OWL 2 Rules

Pascal Hitzler

Kno.e.sis Center, Wright State University, Dayton, OH

Markus Krötzsch

Sebastian Rudolph

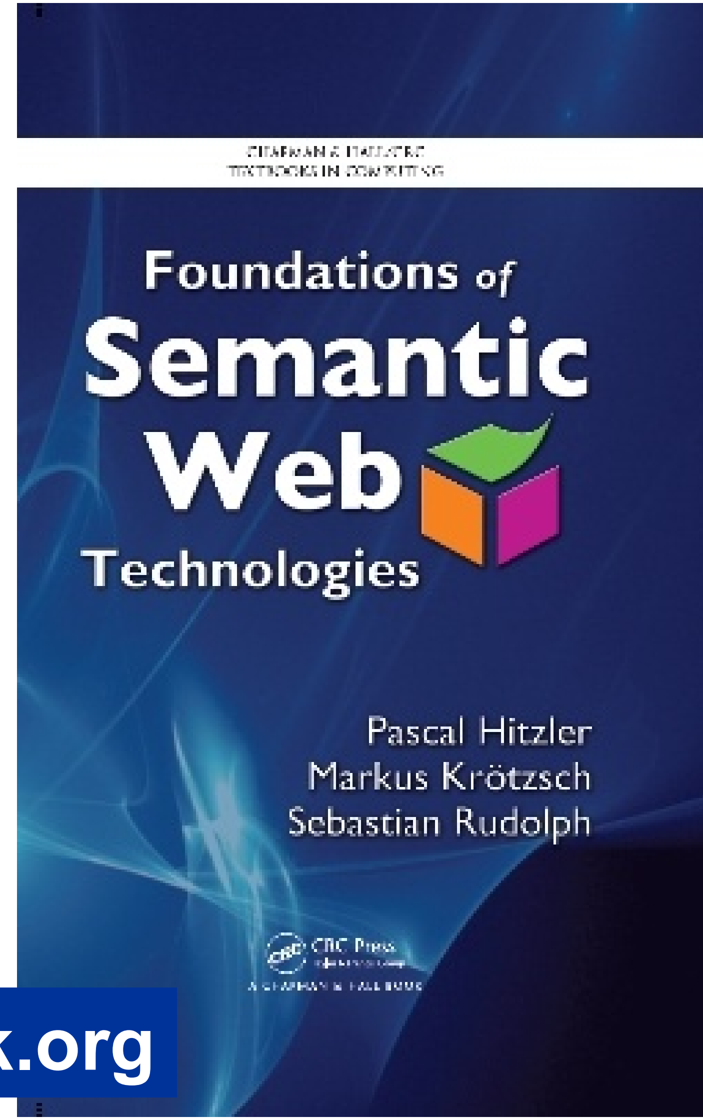
AIFB, University of Karlsruhe, Germany

Our Book

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph

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

Grab a flyer!



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

Slides

Available from

http://www.semantic-web-book.org/page/GeoS2009_Tutorial

Overall Outline

Part 1:

- **OWL 2 – An Introduction from a DL Point of View (ca. 60min)**

Part 2:

- **OWL 2 and Rules – Not as Incompatible as You May Think (ca. 60min)**

Part 1

OWL 2

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

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

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

DL syntax	FOL syntax
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))

ABox statements

TBox statements

DL syntax	RDFS 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 $\equiv$ Woman $\sqcap$ Parent**
  - :Mother owl:equivalentClass _:x .
    _:x rdf:type owl:Class .
    _:x owl:intersectionOf (:Woman :Parent) .

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

- **disjunction**
  - **Parent $\equiv$ Mother $\sqcup$ Father**
  - :Parent owl:equivalentClass _:x .
    _:x rdf:type owl:Class .
    _:x owl:unionOf (:Mother :Father) .

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

- **negation**
  - **ChildlessPerson $\equiv$ Person $\sqcap$ $\neg$Parent**
  - :ChildlessPerson owl:equivalentClass _:x .
    _:x rdf:type owl:Class .
    _:x owl:intersectionOf (:Person _:y) .
    _:y owl:complementOf :Parent .

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

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

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**

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

  - **includes top property and bottom property**

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

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

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
  - $hasParent \equiv hasChild^{-}$
  - $Orphan \equiv \forall hasChild^{-}.Dead$
- 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: 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 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:

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

    $R \circ S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R$ $\quad$ $S_1 \circ S_2 \circ \ldots \circ S_n \circ R \sqsubseteq R$
  - 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

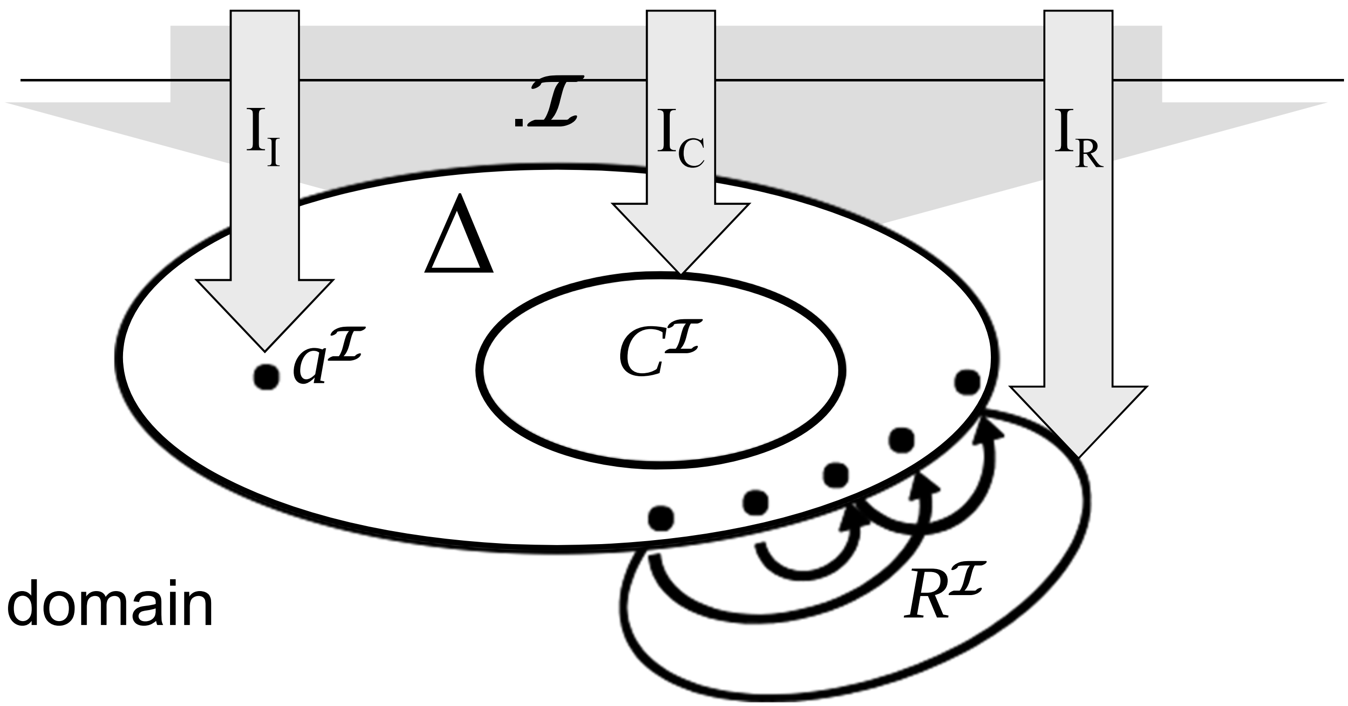
- combining property chain axioms and cardinality constraints may lead to undecidability
- restriction: use only *simple* properties in cardinality expressions (i.e. those which cannot be – directly or indirectly – inferred from property chains)
- technically:
  - for any property chain axiom $S_1 \circ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

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

OWL Direct Semantics

- model-theoretic semantics
- starts with interpretations
- an interpretation maps

individual names, class names and property names...



...into a domain

OWL Direct Semantics

- mapping is extended to complex class expressions:
  - $\top^I = \Delta^I$ $\quad\quad$ $\bot^I = \emptyset$
  - $(C \sqcap D)^I = C^I \cap D^I$ $\quad\quad$ $(C \sqcup D)^I = C^I \cup D^I$ $\quad\quad$ $(\neg C)^I = \Delta^I \setminus C^I$
  - $\forall R.C = \{ x \mid \forall (x,y) \in R^I \rightarrow y \in C^I \}$
    $\exists R.C = \{ x \mid \exists (x,y) \in R^I \wedge y \in C^I \}$
  - $\geq nR.C = \{ x \mid \#\{ y \mid (x,y) \in R^I \wedge y \in C^I \} \geq n \}$
  - $\leq nR.C = \{ x \mid \#\{ y \mid (x,y) \in R^I \wedge y \in C^I \} \leq n \}$
- ...and to role expressions:
  - $U^I = \Delta^I \times \Delta^I$ $\quad\quad$ $(R^-)^I = \{ (y,x) \mid (x,y) \in R^I \}$
- ...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 $R^I \subseteq S^I$
  - $Dis(R,S)$ 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 via FOL

- but often OWL 2 DL is said to be a fragment of FOL (with equality)...
- yes, there is a translation of OWL 2 DL into FOL

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

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

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

Contents

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: Human $\sqsubseteq$ $\exists$hasParent.Person
  $\exists$married.$\top$ $\sqcap$ CatholicPriest $\sqsubseteq$ $\bot$;
  hasParent $\circ$ hasParent $\sqsubseteq$ 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:

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

Contents

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

Proof Theory

- Traditionally using tableaux algorithms (see below)

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

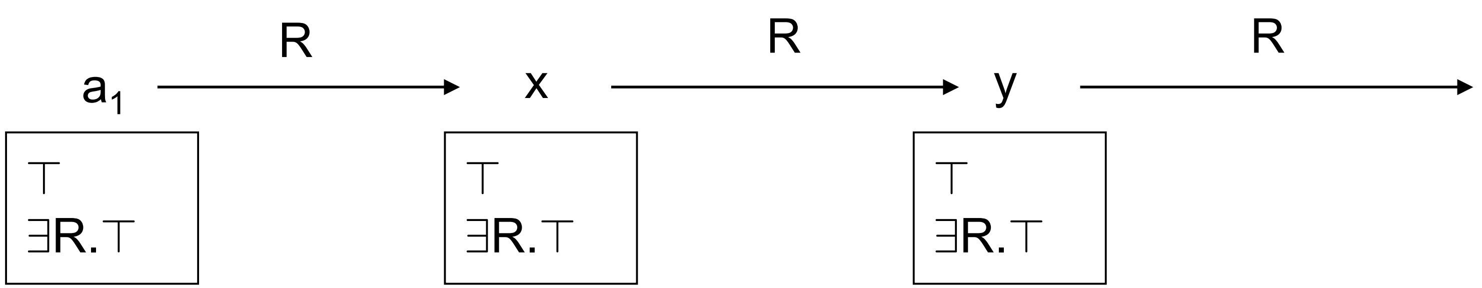
- **Adaptation of FOL tableaux algorithms.**
- **Problem: OWL is decidable, but FOL tableaux algorithms do not guarantee termination.**
- **Solution: *blocking*.**

DL Tableaux Termination Problem

TBox: $\exists R.\top$

ABox: $\top(a_1)$

- **Is satisfiable:**
  Model M contains elements $a_1^M, a_2^M, \ldots$
  and $R^M(a_i^M, a_{i+1}^M)$ for all $i \geq 1$.
- **But naive tableau does not terminate!**

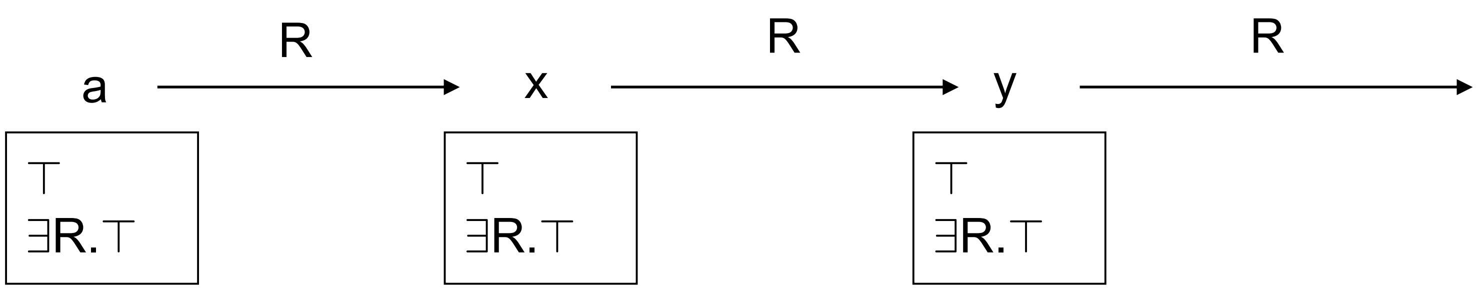


DL Tableaux Termination Problem

Nothing essentially new happens.

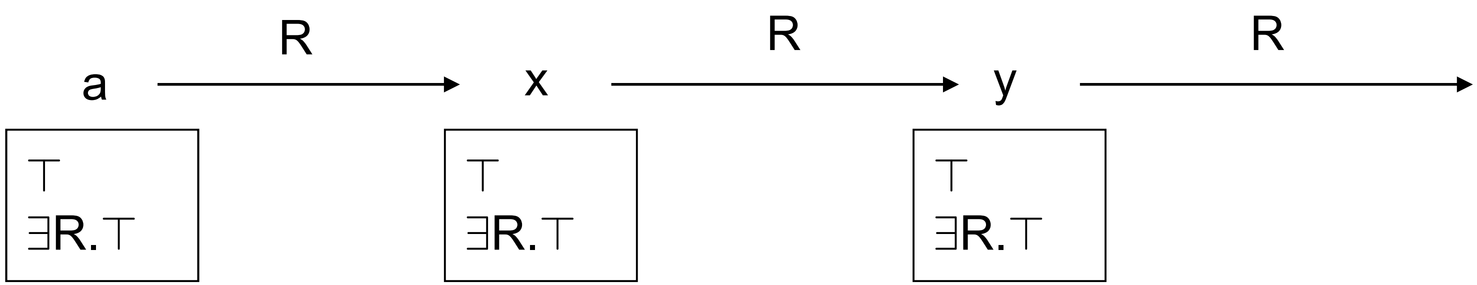
Idea: y does not need to be expanded, because it is basically a copy of x.

⇒ Blocking



Blocking (in ALC)

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



y blocked by x in this example.

Blocking conditions for more expressive DLs are more involved; the idea is the same.

ALC Tableau example

Show that

C(a)	**C(c)**
R(a,b)	**R(a,c)**
S(a,a)	**S(c,b)**

$C \sqsubseteq \forall S.A$

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

$A \sqsubseteq \exists R.C$

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

ALC Tableau Example

TBox:

$\neg C \sqcup \forall S.A$

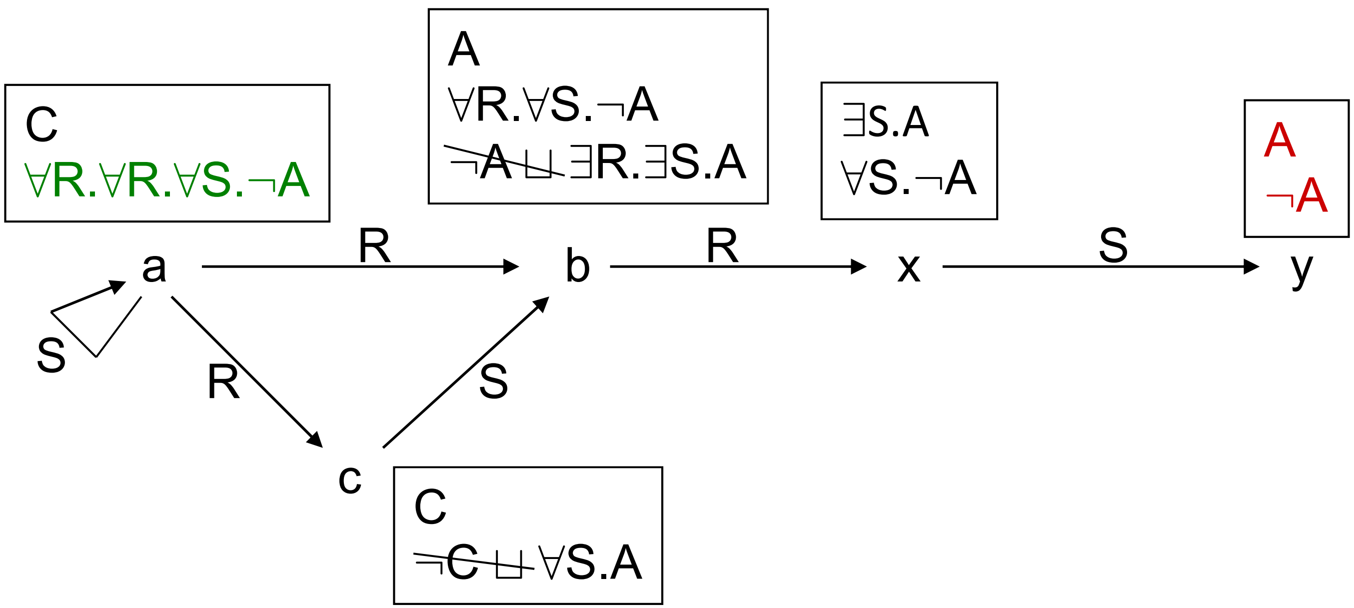
$\neg A \sqcup \exists R.\exists S.A$

$\neg A \sqcup \exists R.C$

ABox

C(a)	C(c)
R(a,b)	R(a,c)
S(a,a)	S(c,b)

$\neg\exists R.\exists R.\exists S.A(a)$ is $\forall R.\forall R.\forall S.\neg A(a)$



Contents

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

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

Part 2

OWL 2 and Rules

Main References Part 2

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:

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

  $\boxed{\text{body} \rightarrow \text{head}}$

  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

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)

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

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

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

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)

$\exists$orderedDish.ThaiCurry(sebastian)

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

$\top$ $\sqsubseteq$ $\forall$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)

$\text{NutAllergic}(\text{sebastian})$

$\text{NutProduct}(\text{peanutOil})$

$\exists\text{orderedDish.ThaiCurry}(\text{sebastian})$

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

$\top \sqsubseteq \forall\text{orderedDish.Dish}$

$\text{NutAllergic}(x) \land \text{NutProduct}(y) \to \text{dislikes}(x,y)$

$\text{dislikes}(x,z) \land \text{Dish}(y) \land \text{contains}(y,z) \to \text{dislikes}(x,y)$

$\text{orderedDish}(x,y) \land \text{dislikes}(x,y) \to \text{Unhappy}(x)$

Conclusions:

$\text{dislikes}(\text{sebastian},\text{peanutOil})$

$\text{orderedDish}(\text{sebastian},y_s)$

$\text{ThaiCurry}(y_s)$

$\text{Dish}(y_s)$

$\text{contains}(y_s,\text{peanutOil})$

SWRL example (running example)

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

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

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

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)

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

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

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

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

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

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

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

Unhappy(sebastian) can*not* be concluded

DL-safe SWRL example

NutAllergic(sebastian)
NutProduct(peanutOil)
∃orderedDish.ThaiCurry(sebastian)

ThaiCurry ⊑ ∃contains.{peanutOil}
⊤ ⊑ ∀orderedDish.Dish

DL-safe
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)~~

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

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

- Man(x) $\wedge$ hasBrother(x,y) $\wedge$ 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) suicide(x) $\rightarrow$ kills(x,x)
  - $\exists$kills.Self $\sqsubseteq$ suicide 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

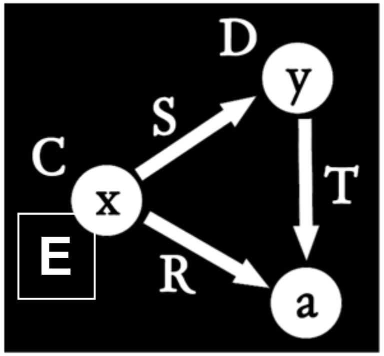
- dislikes(x,z) ∧ Dish(y) ∧ contains(y,z) → dislikes(x,y)
  - Dish ≡ ∃dish.Self
    dislikes o contains⁻ o dish ⊑ dislikes

- 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

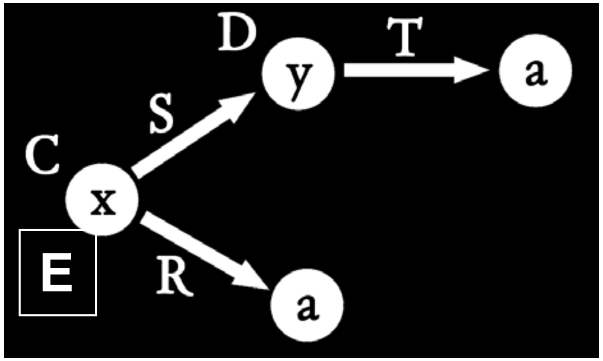
DL Rules: definition

- **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

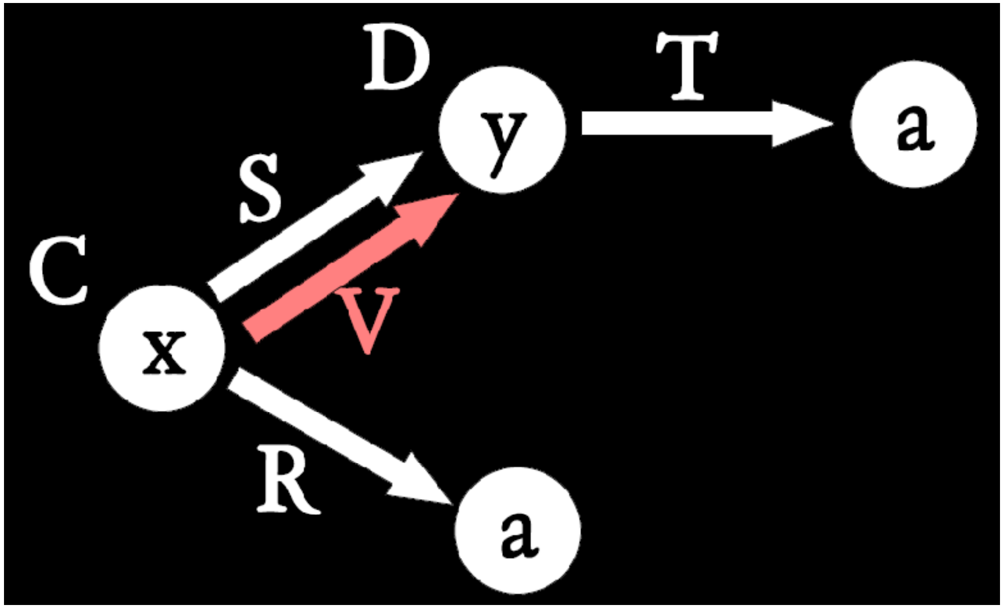


DL Rules: definition

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

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

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



DL Rules: definition

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

- complex classes are allowed in the rules
  - Mouse(x) $\wedge$ $\exists$hasNose.TrunkLike(y) $\rightarrow$ smallerThan(x,y)
  - ThaiCurry(x) $\rightarrow$ $\exists$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

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.

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

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

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

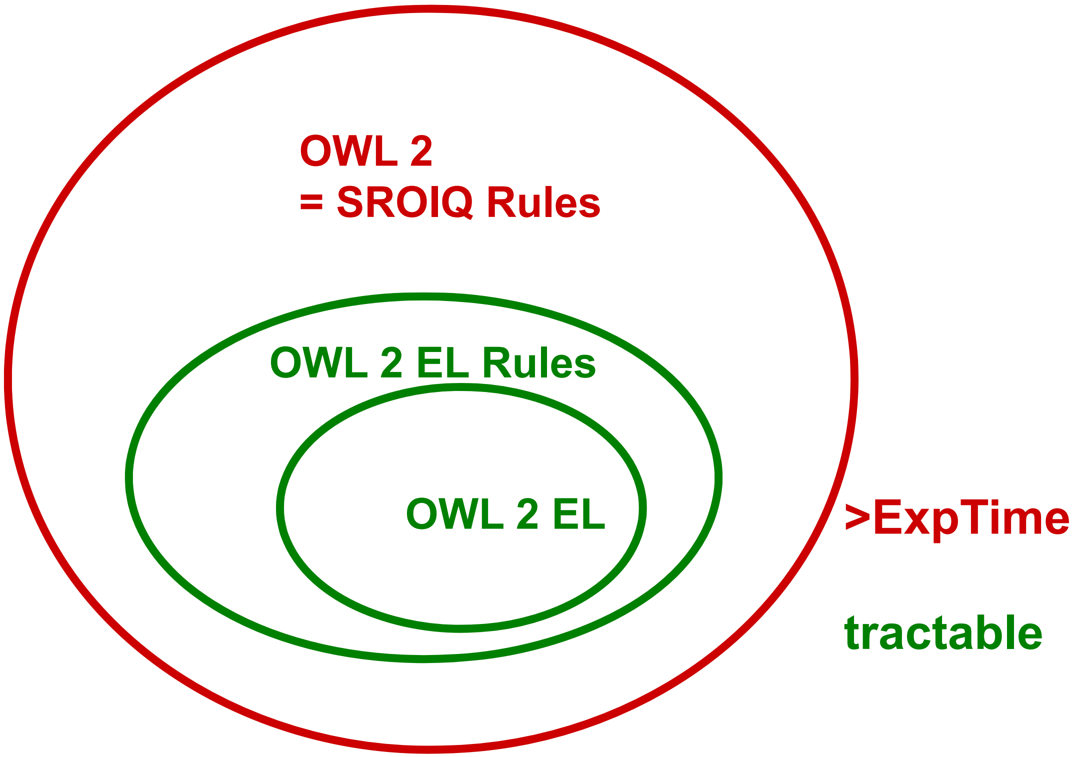
Rules inside OWL

- 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 $\top$ or $\bot$.

- 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 (aka putting it all together)

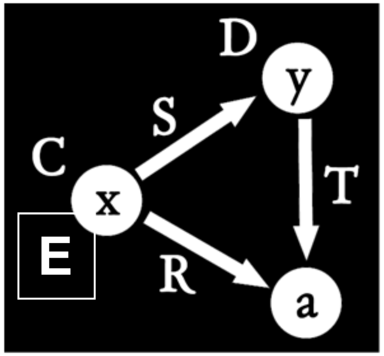
- ELP is
  - OWL 2 EL Rules +
  - a generalisation of DL-safety +
  - variable-restricted DL-safe Datalog +
  - role conjunctions (for simple roles).

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

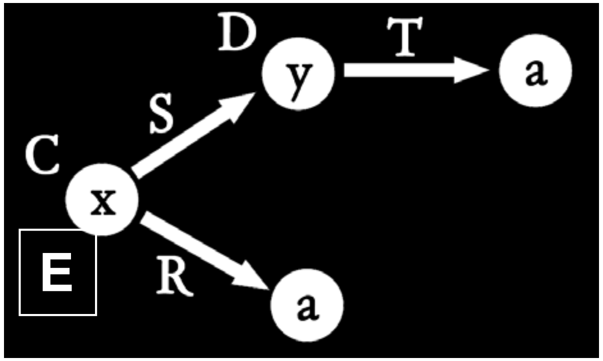
DL-safe variables

- 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 (aka putting it all together)

- 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

KNO.E.SIS

$$\text{NutAllergic}(\text{sebastian})$$

$$\text{NutProduct}(\text{peanutOil})$$

$$\exists\text{orderedDish.ThaiCurry}(\text{sebastian})$$

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

$$\top \sqsubseteq \forall\text{orderedDish.Dish}$$

[okay] $\text{NutAllergic}(x) \wedge \text{NutProduct}(y) \rightarrow \text{dislikes}(x,y)$

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

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

[okay – role conjunction]

not an EL++ rule

ELP example

- dislikes(x,z) ∧ Dish(y) ∧ contains(y,z) → dislikes(x,y)
  as SROIQ rule translates to

  Dish ≡ ∃dish.Self
  dislikes o contains⁻ o dish ⊑ dislikes

  but we don't have inverse roles in ELP!

- solution: make z a DL-safe variable:

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

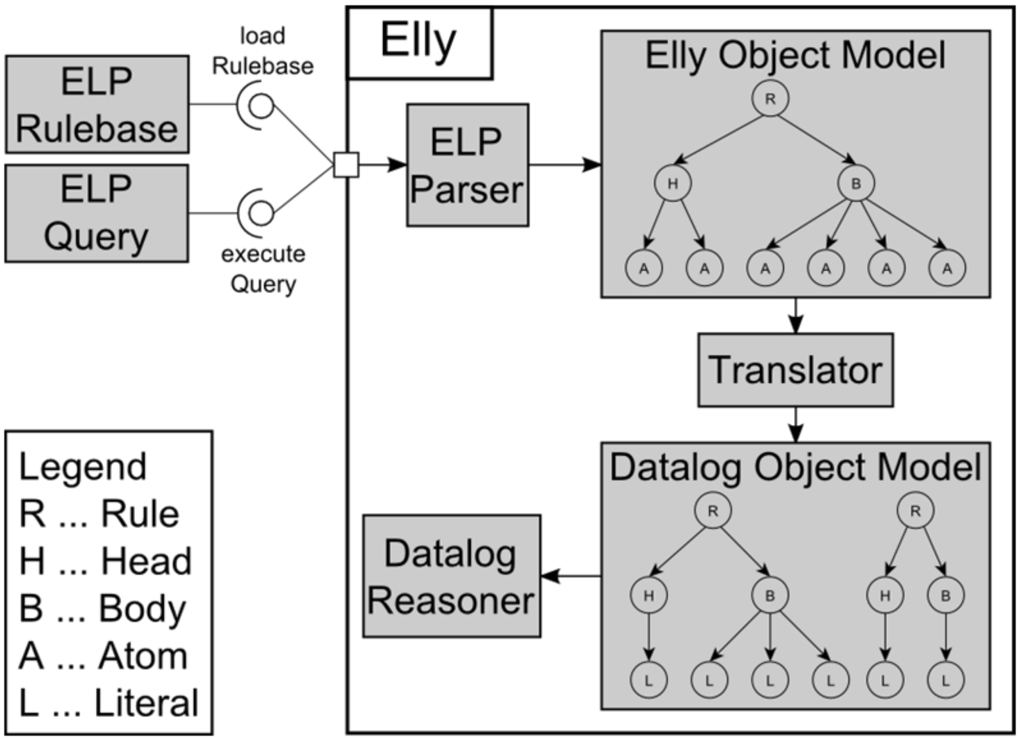
ThaiCurry $\sqsubseteq$ $\exists$contains.{peanutOil}
$\top$ $\sqsubseteq$ $\forall$orderedDish.Dish

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)

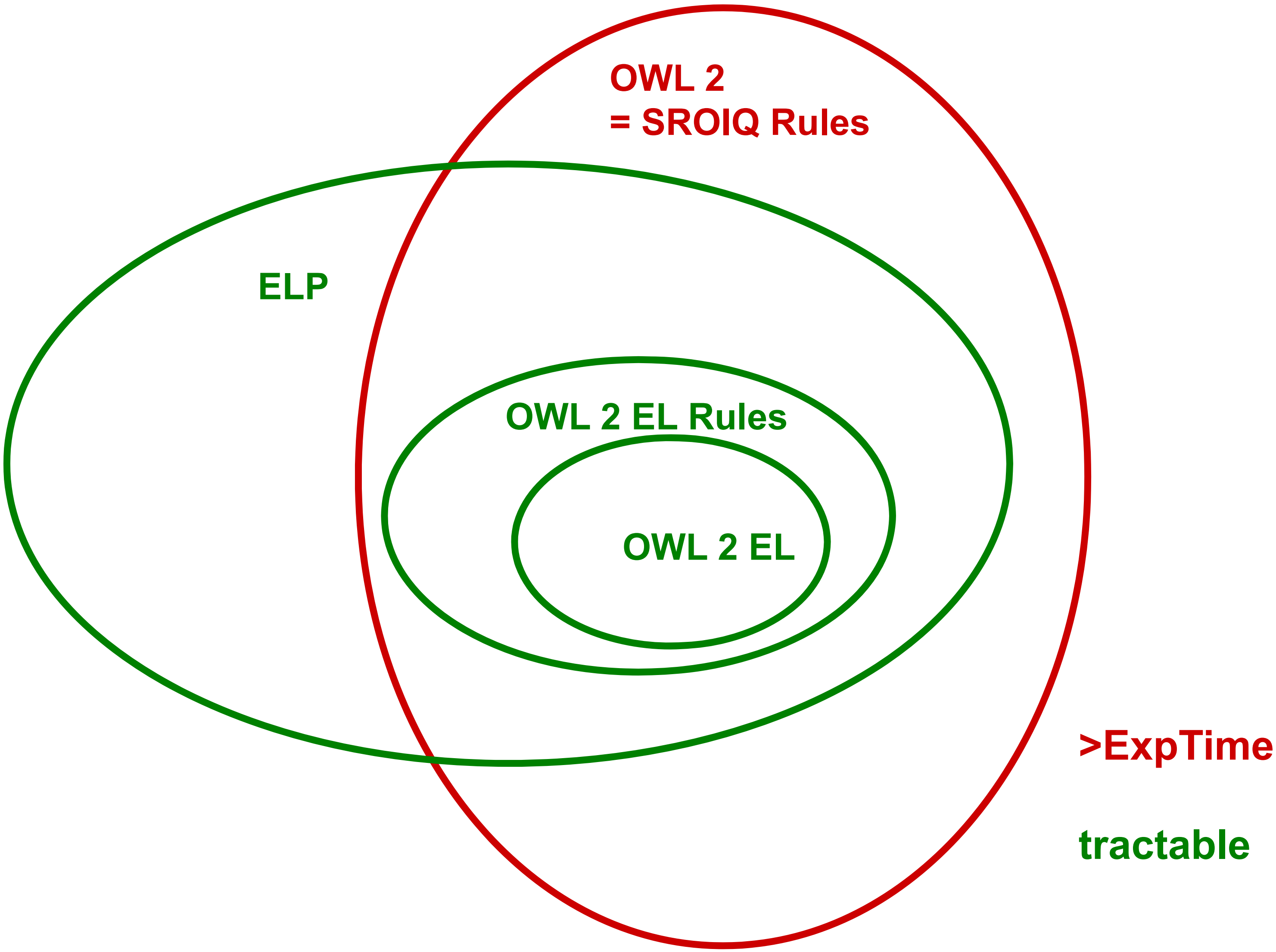
Conclusion: Unhappy(sebastian)

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



Thanks!

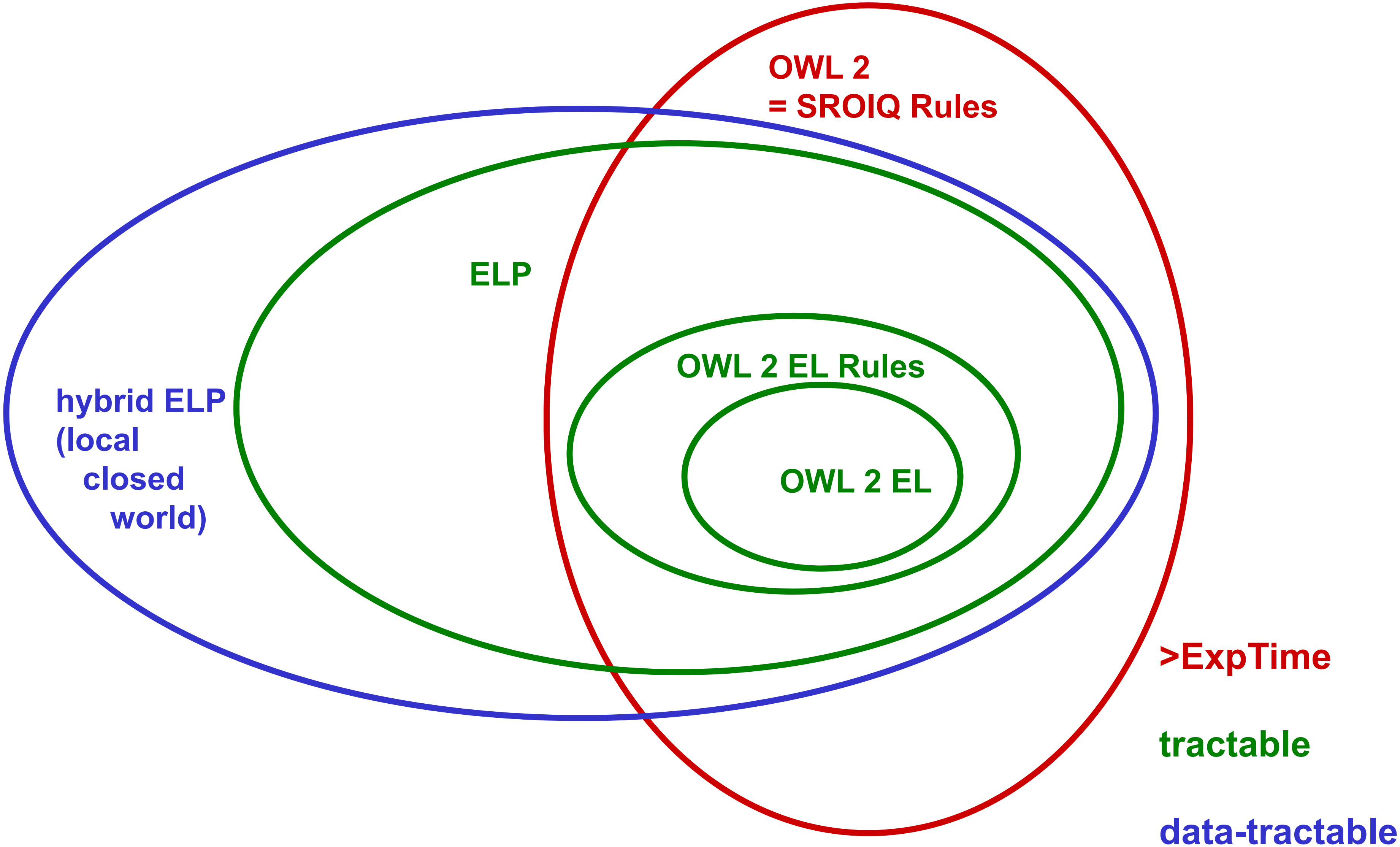
http://www.semantic-web-book.org/page/GeoS2009_Tutorial

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



References Part 2

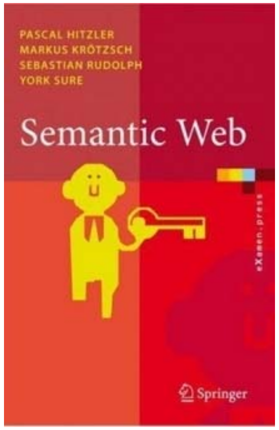
- Markus Krötzsch, Sebastian Rudolph, Pascal Hitzler, Description Logic Rules. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, Nikos Avouris, eds.: Proceedings of the 18th European Conference on Artificial Intelligence (ECAI-08), pp. 80–84. IOS Press 2008.
- Markus Krötzsch, Sebastian 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 Part 2

- 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.)