

## OWL 2 – Theory and Practice



**Bernardo Cuenca Grau**  
University of Oxford  
UK



**Birte Glimm**  
University of Oxford  
UK



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



**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**

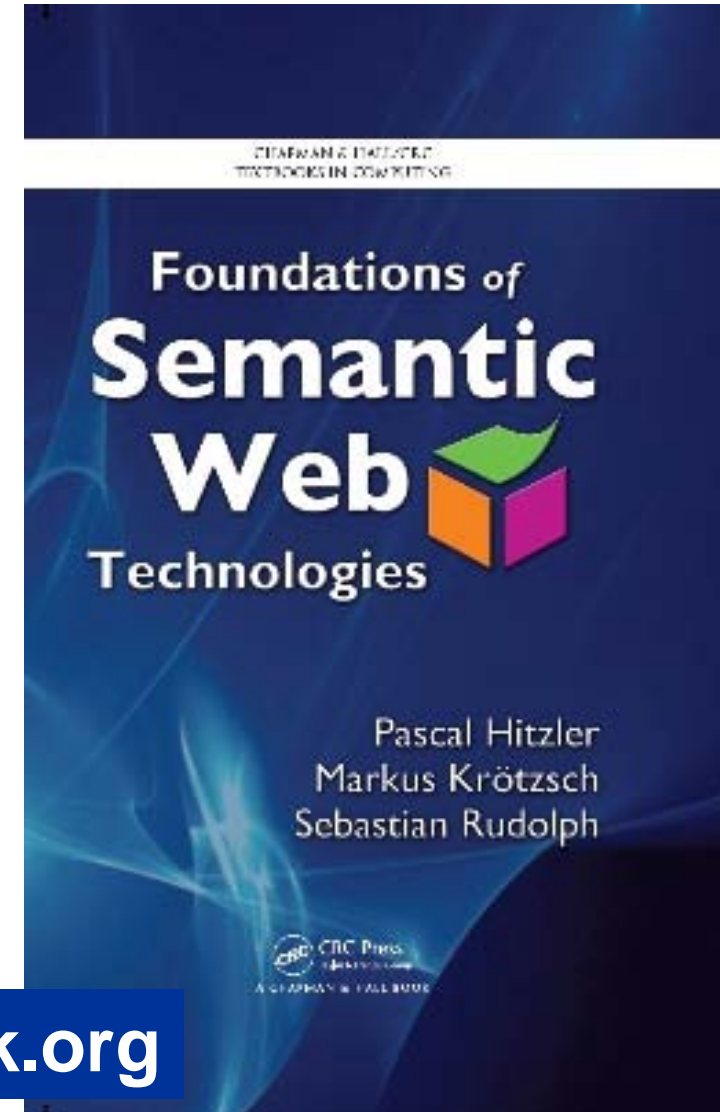
**16:30 - 18:30 applications (Bernardo)**

**Pascal Hitzler, Markus Krötzsch,  
Sebastian Rudolph**

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

**Grab a flyer!**

**<http://www.semantic-web-book.org>**



**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)**

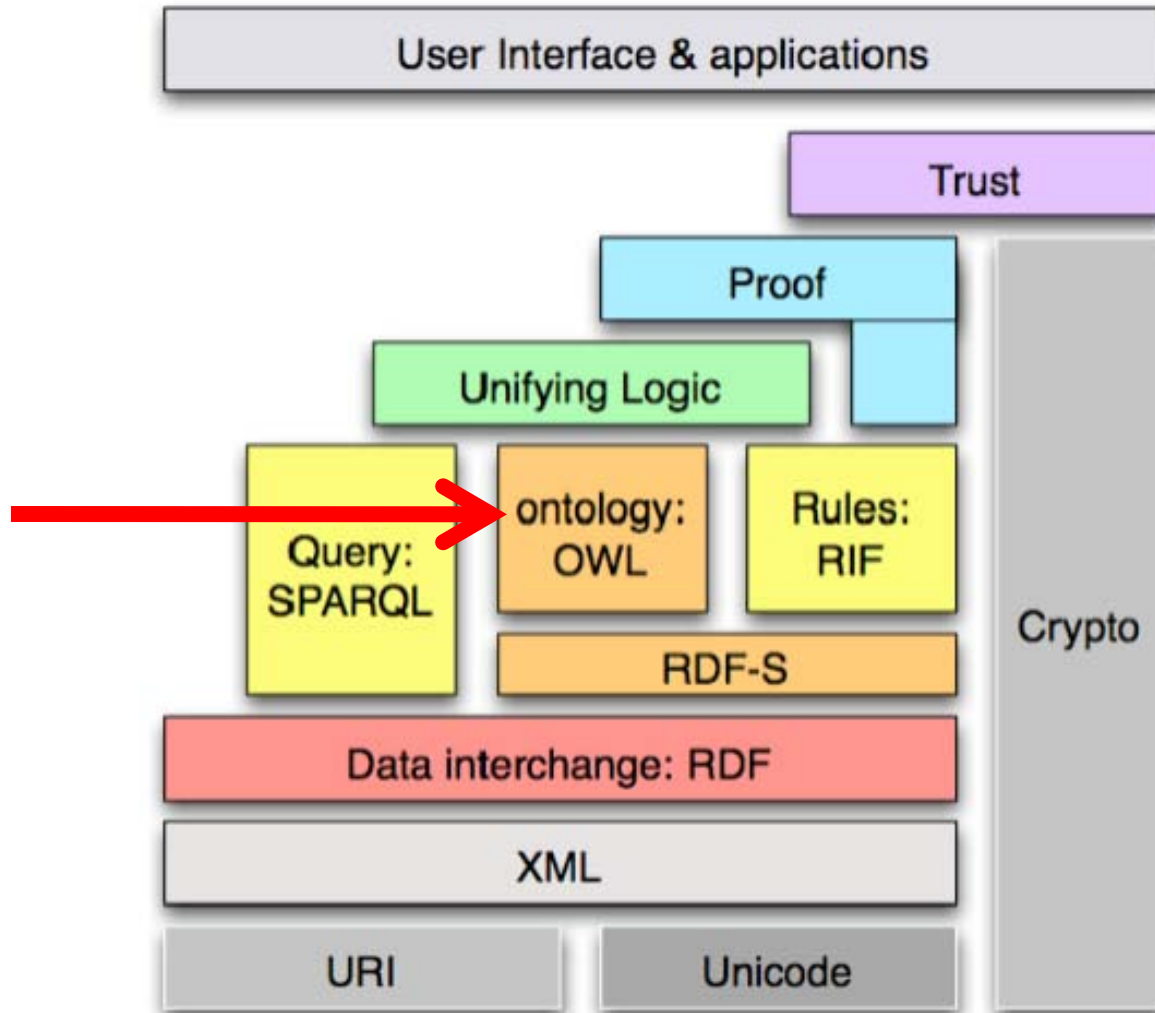
**<http://www.semantic-web-book.org>**

**Available from**

**[http://www.semantic-web-book.org/page/ISWC2010\\_Tutorial](http://www.semantic-web-book.org/page/ISWC2010_Tutorial)**

# Part 1

## OWL 2 – Syntax, Semantics, Reasoning



**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/>**

- **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**

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

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

- **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**

- **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"

## ABox statements

- **Person(mary)**
- **Person(mary)**
- **Woman  $\sqsubseteq$  Person**
  - **Person  $\equiv$  HumanBeing**
- **$\forall x$  (Woman(x)  $\rightarrow$  Person(x))**
- **hasWife(john,mary)**
- **hasWife(john,mary)**
- **hasWife  $\sqsubseteq$  hasSpouse**
  - **hasSpouse  $\equiv$  marriedWith**
- **$\forall x \forall y$  (hasWife(x,y)  $\rightarrow$  hasSpouse(x,y))**

## TBox statements

- **Person(mary)**
  - **Woman  $\sqsubseteq$  Person**
    - **Person  $\equiv$  HumanBeing**
  - **hasWife(john,mary)**
  - **hasWife  $\sqsubseteq$  hasSpouse**
    - **hasSpouse  $\equiv$  marriedWith**
- **:mary rdf:type :Person .**
  - **:Woman rdfs:subClassOf :Person .**
  - **:john :hasWife :mary .**
  - **:hasWife rdfs:subPropertyOf :hasSpouse .**

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

- **conjunction**

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

- **Mother**  $\equiv$  **Woman**  $\sqcap$  **Parent**

- **:Mother owl:equivalentClass \_:x .**

- **\_:x rdf:type owl:Class .**

- **\_:x owl:intersectionOf ( :Woman :Parent ) .**

- **disjunction**

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

- **Parent**  $\equiv$  **Mother**  $\sqcup$  **Father**

- **:Parent owl:equivalentClass \_:x .**

- **\_:x rdf:type owl:Class .**

- **\_:x owl:unionOf ( :Mother :Father ) .**

- **negation**

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

- **ChildlessPerson**  $\equiv$  **Person**  $\sqcap$   $\neg$ **Parent**

- **:ChildlessPerson owl:equivalentClass \_:x .**

- **\_:x rdf:type owl:Class .**

- **\_:x owl:intersectionOf ( :Person \_:y ) .**

- **\_:y owl:complementOf :Parent .**

- **existential quantification**
    - only to be used with a role – also called a *property restriction*
    - $\text{Parent} \equiv \exists \text{hasChild}.\text{Person}$
    - **`:Parent owl:equivalentClass _:x .`**  
**`_:x rdf:type owl:Restriction .`**  
**`_:x owl:onProperty :hasChild .`**  
**`_:x owl:someValuesFrom :Person .`**
- $$\forall x (\text{Parent}(x) \leftrightarrow \exists y (\text{hasChild}(x,y) \wedge \text{Person}(y)))$$
- **universal quantification**
    - only to be used with a role – also called a *property restriction*
    - $\text{Person} \sqcap \text{Happy} \equiv \forall \text{hasChild}.\text{Happy}$
    - **`_: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 .`**
- $$\forall x (\text{Person}(x) \wedge \text{Happy}(x) \leftrightarrow \forall y (\text{hasChild}(x,y) \rightarrow \text{Happy}(y)))$$
- **Class constructors can be nested arbitrarily**

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

## The description logic ALC

Complexity: ExpTime

- **ABox expressions:**  
Individual assignments  
Property assignments

Father(john)  
hasWife(john,mary)

- **TBox expressions**  
subclass relationships

$\sqsubseteq$

conjunction

$\sqcap$

disjunction

$\sqcup$

negation

$\neg$

Also:  $\top$ ,  $\perp$

property restrictions

$\forall$

$\exists$

## ALC + role chains = SR

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

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

- **includes top property and bottom property**
- **includes S = ALC + transitivity**
  - **hasAncestor o hasAncestor  $\sqsubseteq$  hasAncestor**
- **includes SH = S + role hierarchies**
  - **hasFather  $\sqsubseteq$  hasParent**

- **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}  $\equiv \perp$**

- **I – inverse roles**
  - **hasParent  $\equiv$  hasChild<sup>-</sup>**
  - **Orphan  $\equiv \forall$ hasChild<sup>-</sup>.Dead**
- **Q – qualified cardinality restrictions**
  - **$\leq 4$  hasChild.Parent(john)**
  - **HappyFather  $\equiv \geq 2$  hasChild.Female**
  - **Car  $\sqsubseteq =4$ hasTyre. $\top$**
- **Complexity SHIQ, SHOQ, SHIO: ExpTime.**  
**Complexity SHOIQ: NExpTime**  
**Complexity SROIQ: N<sup>2</sup>ExpTime**

Properties can be declared to be

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

called *property characteristics*

## (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:boolean`, `xsd:anyURI`, `xsd:dateTime`

and also e.g. `owl:real`

## (D) – datatypes

- `hasAge(john, "51"^^xsd:integer)`
- additional use of *constraining facets* (from XML Schema)
  - e.g. `Teenager ≡ Person ⊓ ∃hasAge.(xsd:integer: ≥12 and ≤19)`  
note: this is not standard DL notation!

## further expressive features

- **Self**
  - **PersonCommittingSuicide  $\equiv \exists \text{kills.Self}$**
- **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**

- ABox assignments of individuals to classes or properties
- ALC:  $\sqsubseteq, \equiv$  for classes  
 $\sqcap, \sqcup, \neg, \exists, \forall$   
 $\top, \perp$
- 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)

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

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

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

- **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.**

- **Description logics can be understood from a modal logic perspective.**
- **Each pair of  $\forall R$  and  $\exists R$  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.**

## RDFS semantics is weaker

- `:mary rdf:type :Person .`
- `:Mother rdfs:subClassOf :Woman .`
- `:john :hasWife :Mary .`
- `:hasWife rdfs:subPropertyOf :hasSpouse`
- `:hasWife rdfs:range :Woman .`
- `:hasWife rdfs:domain :Man .`
- `Person(mary)`
- `Mother  $\sqsubseteq$  Woman`
- `hasWife(john,mary)`
- `hasWife  $\sqsubseteq$  hasSpouse`
- `$\top \sqsubseteq \forall \text{hasWife.Woman}$`
- `$\top \sqsubseteq \forall \text{hasWife}^{\neg} .\text{Man}$`  or  `$\exists \text{hasWife} .\top \sqsubseteq \text{Man}$`

RDFS also allows to

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

- 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.

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

- **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.**

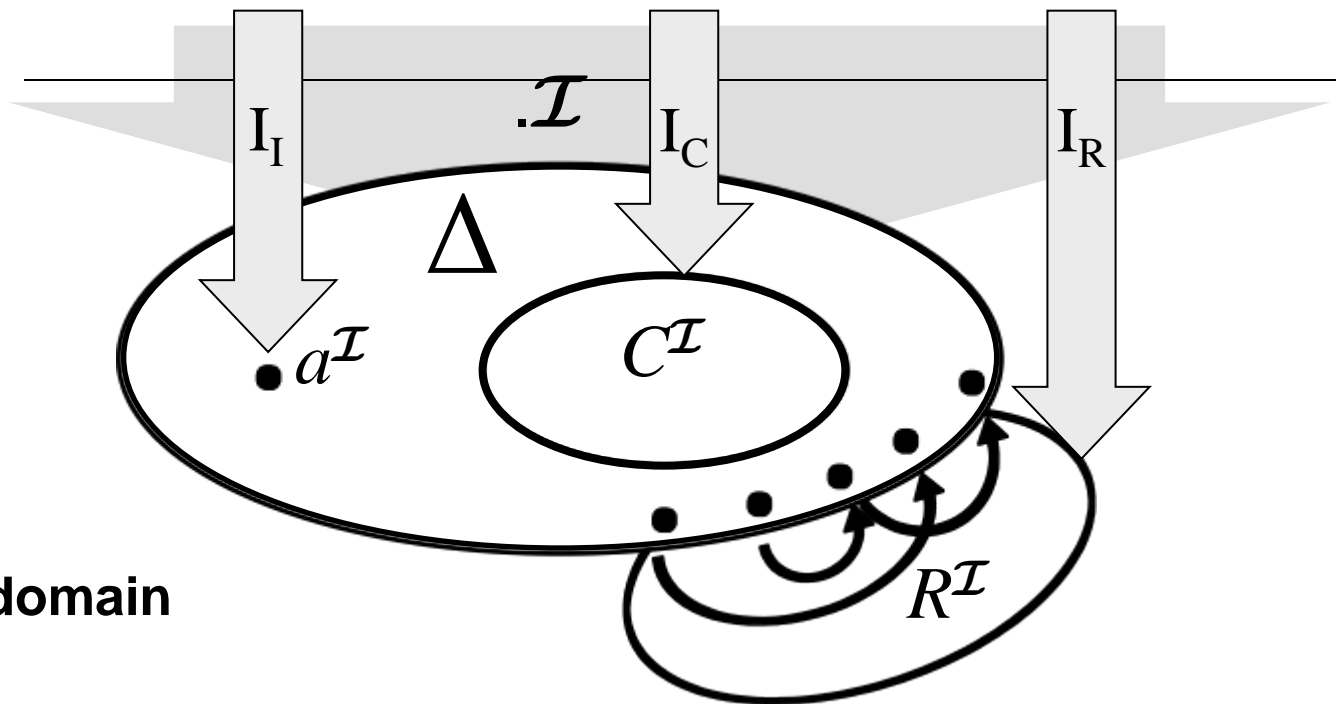
**To obtain decidability, syntactic restrictions apply.**

- **Type separation / punning**
- **No cycles in property chains.**
- **No transitive properties in cardinality 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  $<$  on the properties
  - every property chain axiom has to have one of the following forms:  
 $R \circ R \sqsubseteq R$                        $S^- \sqsubseteq R$                        $S_1 \circ S_2 \circ \dots \circ S_n \sqsubseteq R$   
 $R \circ S_1 \circ S_2 \circ \dots \circ S_n \sqsubseteq R$                        $S_1 \circ S_2 \circ \dots \circ S_n \circ R \sqsubseteq R$
  - thereby,  $S_i < R$  for all  $i = 1, 2, \dots, n$ .
- 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

- 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:  $P, Q$

- 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

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

then we could set

$$\begin{aligned} \Delta &= \{a, b, \text{Ian}\} \\ I_I(\text{rudiStuder}) &= \text{Ian} \\ I_I(\text{aifb}) &= b \\ I_C(\text{Professor}) &= \{a\} \\ I_C(\text{FacultyMember}) &= \{a, b\} \\ I_R(\text{hasAffiliation}) &= \{(a, b), (b, \text{Ian})\} \end{aligned}$$

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

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

- 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.

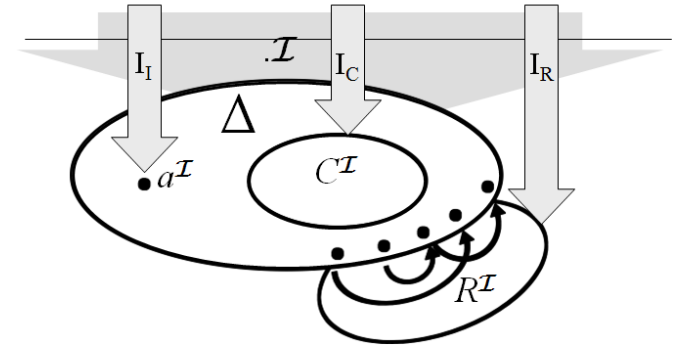
- Notion of *logical consequence* obtained via models (below).

- ...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  $R^I \subseteq S^I$
- $\text{Disjoint}(R,S)$  holds if  $R^I \cap S^I = \emptyset$
- $S_1 \circ S_2 \circ \dots \circ S_n \sqsubseteq R$  holds if  $S_1^I \circ S_2^I \circ \dots \circ S_n^I \subseteq R^I$

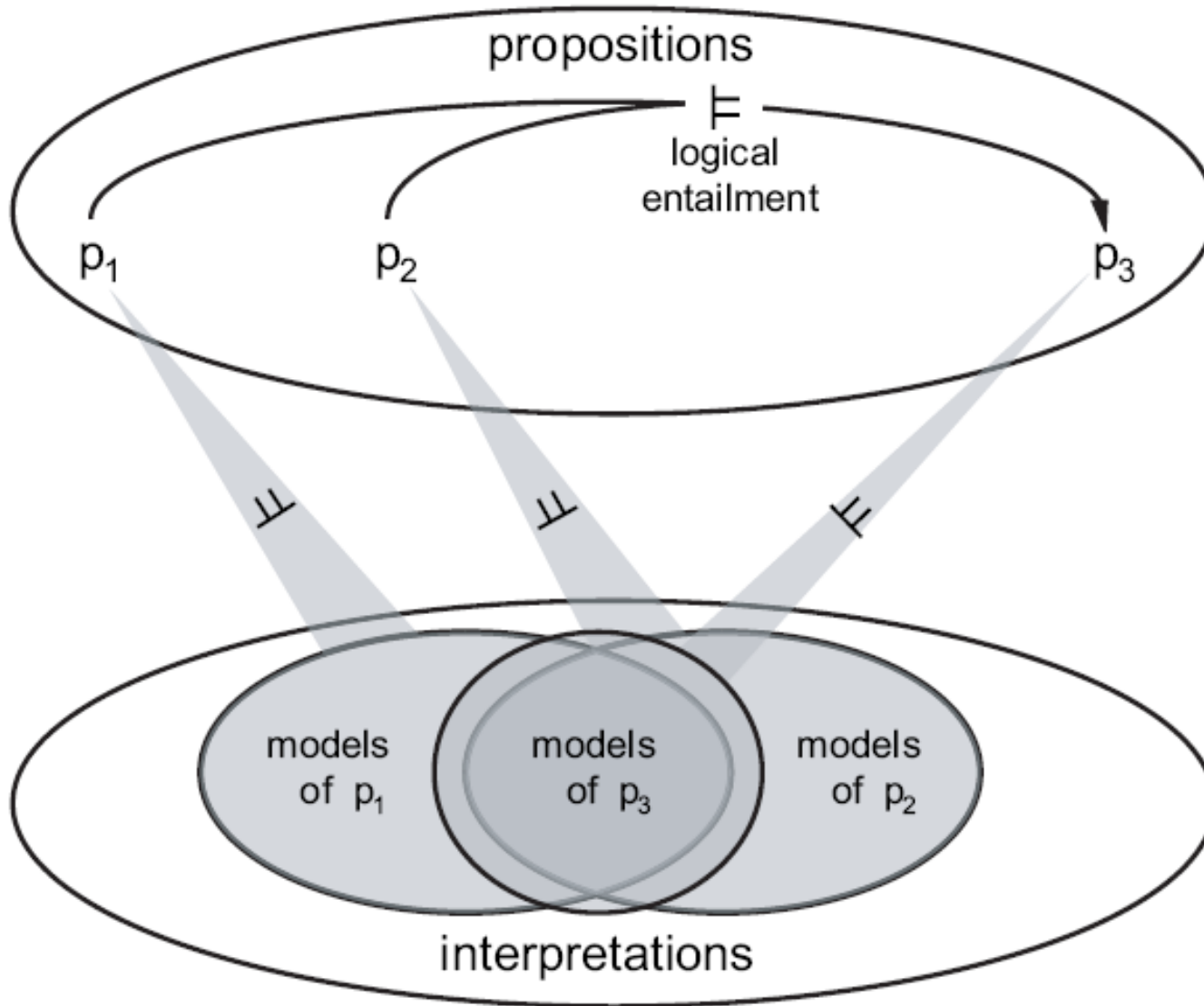
A *model* for an OWL KB is such a mapping  $I$  which satisfies all axioms in the KB.

An axiom  $\alpha$  is a *logical consequence* of a KB if every model of the KB is also a model of  $\alpha$ .



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

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

then we could set

$$\begin{aligned} \Delta &= \{a, b, \text{Ian}\} \\ I_I(\text{rudiStuder}) &= \text{Ian} \\ I_I(\text{aifb}) &= b \\ I_C(\text{Professor}) &= \{a\} \\ I_C(\text{FacultyMember}) &= \{a, b\} \\ I_R(\text{hasAffiliation}) &= \{(a, b), (b, \text{Ian})\} \end{aligned}$$

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

```
Professor  $\sqsubseteq$  FacultyMember  
Professor(rudiStuder)  
hasAffiliation(rudiStuder, aifb)
```

$$\Delta = \{a, r, s\}$$
$$I_I(\text{rudiStuder}) = r$$
$$I_I(\text{aifb}) = a$$
$$I_C(\text{Professor}) = \{r\}$$
$$I_C(\text{FacultyMember}) = \{r, s\}$$
$$I_R(\text{hasAffiliation}) = \{(r, a)\}$$

```
Professor  $\sqsubseteq$  FacultyMember  
Professor(rudiStuder)  
hasAffiliation(rudiStuder, aifb)
```

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

**Is FacultyMember(aifb) a logical consequence?**

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

$$\Delta = \{a, r\}$$

$$I_I(\text{rudiStuder}) = r$$

$$I_I(\text{aifb}) = a$$

$$I_C(\text{Professor}) = \{r\}$$

$$I_C(\text{FacultyMember}) = \{r\}$$

$$I_R(\text{hasAffiliation}) = \{(r, a)\}$$

```
Professor  $\sqsubseteq$  FacultyMember  
Professor(rudiStuder)  
hasAffiliation(rudiStuder, aifb)
```

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

**Is FacultyMember(rudiStuder) a logical consequence?**

- 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

$$\begin{aligned}\pi(C \sqsubseteq D) &= (\forall x)(\pi_x(C) \rightarrow \pi_x(D)) \\ \pi_x(A) &= A(x) \\ \pi_x(\neg C) &= \neg \pi_x(C) \\ \pi_x(C \sqcap D) &= \pi_x(C) \wedge \pi_x(D) \\ \pi_x(C \sqcup D) &= \pi_x(C) \vee \pi_x(D) \\ \pi_x(\forall R.C) &= (\forall x_1)(R(x, x_1) \rightarrow \pi_{x_1}(C)) \\ \pi_x(\exists R.C) &= (\exists x_1)(R(x, x_1) \wedge \pi_{x_1}(C)) \\ \pi_x(\geq n S.C) &= (\exists x_1) \dots (\exists x_n) \left( \bigwedge_{i \neq j} (x_i \neq x_j) \wedge \bigwedge_i (S(x, x_i) \wedge \pi_{x_i}(C)) \right) \\ \pi_x(\leq n S.C) &= \neg (\exists x_1) \dots (\exists x_{n+1}) \left( \bigwedge_{i \neq j} (x_i \neq x_j) \wedge \bigwedge_i (S(x, x_i) \wedge \pi_{x_i}(C)) \right) \\ \pi_x(\{a\}) &= (x = a) \\ \pi_x(\exists S.\text{Self}) &= S(x, x)\end{aligned}$$
$$\begin{aligned}\pi(R_1 \sqsubseteq R_2) &= (\forall x)(\forall y)(\pi_{x,y}(R_1) \rightarrow \pi_{x,y}(R_2)) \\ \pi_{x,y}(S) &= S(x, y) \\ \pi_{x,y}(R^-) &= \pi_{y,x}(R) \\ \pi_{x,y}(R_1 \circ \dots \circ R_n) &= (\exists x_1) \dots (\exists x_{n-1}) \\ &\quad \left( \pi_{x,x_1}(R_1) \wedge \bigwedge_{i=1}^{n-2} \pi_{x_i,x_{i+1}}(R_{i+1}) \wedge \pi_{x_{n-1},y}(R_n) \right) \\ \pi(\text{Ref}(R)) &= (\forall x)\pi_{x,x}(R) \\ \pi(\text{Asy}(R)) &= (\forall x)(\forall y)(\pi_{x,y}(R) \rightarrow \neg \pi_{y,x}(R)) \\ \pi(\text{Dis}(R_1, R_2)) &= \neg (\exists x)(\exists y)(\pi_{x,y}(R_1) \wedge \pi_{x,y}(R_2))\end{aligned}$$

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

- 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

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

```
Unicorn  $\sqsubseteq$  Fictitious  
Unicorn  $\sqsubseteq$  Animal  
Fictitious  $\sqcap$  Animal  $\sqsubseteq$   $\perp$ 
```

## 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/>

Person  $\sqsubseteq$   $\neg$ Movie

RRated  $\sqsubseteq$  CatMovie

CatMovie  $\sqsubseteq$  Movie

RRated  $\equiv$  ( $\exists$ hasScript.ThrillerScript)  $\sqcap$  ( $\forall$ hasViolenceLevel.High)

Domain(hasViolenceLevel, Movie)

**Fig. 1.** A justification for Person  $\sqsubseteq \perp$

- **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!**

**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.**

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



KNOWLEDGE SOCIETY

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

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

Are lobsters spiders?

What is "Käuzchen" in english?

- **SNOMED CT: commercial ontology, medical domain  
ca. 300,000 axioms**
- **InjuryOfFinger**  $\equiv$  **Injury**  $\sqcap$   $\exists$ **site.Finger**<sub>S</sub>  
**InjuryOfHand**  $\equiv$  **Injury**  $\sqcap$   $\exists$ **site.Hand**<sub>S</sub>  
**Finger**<sub>S</sub>  $\sqsubseteq$  **Hand**<sub>P</sub>  
**Hand**<sub>P</sub>  $\sqsubseteq$  **Hand**<sub>S</sub>  $\sqcap$   $\exists$ **part.Hand**<sub>E</sub>
- **Reasoning has been used e.g. for**
  - **classification (computing the hidden taxonomy)**  
e.g., **InjuryOfFinger**  $\sqsubseteq$  **InjuryOfHand**
  - **bug finding**

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

- **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<sub>R</sub>)
- **OWL RL** (DLP)

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

$$\neg \exists T \perp \sqsubseteq \neg \exists T \perp$$

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

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

- **Syntactic characterization:**

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



- cardinality restrictions: only on rhs and only  $\leq 1$  and  $\leq 0$  allowed
  - closed classes: only with one member
- **Reasoner conformance requires only soundness.**

- **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  $\exists r.T$
    - right hand side: intersection of class names, expressions of type  $\exists r.C$  and negations of lhs expressions
    - no closed classes!
- **Example:**  
 $\exists \text{married}.T \sqsubseteq \neg \text{Free} \sqcap \exists \text{has.Sorrow}$

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

**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!!!**

**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.**

**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**

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

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

- **Global consistency of a knowledge base.** **KB  $\models$  false?**
  - Is the knowledge base meaningful?
- **Class consistency** **C  $\equiv \perp$ ?**
  - 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 =  $\perp$ ?**
  - 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.

- **Global consistency of a knowledge base.** **KB unsatisfiable**
  - Failure to find a model.
- **Class consistency**  **$C \equiv \perp?$** 
  - $KB \cup \{C(a)\}$  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**  **$C \sqcap D = \perp?$** 
  - $KB \cup \{(C \sqcap D)(a)\}$  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.

- **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.**
- **Hence the reduction of all inference problems to the checking of unsatisfiability of the knowledge base!**

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

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

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.**

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

**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:

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

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

## 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.

- **Nodes represent elements of the domain of the model**
  - **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:**
  - **Every edge  $\langle x,y \rangle$  is labeled with a set  $L(\langle x,y \rangle)$  of role names.**
  - $R \in L(\langle x,y \rangle)$  means: " $(x,y)$  is in the extension of  $R$ "**

# Simple example

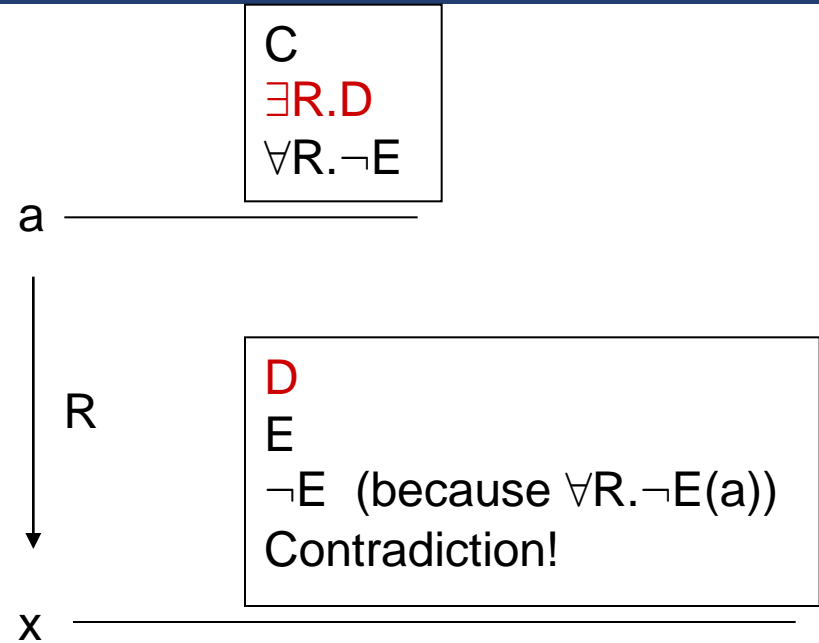
**C(a)**

**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

$C(a)$

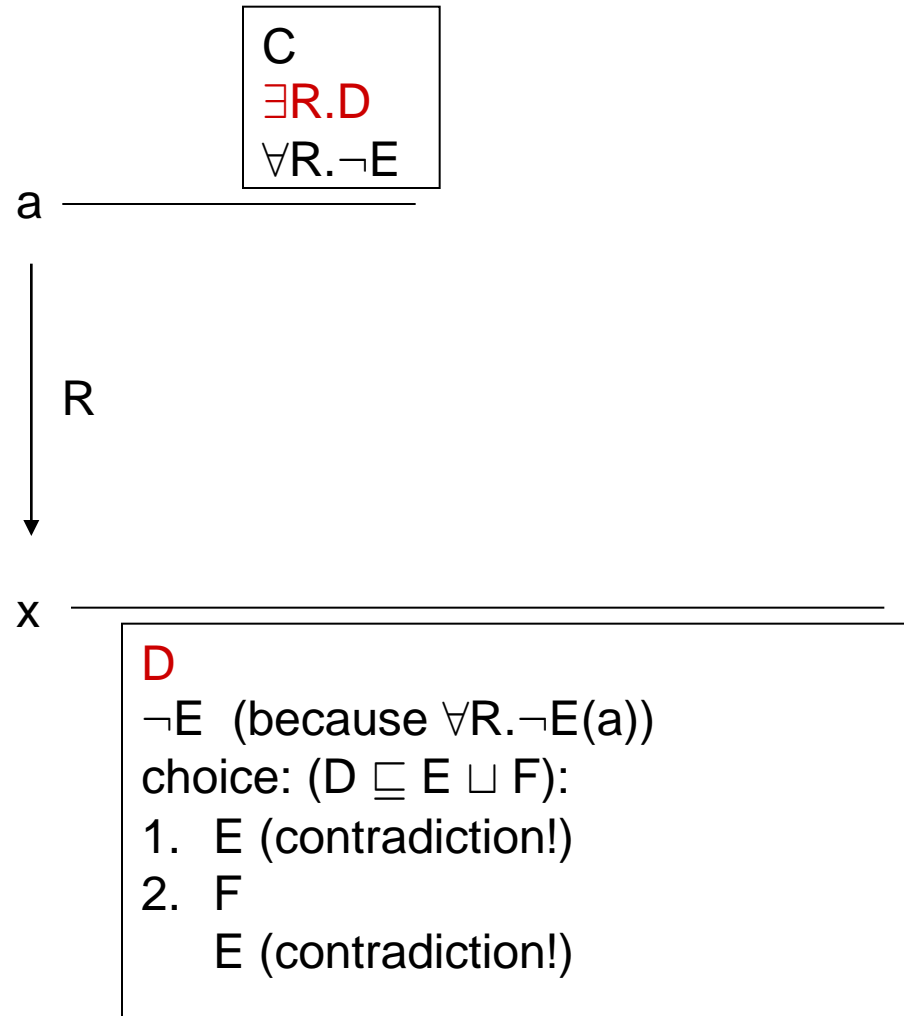
$C \sqsubseteq \exists R.D$

$D \sqsubseteq E \sqcup F$

$F \sqsubseteq E$

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

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



- **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  $\langle x, y \rangle$  are labeled with sets  $L(\langle x, y \rangle)$  of role names

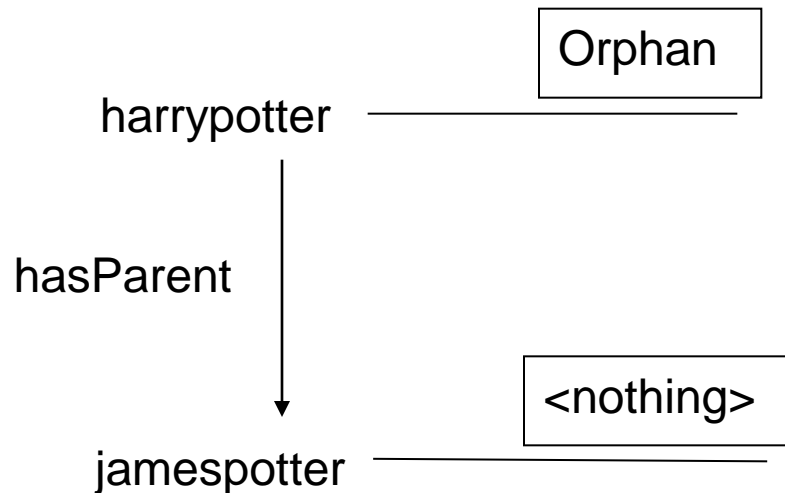
- **Make a node for every individual in the ABox.**
- **Every node is labeled with the corresponding class names from the ABox.**
- **There is an edge, labeled with  $R$ , between  $a$  and  $b$ , if  $R(a,b)$  is in the ABox.**
  
- **(If there is no ABox, the initial tableau consists of a node  $x$  with empty label.)**

**Human  $\sqsubseteq \exists \text{hasParent.Human}$**

**Orphan  $\sqsubseteq \text{Human} \sqcap \neg \exists \text{hasParent.Alive}$**

**Orphan(harrypotter)**

**hasParent(harrypotter,jamespotter)**

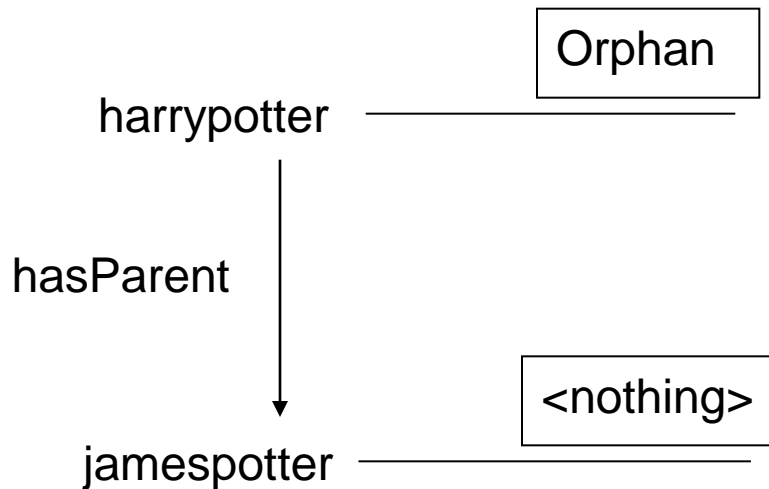


$\neg$ Human  $\sqcup$   $\exists$ hasParent.Human

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

Orphan(harrypotter)

hasParent(harrypotter,jamespotter)



- **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  $\neg C$ , or it contains  $\perp$ ), 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.**

**$\sqcap$ -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

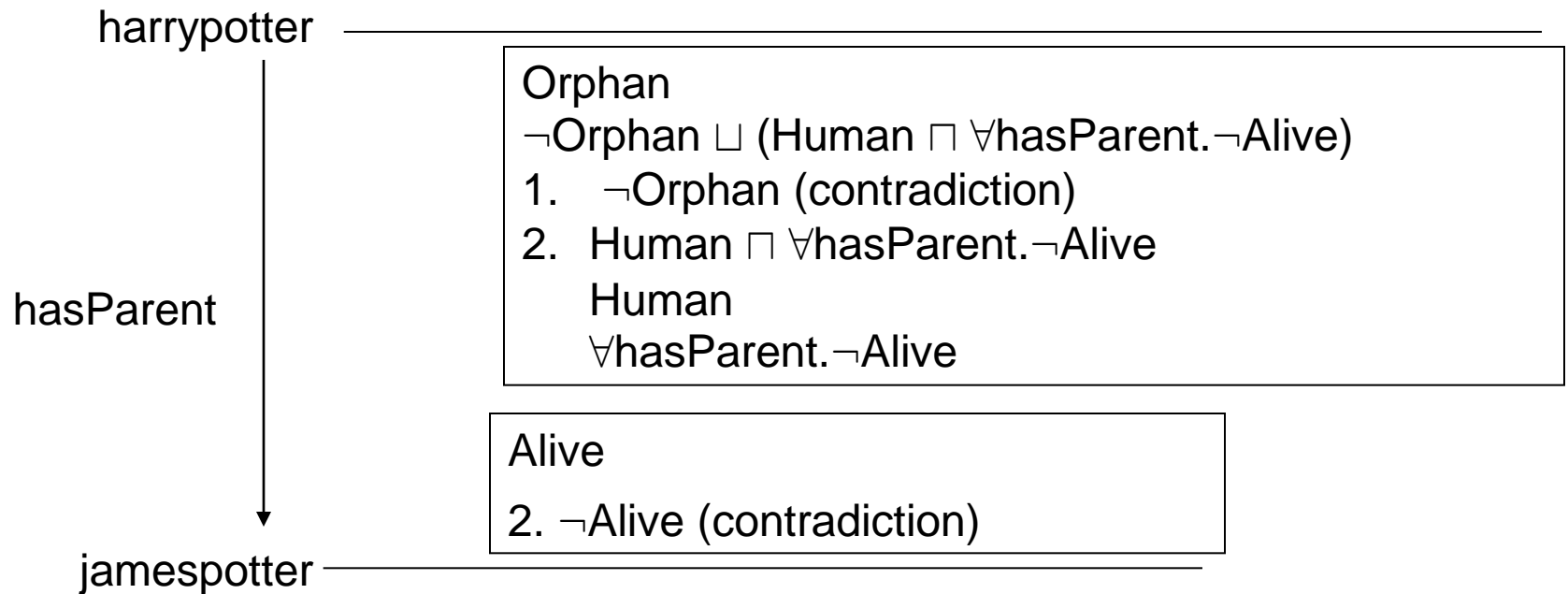
$\neg$ Alive(jamespotter)  
i.e. add: Alive(jamespotter)  
and search for contradiction

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

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

Orphan(harrypotter)

hasParent(harrypotter,jamespotter)



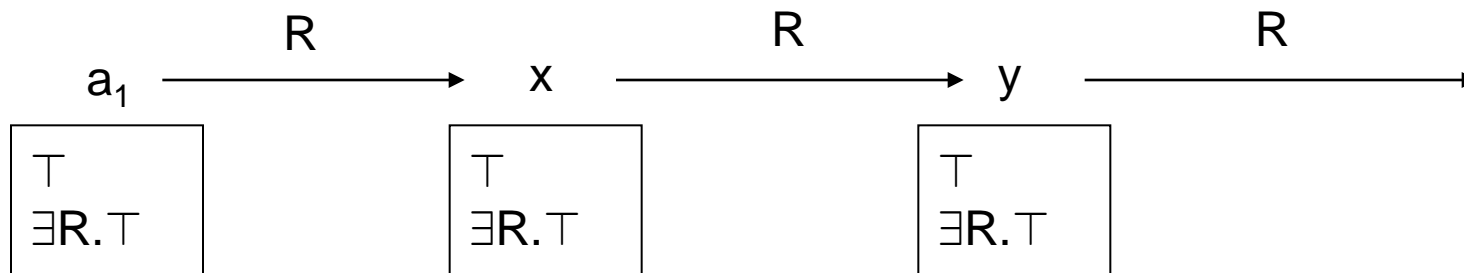
- Transformation to negation normal form
- Naive tableaux algorithm
- **Tableaux algorithm with blocking**

# There's a termination problem

**TBox:**  $\exists R.T$

**ABox:**  $\top(a_1)$

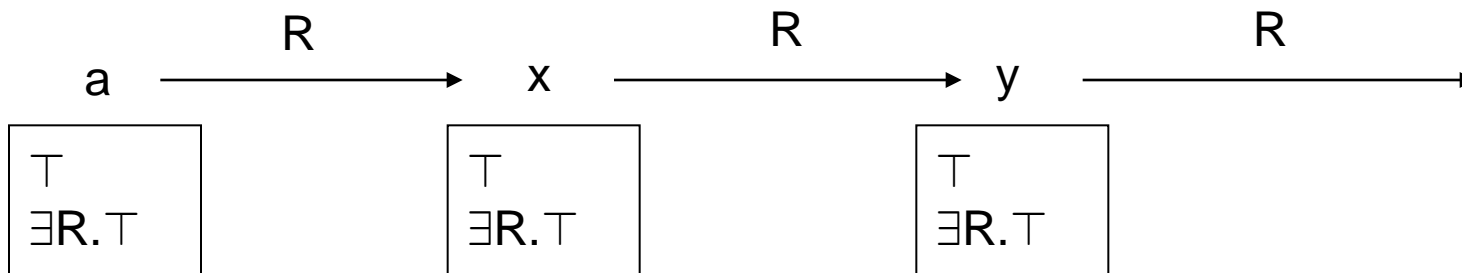
- **Obviously satisfiable:**  
Model  $M$  with domain elements  $a_1^M, a_2^M, \dots$   
and  $R^M(a_i^M, a_{i+1}^M)$  for all  $i \geq 1$
- **but tableaux algorithm does not terminate!**



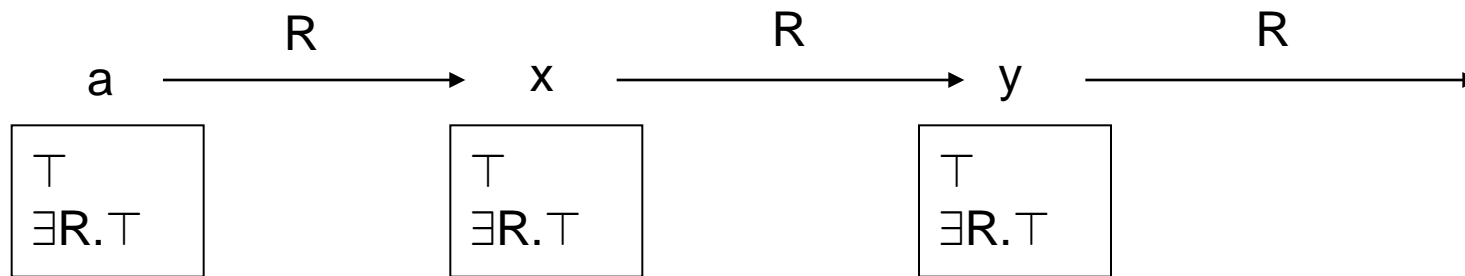
**Actually, things repeat!**

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

**⇒ 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  $a$ .

- 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  $\neg C$ ), 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.

**$\sqcap$ -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  $\mathcal{L}(x) \leftarrow C$ .

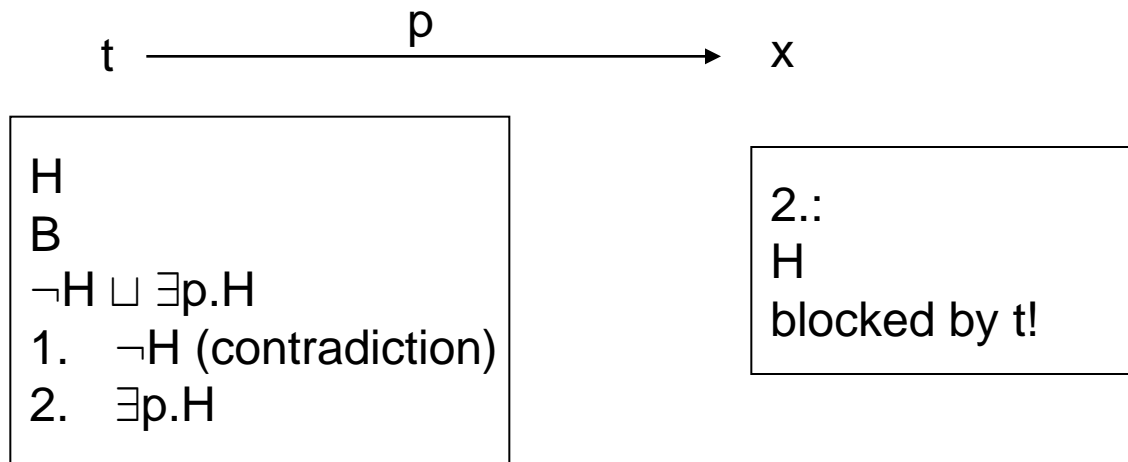
**Apply only if x is not blocked!**

# Example (0)

- Knowledge base {Human  $\sqsubseteq$   $\exists$ hasParent.Human, Bird(tweety)}
- We want to show that Human(tweety) does *not* hold, i.e. that  $\neg$ 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)  
  
 $\neg$ H(t) entailed?

# Example (0)

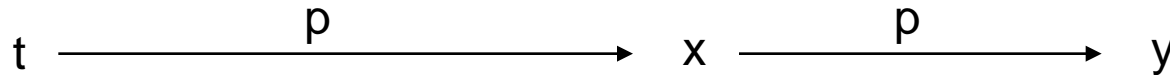
Knowledge base  $\{\neg H \sqcup \exists p.H, B(t), H(t)\}$



**expansion stops. Cannot find contradiction!**

# Example (0) the other case

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



$\neg H$   
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

$\text{Professor} \sqsubseteq (\text{Person} \sqcap \text{Unversitymember})$

$\sqcup (\text{Person} \sqcap \neg \text{PhDstudent})$

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

$\text{hasChild}(\text{john}, \text{peter})$

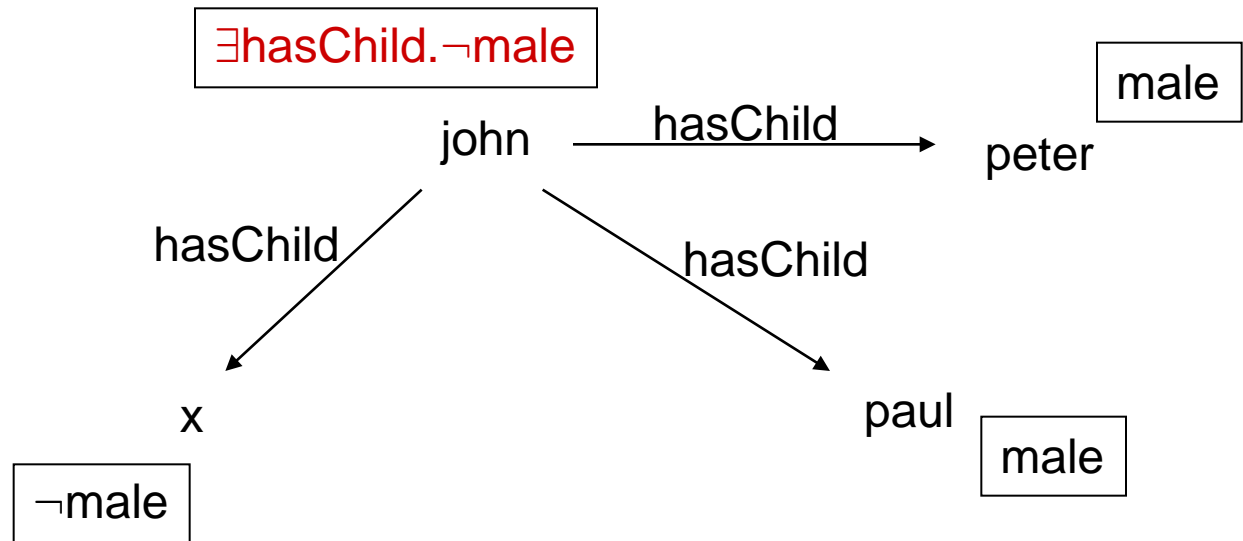
$\text{hasChild}(\text{john}, \text{paul})$

$\text{male}(\text{peter})$

$\text{male}(\text{paul})$

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

$$\neg \forall \text{hasChild.male} \equiv \exists \text{hasChild.}\neg \text{male}$$



# Example (3)

Show that the knowledge base

$\text{Bird} \sqsubseteq \text{Flies}$

$\text{Penguin} \sqsubseteq \text{Bird}$

$\text{Penguin} \sqcap \text{Flies} \sqsubseteq \perp$

$\text{Penguin}(\text{tweety})$

is unsatisfiable.

TBox:

$\neg B \sqcup F$

$\neg P \sqcup B$

$\neg P \sqcup \neg F \sqcup \perp$

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

$C(a)$

$C(c)$

$R(a,b)$

$R(a,c)$

$S(a,a)$

$S(c,b)$

$C \sqsubseteq \forall S.A$

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

$A \sqsubseteq \exists R.C$

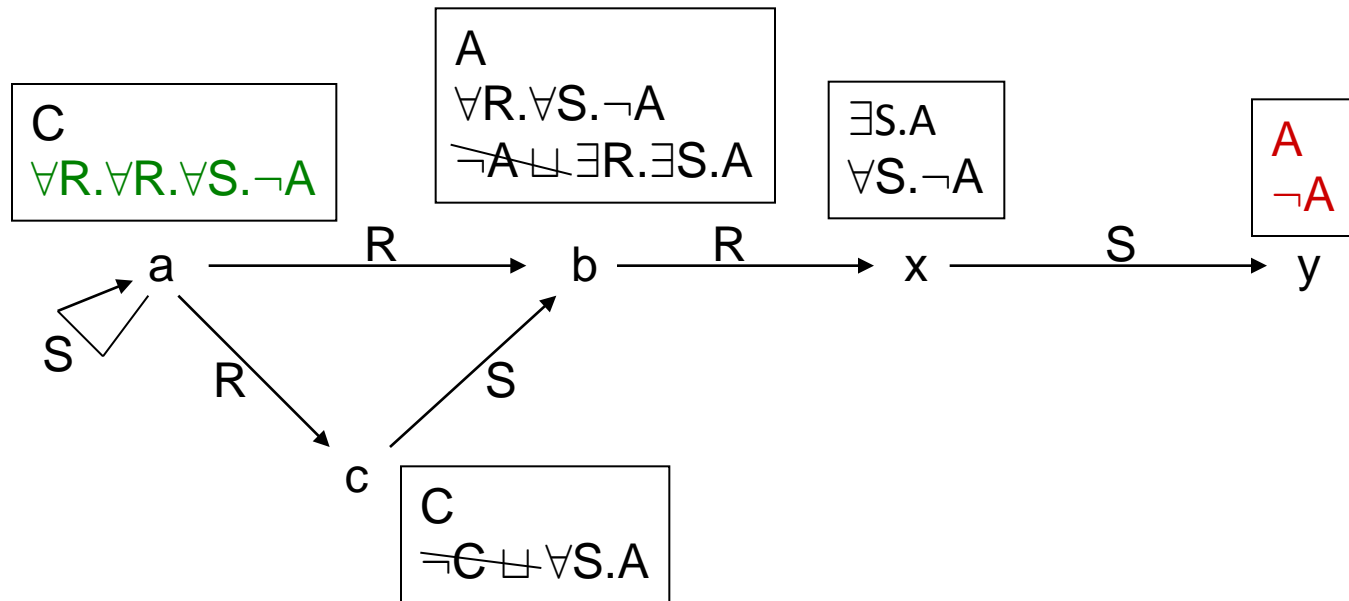
entails  $\exists R.\exists R.\exists S.A(a)$ .

# Example (4)

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

TBox:

- $\neg C \sqcup \forall S. A$
- $\neg A \sqcup \exists R. \exists S. A$
- $\neg A \sqcup \exists R. C$



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

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

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.**

$$\text{NNF}(C) = C \quad \text{if } C \text{ is a class name}$$

$$\text{NNF}(\neg C) = \neg C \quad \text{if } C \text{ is a class name}$$

$$\text{NNF}(\neg\neg C) = \text{NNF}(C)$$

$$\text{NNF}(C \sqcup D) = \text{NNF}(C) \sqcup \text{NNF}(D)$$

$$\text{NNF}(C \sqcap D) = \text{NNF}(C) \sqcap \text{NNF}(D)$$

$$\text{NNF}(\neg(C \sqcup D)) = \text{NNF}(\neg C) \sqcap \text{NNF}(\neg D)$$

$$\text{NNF}(\neg(C \sqcap D)) = \text{NNF}(\neg C) \sqcup \text{NNF}(\neg D)$$

$$\text{NNF}(\forall R.C) = \forall R.\text{NNF}(C)$$

$$\text{NNF}(\exists R.C) = \exists R.\text{NNF}(C)$$

$$\text{NNF}(\neg\forall R.C) = \exists R.\text{NNF}(\neg C)$$

$$\text{NNF}(\neg\exists R.C) = \forall R.\text{NNF}(\neg C)$$

$$\text{NNF}(\leq n R.C) = \leq n R.\text{NNF}(C)$$

$$\text{NNF}(\geq n R.C) = \geq n R.\text{NNF}(C)$$

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

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

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

- 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  $\langle x,y \rangle$  are labeled
    - either with sets  $L(\langle x,y \rangle)$  of role names or inverse role names
    - or with the symbol  $=$  (for equality)
    - or with the symbol  $\neq$  (for inequality)

- 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  $\neq$ , between  $a$  and  $b$  if  $a \neq b$  is in the ABox.**
- **There are no  $=$  relations (yet).**

- We write  $S^{-}$  as  $S$ .
- If  $R \in L(\langle x, y \rangle)$  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 a *neighbor* of  $x$ .
- *Ancestor* is the transitive closure of *Predecessor*.

- $x$  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  $L(x') = L(y')$
    - $L(\langle x', x \rangle) = L(\langle y', y \rangle)$
  - or the following holds: [" $x$  is indirectly blocked"]
    - an ancestor of  $x$  is blocked or
    - $x$  is successor of some  $y$  with  $L(\langle y, x \rangle) = \emptyset$

- **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  $\perp$  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.**

**$\sqcap$ -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  $\{C, D\} \cap \mathcal{L}(x) = \emptyset$ , 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  $\mathcal{L}(x) \leftarrow C$ .

**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,  $\leq_n S.C \in \mathcal{L}(x)$  or  $\geq_n S.C \in \mathcal{L}(x)$ , and there is an  $S$ -neighbor  $y$  of  $x$  with  $\{C, \text{NNF}(\neg C)\} \cap \mathcal{L}(y) = \emptyset$ , then set  $\mathcal{L}(y) \leftarrow C$  or  $\mathcal{L}(y) \leftarrow \text{NNF}(\neg C)$ .

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

1. create  $n$  new nodes with labels  $y_1, \dots, y_n$  (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  $i, j \in \{1, \dots, n\}$  with  $i \neq j$ .

**$\leq$ -rule:** If  $x$  is not indirectly blocked,  $\leq_n S.C \in \mathcal{L}(x)$ , 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  $\mathcal{L}(z, x) \leftarrow \{\text{Inv}(R) \mid R \in \mathcal{L}(x, y)\}$ ,
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  $\leq_n S.C \in \mathcal{L}(x)$ , 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

$\text{hasChild}(\text{john}, \text{peter})$

$\text{hasChild}(\text{john}, \text{paul})$

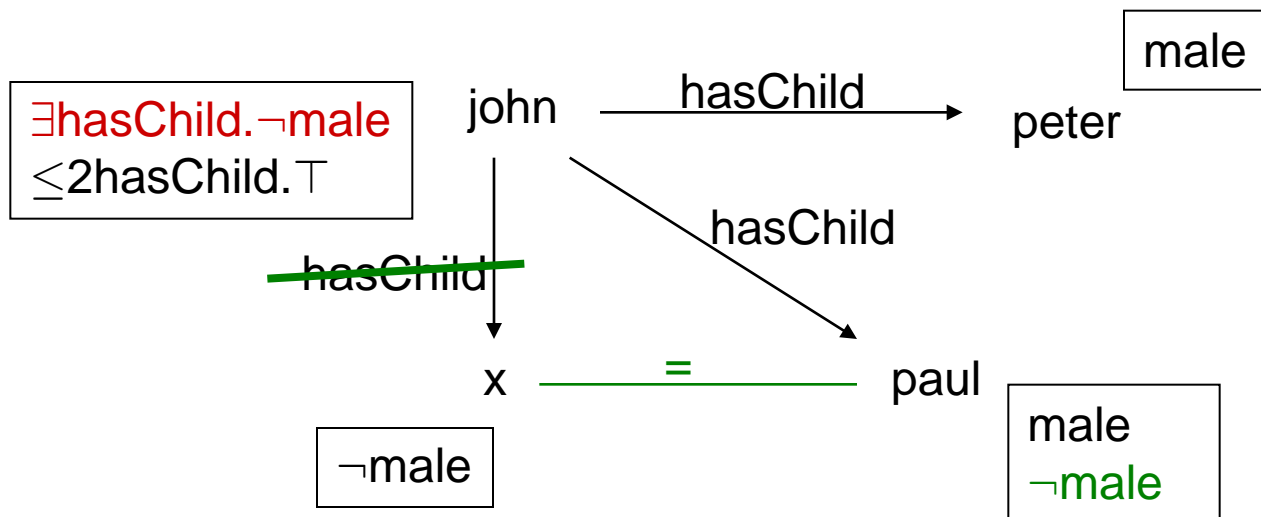
$\text{male}(\text{peter})$

$\text{male}(\text{paul})$

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

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

$$\neg \forall \text{hasChild}.\text{male} \equiv \exists \text{hasChild}.\neg \text{male}$$



# Example (1): cardinalities

Show, that

$\text{hasChild}(\text{john}, \text{peter})$

$\text{hasChild}(\text{john}, \text{paul})$

$\text{male}(\text{peter})$

$\text{male}(\text{paul})$

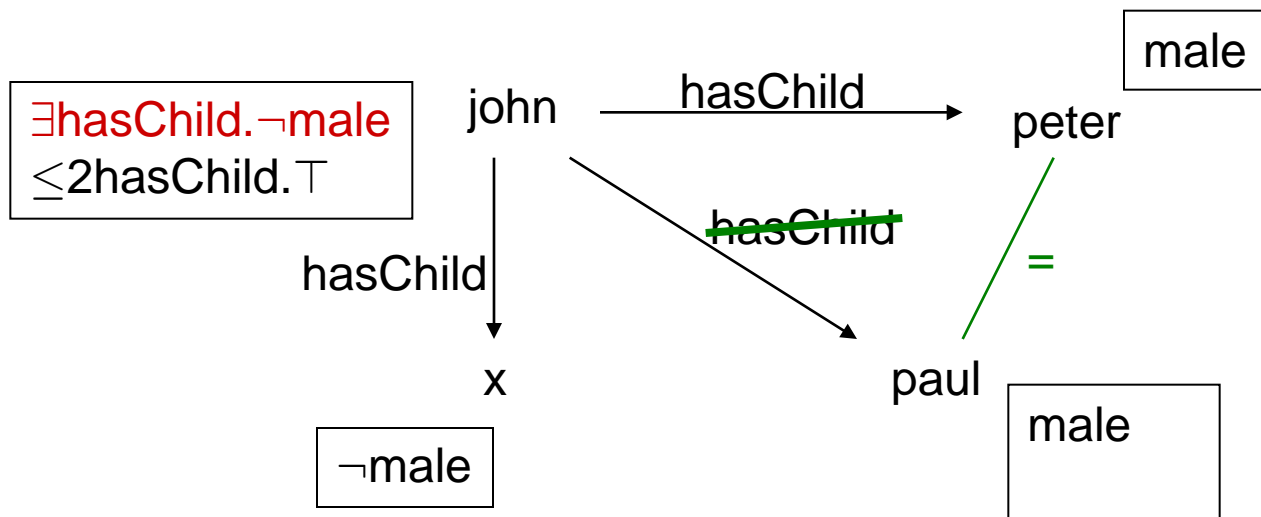
$\leq 2 \text{hasChild}.\top(\text{john})$

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

$$\neg \forall \text{hasChild}.\text{male} \equiv \exists \text{hasChild}.\neg \text{male}$$

backtracking!

now apply  $\leq$



# Example (1): cardinalities – again

Show, that

$\text{hasChild}(\text{john}, \text{peter})$

$\text{hasChild}(\text{john}, \text{paul})$

$\text{male}(\text{peter})$

$\text{male}(\text{paul})$

$\leq 2\text{hasChild}.\top(\text{john})$  and **peter  $\neq$  paul**

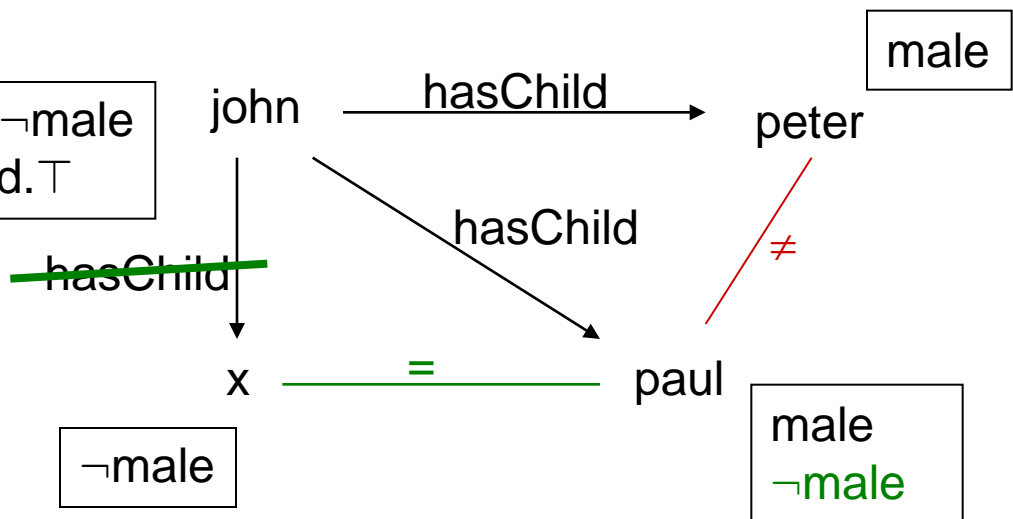
does *not* entail  $\forall \text{hasChild}.\text{male}(\text{john})$ .

$$\neg \forall \text{hasChild}.\text{male} \equiv \exists \text{hasChild}.\neg \text{male}$$

$$\begin{array}{l} \exists \text{hasChild}.\neg \text{male} \\ \leq 2\text{hasChild}.\top \end{array}$$

now apply  $\leq$

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



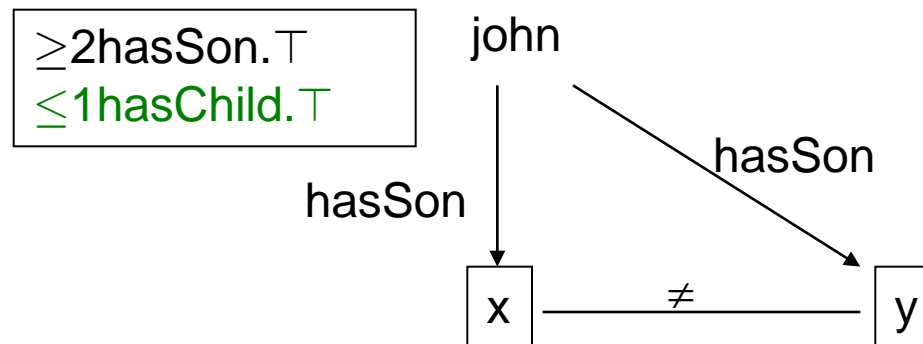
# Example (2): cardinalities

Show, that

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

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

$\text{hasSon} \sqsubseteq \text{hasChild}$



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

# Example (3): choose

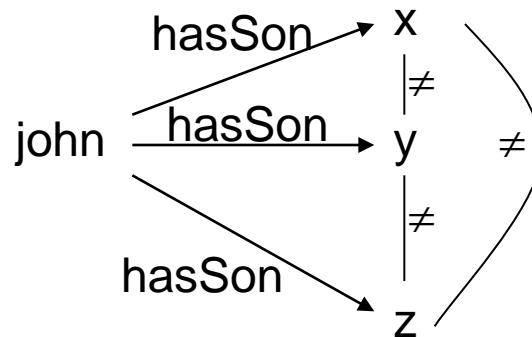
$\geq 3$ hasSon(john)

$\leq 2$ hasSon.male(john)

Is this contradictory?

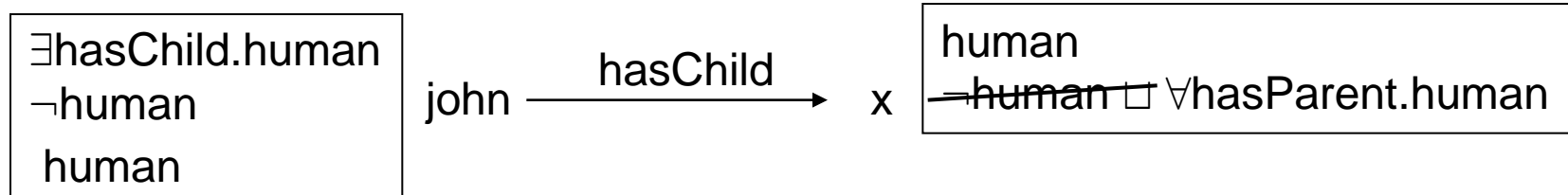
No, because the following tableau is complete.

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



# Example (4): inverse roles

$\exists \text{hasChild.human}(\text{john})$   
 $\text{human} \sqsubseteq \forall \text{hasParent.human}$   
 $\text{hasChild} \sqsubseteq \text{hasParent}^{-}$   
zu zeigen:  $\text{human}(\text{john})$



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

# Example (5): Transitivity and Blocking

**human  $\sqsubseteq \exists \text{hasFather}.\top$**

**human  $\sqsubseteq \forall \text{hasAncestor}.\text{human}$**

**hasFather  $\sqsubseteq \text{hasAncestor}$       **Trans(hasAncestor)****

**human(john)**

**Does this entail  $\leq 1 \text{hasFather}.\top(\text{john})$ ?**

**Negation:  $\geq 2 \text{hasFather}.\top(\text{john})$**

# Example (5): Transitivity and Blocking

$\text{human} \sqsubseteq \exists \text{hasFather}.\top$

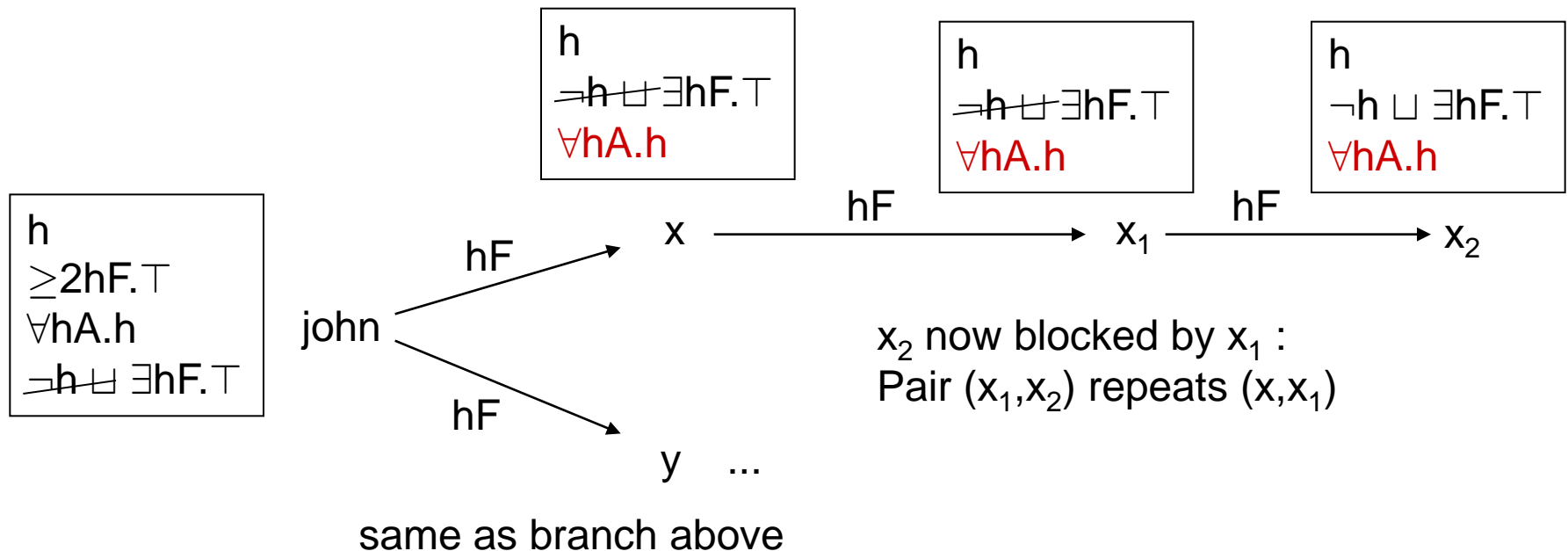
$\text{hasFather} \sqsubseteq \text{hasAncestor}$

$\forall \text{hasAncestor}.\text{human}(\text{john})$

$\text{human}(\text{john})$

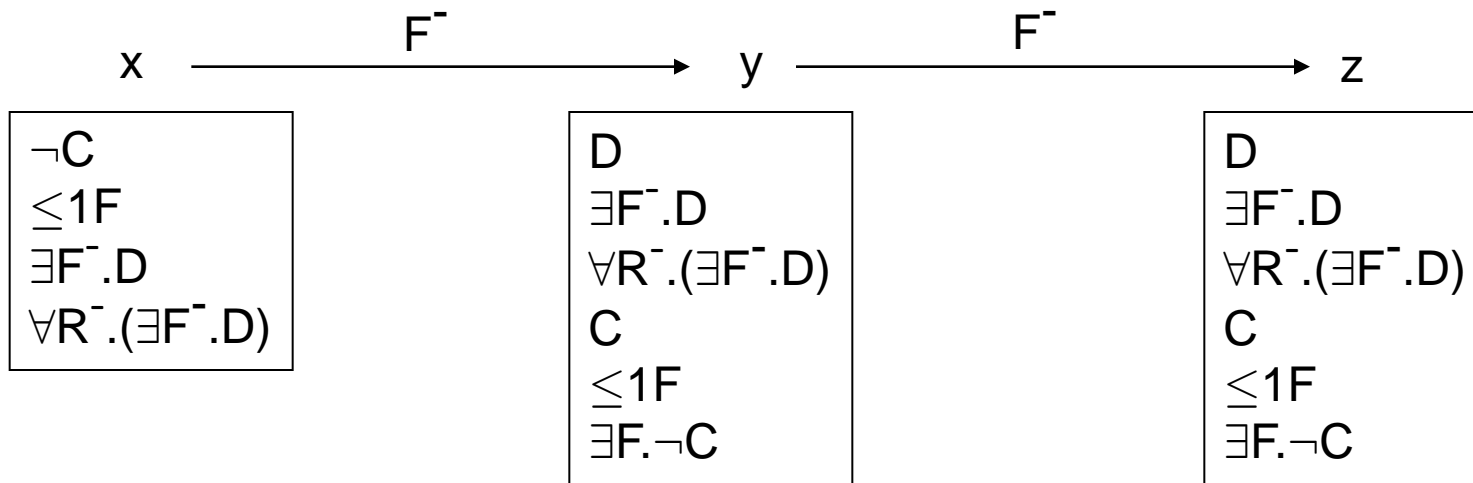
$\text{Trans}(\text{hasAncestor})$

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



# Example (6): Pairwise Blocking

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

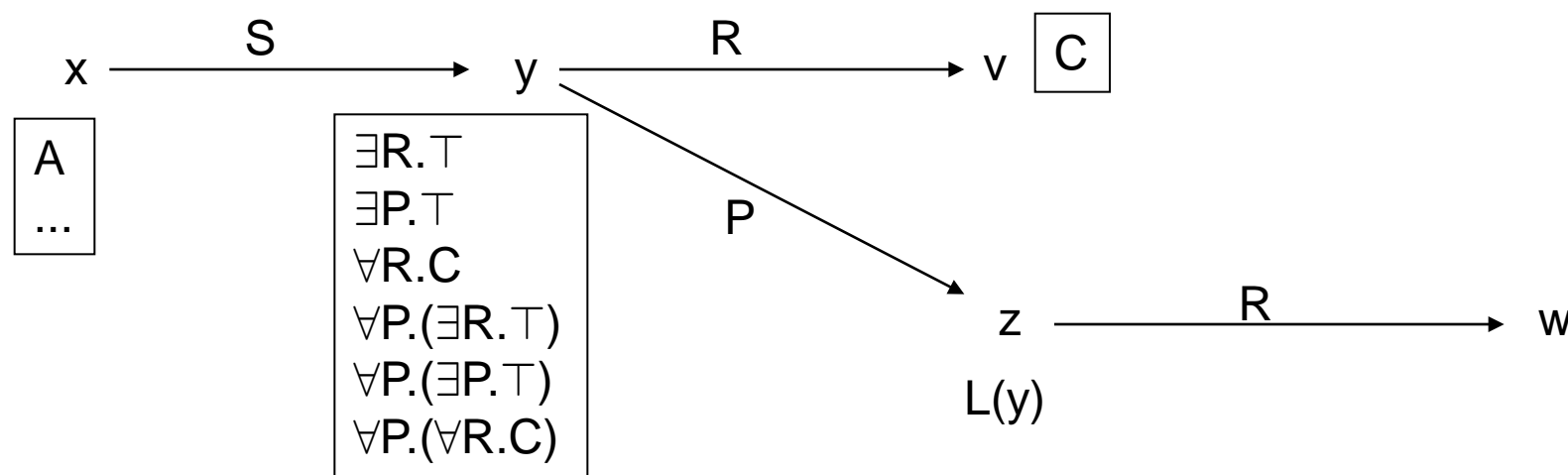


Without pairwise blocking,  $z$  would be blocked, which shouldn't happen:  
 Expansion of  $\exists F^-.C$  yields  $\neg C$  for node  $y$  as required.

# Example (7): Dynamic Blocking

$A \sqcap \exists S.(\exists R.T \sqcap \exists P.T \sqcap \forall R.C \sqcap \forall P.(\exists R.T) \sqcap \forall P.(\forall R.C) \sqcap \forall P.(\exists P.T))$   
 with  $C = \forall R.(\forall P.(\forall S.\neg A))$  and  $\text{Trans}(P)$ , 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.

- **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/>

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

## Reasoner:

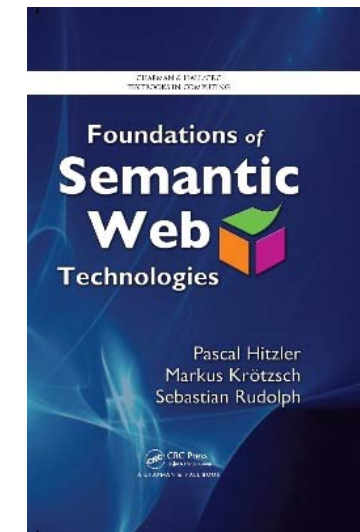
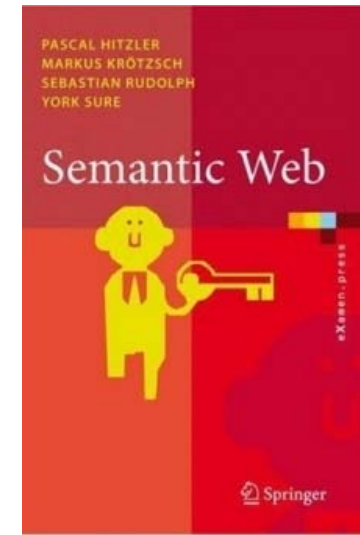
- **OWL 2 DL:**
  - Pellet <http://clarkparsia.com/pellet/>
  - Hermit <http://www.hermit-reasoner.com/>
- **OWL 2 EL:**
  - CEL <http://code.google.com/p/cel/>
- **OWL 2 RL:**
  - essentially any rule engine
- **OWL 2 QL:**
  - essentially any SQL engine (with a bit of query rewriting on top)

## Editors:

- Protégé
- NeOn Toolkit
- TopBraid Composer

- **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.**

- **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.)



- **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](http://www.semantic-web-book.org/page/ISWC2010_Tutorial)**

# OWL 2 and Rules

–

## Optional Part, If Enough Time

## 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.

- **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**
- 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

- **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]**

- 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

- **Essentially Horn-rules without function symbols**

**general form of the rules:**

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

**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**

- **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

- 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

- **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/>**

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

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

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

**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)**

**NutAllergic(sebastian)**

**NutProduct(peanutOil)**

**$\exists$ orderedDish.ThaiCurry(sebastian)**

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

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

orderedDish rdfs:range Dish.

**$\text{NutAllergic}(x) \wedge \text{NutProduct}(y) \rightarrow \text{dislikes}(x,y)$**

**$\text{dislikes}(x,z) \wedge \text{Dish}(y) \wedge \text{contains}(y,z) \rightarrow \text{dislikes}(x,y)$**

**$\text{orderedDish}(x,y) \wedge \text{dislikes}(x,y) \rightarrow \text{Unhappy}(x)$**

**Conclusions:**

**dislikes(sebastian,peanutOil)**

**orderedDish(sebastian, $y_s$ )**

**ThaiCurry( $y_s$ )**

**Dish( $y_s$ )**

**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<sub>s</sub>)**

**ThaiCurry(y<sub>s</sub>)**

**Dish(y<sub>s</sub>)**

**contains(y<sub>s</sub>,peanutOil)**

**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<sub>s</sub>)**

**ThaiCurry(y<sub>s</sub>)**

**Dish(y<sub>s</sub>)**

**contains(y<sub>s</sub>,peanutOil)**

**dislikes(sebastian,y<sub>s</sub>)**

**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<sub>s</sub>)**

**ThaiCurry(y<sub>s</sub>)**

**Dish(y<sub>s</sub>)**

**contains(y<sub>s</sub>,peanutOil)**

**dislikes(sebastian,y<sub>s</sub>)**

**Unhappy(sebastian)**

**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)**

- 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

- **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.

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

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

DL-safe {  
**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)**

**Unhappy(sebastian) *cannot* be concluded**

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

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

DL-safe {  
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<sub>s</sub>)

ThaiCurry(y<sub>s</sub>)

**Dish(y<sub>s</sub>)**

**contains(y<sub>s</sub>,peanutOil)**

~~dislikes(sebastian,y<sub>s</sub>)~~

- 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

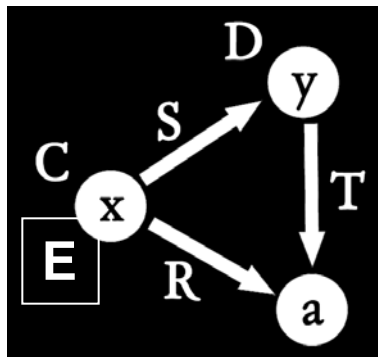
- **General idea:**  
Find out which rules can be encoded in OWL (2 DL) anyway
- **$\text{Man}(x) \wedge \text{hasBrother}(x,y) \wedge \text{hasChild}(y,z) \rightarrow \text{Uncle}(x)$** 
  - **$\text{Man} \sqcap \exists \text{hasBrother} . \exists \text{hasChild} . \top \sqsubseteq \text{Uncle}$**
- **$\text{ThaiCurry}(x) \rightarrow \exists \text{contains} . \text{FishProduct}(x)$** 
  - **$\text{ThaiCurry} \sqsubseteq \exists \text{contains} . \text{FishProduct}$**
- **$\text{kills}(x,x) \rightarrow \text{suicide}(x)$**                        **$\text{suicide}(x) \rightarrow \text{kills}(x,x)$** 
  - **$\exists \text{kills} . \text{Self} \sqsubseteq \text{suicide}$**                        **$\text{suicide} \sqsubseteq \exists \text{kills} . \text{Self}$**

**Note: with these two axioms,**

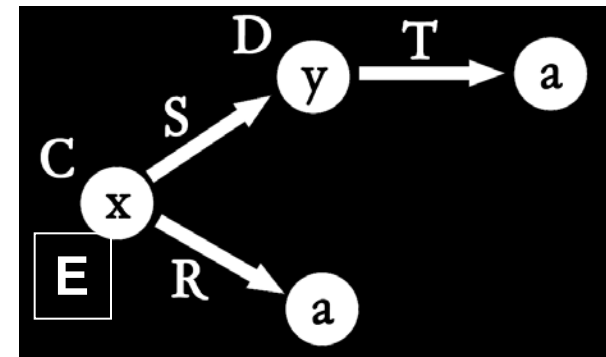
***suicide* is basically the same as *kills***

- **$\text{NutAllergic}(x) \wedge \text{NutProduct}(y) \rightarrow \text{dislikes}(x,y)$** 
  - **$\text{NutAllergic} \equiv \exists \text{nutAllergic}.\text{Self}$**
  - **$\text{NutProduct} \equiv \exists \text{nutProduct}.\text{Self}$**
  - **$\text{nutAllergic} \circ \text{U} \circ \text{nutProduct} \sqsubseteq \text{dislikes}$**
- **$\text{dislikes}(x,z) \wedge \text{Dish}(y) \wedge \text{contains}(y,z) \rightarrow \text{dislikes}(x,y)$** 
  - **$\text{Dish} \equiv \exists \text{dish}.\text{Self}$**
  - **$\text{dislikes} \circ \text{contains}^- \circ \text{dish} \sqsubseteq \text{dislikes}$**
- **$\text{worksAt}(x,y) \wedge \text{University}(y) \wedge \text{supervises}(x,z) \wedge \text{PhDStudent}(z) \rightarrow \text{professorOf}(x,z)$** 
  - **$\exists \text{worksAt}.\text{University} \equiv \exists \text{worksAt}.\text{University}.\text{Self}$**
  - **$\text{PhDStudent} \equiv \exists \text{phDStudent}.\text{Self}$**
  - **$\text{worksAt}.\text{University} \circ \text{supervises} \circ \text{phDStudent} \sqsubseteq \text{professorOf}$**

- 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

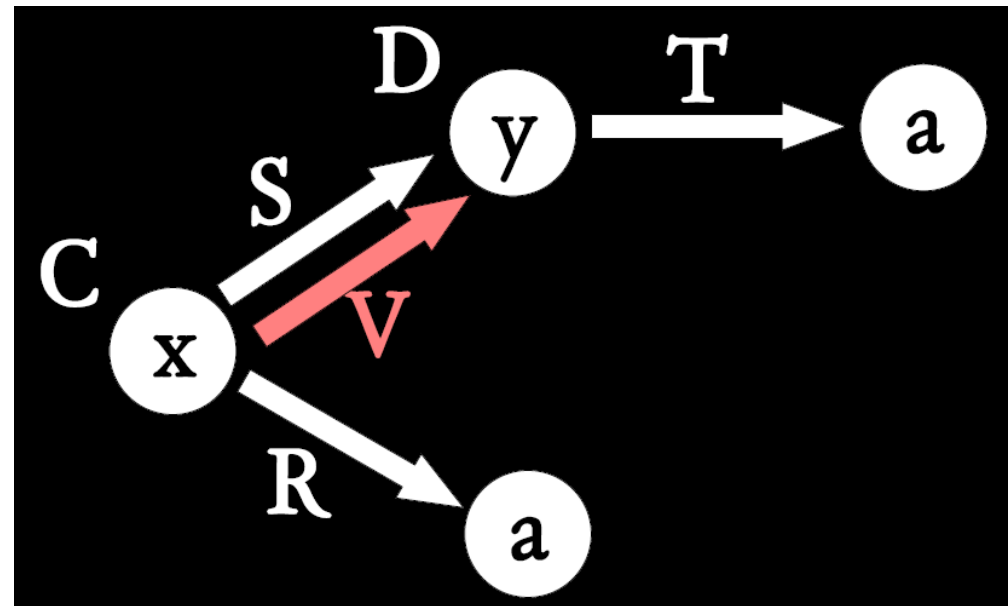


- Tree-shaped bodies
- First argument of the conclusion is the root
- $C(x) \wedge R(x,a) \wedge S(x,y) \wedge D(y) \wedge T(y,a) \rightarrow V(x,y)$

$C \sqcap \exists R.\{a\} \sqsubseteq \exists R1.Self$

$D \sqcap \exists T.\{a\} \sqsubseteq \exists R2.Self$

$R1 \circ S \circ R2 \sqsubseteq V$



- Tree-shaped bodies
- First argument of the conclusion is the root
- complex classes are allowed in the rules
  - $\text{Mouse}(x) \wedge \exists \text{hasNose.TrunkLike}(y) \rightarrow \text{smallerThan}(x,y)$
  - $\text{ThaiCurry}(x) \rightarrow \exists \text{contains.FishProduct}(x)$

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

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

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  $\mathcal{D}$ >

- 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

- **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.**

**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!**

- 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.
- $\text{worksAt}(x,y) \wedge \text{University}(y) \wedge \text{supervises}(x,z) \wedge \text{PhDStudent}(z) \rightarrow \text{professorOf}(x,z)$ 
  - $\exists \text{worksAt}.\text{University}(x) \wedge \text{supervises}(x,z) \wedge \text{PhDStudent}(z) \rightarrow \text{professorOf}(x,z)$
- $C(x) \wedge R(x,a) \wedge S(x,y) \wedge D(y) \wedge T(y,a) \rightarrow V(x,y)$ 
  - $(C \sqcap \exists R.\{a\})(x) \wedge S(x,y) \wedge (D \sqcap \exists T.\{a\})(y) \rightarrow V(x,y)$

- 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

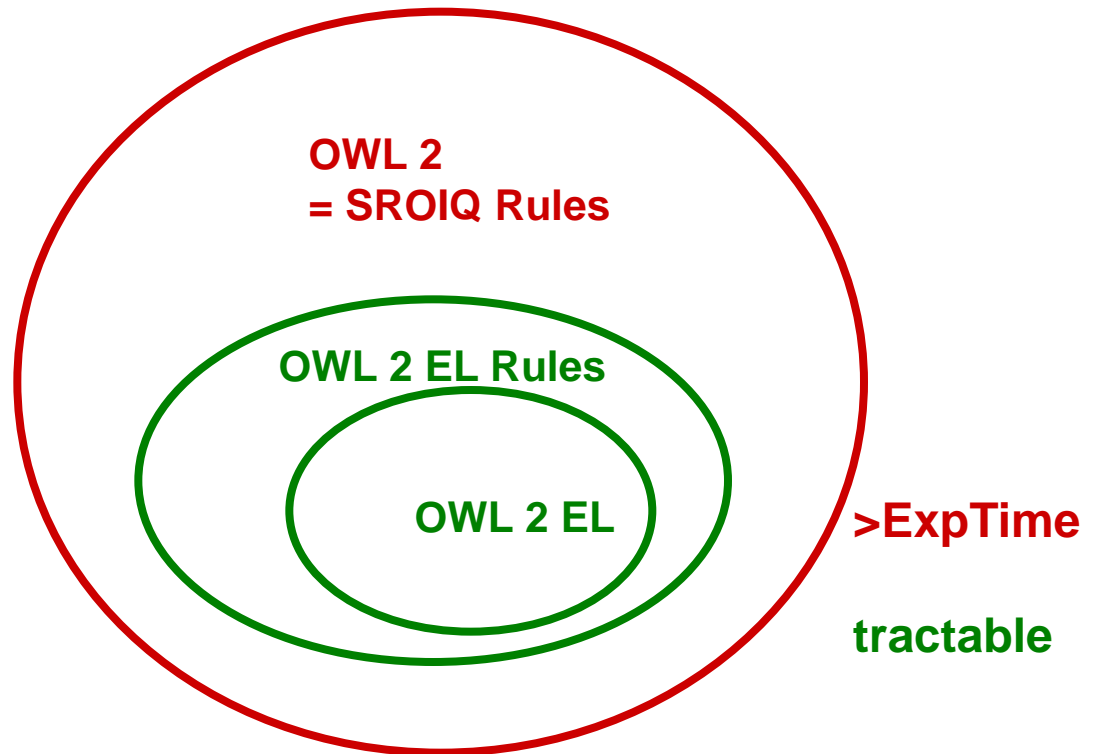
- The rules hidden in OWL 2: SROIQ Rules
- **Retaining tractability I: OWL 2 EL Rules**
- Retaining tractability II: DLP 2

Rules  
inside OWL

- Retaining tractability III: ELP

putting it  
all together

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



- 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  $A(x)$ ,  $\exists R.A(x)$ ,  $R(x,y)$ , where  $A$  is an atomic class or a nominal or  $\top$  or  $\perp$ .
- Translation is polynomial.
- $\exists \text{worksAt.University}(x) \wedge \text{supervises}(x,z) \wedge \text{PhDStudent}(z) \rightarrow \text{professorOf}(x,z)$ 
  - $\text{worksAt}(x,y) \wedge \text{University}(y) \wedge \text{supervises}(x,z) \wedge \text{PhDStudent}(z) \rightarrow \text{professorOf}(x,z)$
- $\text{ThaiCurry}(x) \rightarrow \exists \text{contains.FishProduct}(x)$

**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)**

- 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

- **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 ...**

- 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

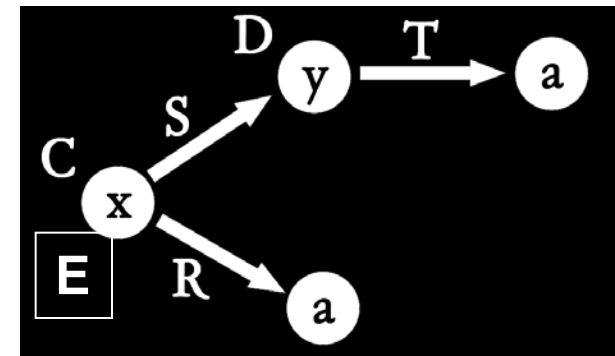
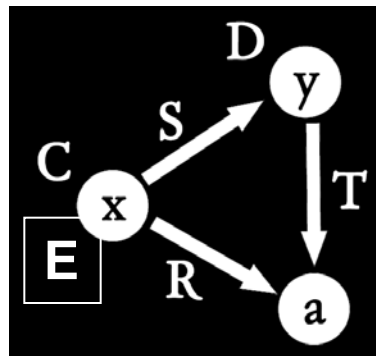
- **Retaining tractability III: ELP**

putting it  
all together

## 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.**

- 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.



- **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.**

- **$\text{orderedDish}(x,y) \wedge \text{dislikes}(x,y) \rightarrow \text{Unhappy}(x)$**
- **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.

- **ELP<sub>n</sub> 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. (!)**

**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**

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

$\text{Dish} \equiv \exists \text{dish}.\text{Self}$

$\text{dislikes} \text{ o contains}^{-1} \text{ o dish} \sqsubseteq \text{dislikes}$

but we don't have inverse roles in ELP!

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

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

this is fine 😊

**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<sub>s</sub>)**

**ThaiCurry(y<sub>s</sub>)**

**Dish(y<sub>s</sub>)**

**contains(y<sub>s</sub>,peanutOil)**

**dislikes(sebastian,y<sub>s</sub>)**

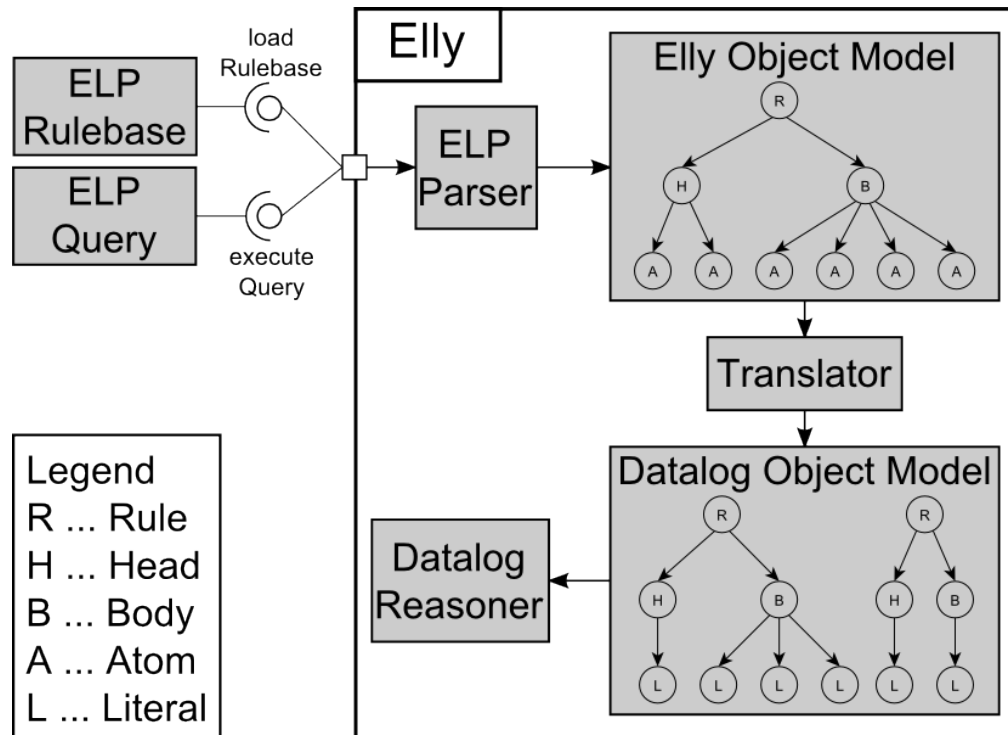
**NutAllergic(sebastian)**  
**NutProduct(peanutOil)**  
 **$\exists$ orderedDish.ThaiCurry(sebastian)**

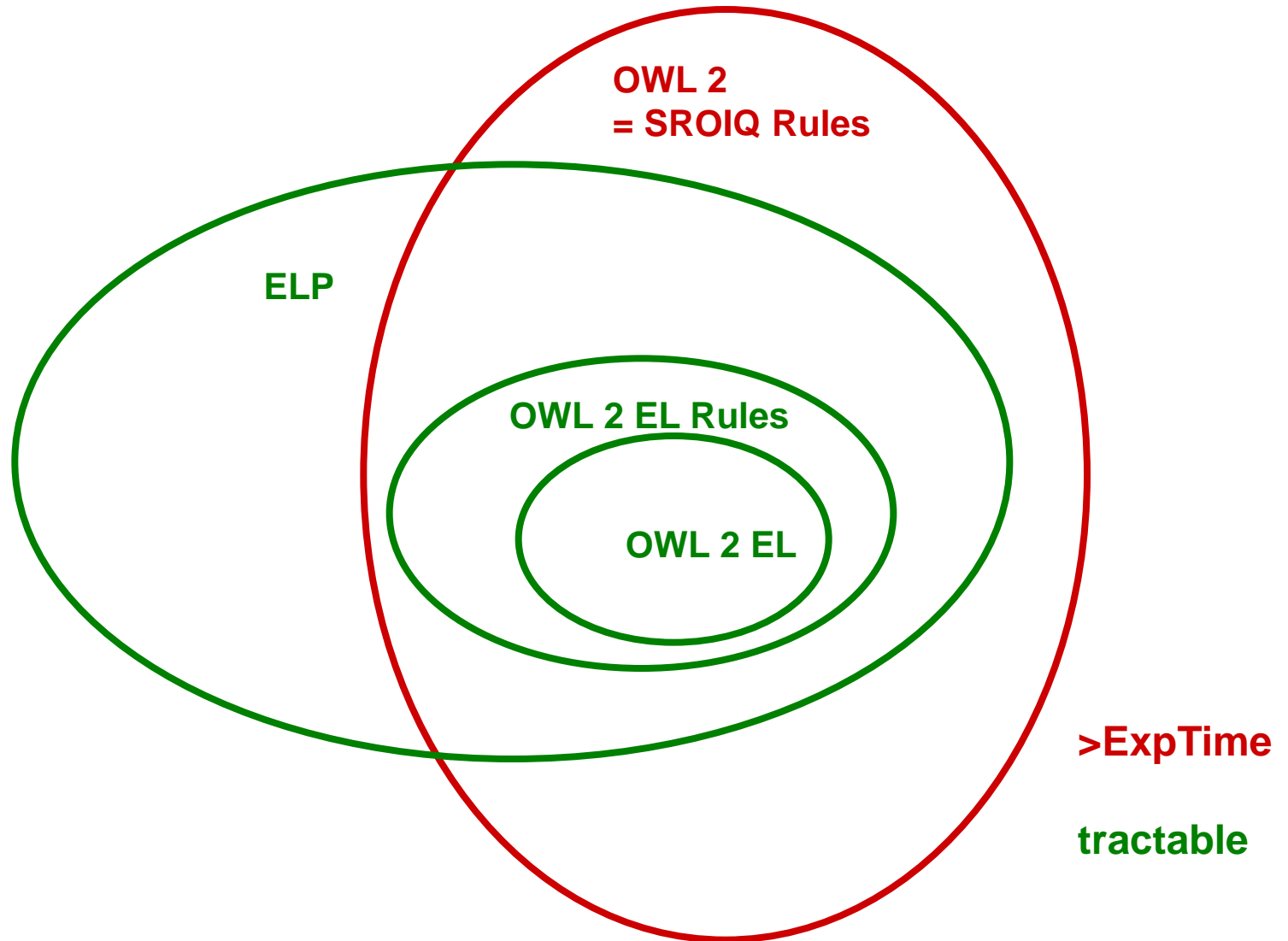
**ThaiCurry  $\sqsubseteq$   $\exists$ contains.{peanutOil}**  
 **$\top \sqsubseteq \forall$ orderedDish.Dish**

**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)**

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

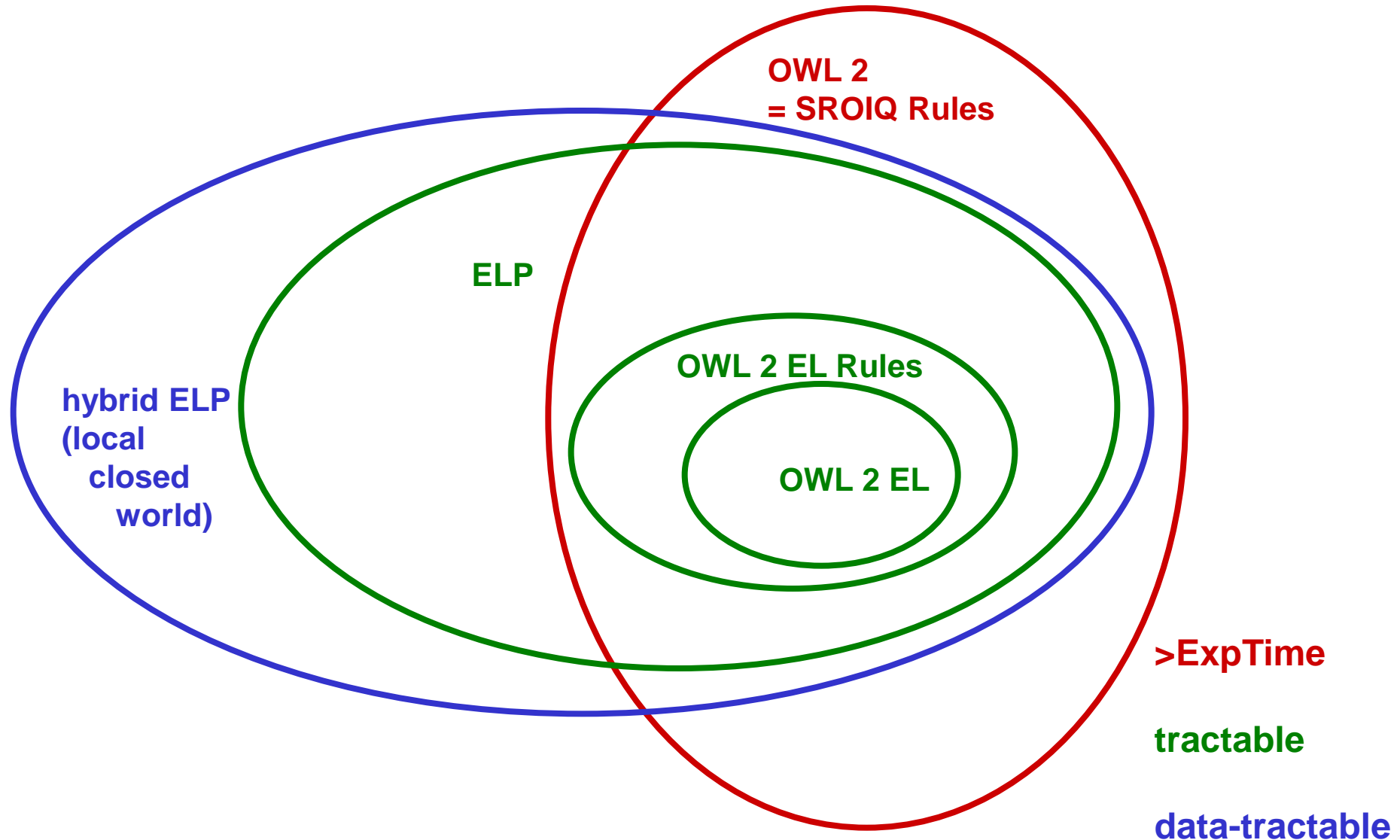




>ExpTime

tractable

- **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.



**Thanks!**

**[http://www.semantic-web-book.org/page/ISWC2010\\_Tutorial](http://www.semantic-web-book.org/page/ISWC2010_Tutorial)**

- 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.

- 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.

- **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.)**

