISWC-08), pp. 649–664. Springer 2008.
- http://www.w3.org/Submission/SWRL/
- Boris Motik, Ulrike Sattler, and Rudi Studer. Query Answering for OWL-DL with Rules. Journal of Web Semantics 3(1):41–60, 2005.

References OWL and Rules

- Sebastian Rudolph, Markus Krötzsch, Pascal Hitzler, Cheap Boolean Role Constructors for Description Logics. In: Steffen Hölldobler and Carsten Lutz and Heinrich Wansing (eds.), Proceedings of 11th European Conference on Logics in Artificial Intelligence (JELIA), volume 5293 of LNAI, pp. 362-374. Springer, September 2008.
- Matthias Knorr, Jose Julio Alferes, Pascal Hitzler, A Coherent Well-founded model for Hybrid MKNF knowledge bases. In: Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, Nikos Avouris (eds.), Proceedings of the 18th European Conference on Artificial Intelligence, ECAI2008, Patras, Greece, July 2008. IOS Press, 2008, pp. 99-103.

See also our books

- Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, York Sure,
  Semantic Web – Grundlagen. Springer, 2008.
  http://www.semantic-web-grundlagen.de/



- Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph,
  Foundations of Semantic Web Technologies.
  Chapman & Hall/CRC, 2009.
  http://www.semantic-web-book.org/wiki/FOST

  (Grab a flyer.)