

## Semantic Web Technologies

### OWL DL and RDF Rules

Jos de Bruijn  
debruijn AT inf.unibz.it

KRDB Research Group  
Free University of Bolzano, Italy

21 November 2007

1/59

## Outline

OWL in RDF  
Mapping OWL DL to RDF

Reasoning with OWL  
RDF-based Rule Reasoning  
DL-based Reasoning

OWL DL and Logic Programming  
Description Logic Programs  
Beyond DLP

2/59

## OWL Lite/DL vs. RDF

- ▶ RDF Graph defined through translation from Abstract Syntax
- ▶ Example:

```
Class(Human partial Animal
      restriction(hasLegs cardinality(2))
      restriction(hasName allValuesFrom(xsd:string)))
```

Human	rdf:type	owl:Class
Human	rdfs:subClassOf	Animal
Human	rdfs:subClassOf	_:X1
_:X1	rdf:type	owl:Restriction
_:X1	owl:onProperty	hasLegs
_:X1	owl:cardinality	"2" 8sd:nonNegativeInteger
Human	rdfs:subClassOf	_:X2
_:X2	rdf:type	owl:Restriction
_:X2	owl:onProperty	hasName
_:X2	owl:allValuesFrom	xsd:string

4/59

## OWL Lite/DL vs. RDF

- ▶ Not every RDF graph is OWL Lite/DL ontology
- ▶ Example:
  - A** rdf:type **A**
- ▶ How to check whether an RDF graph **G** is OWL DL?
  1. Construct an OWL ontology **O** in Abstract Syntax
  2. Translate to RDF graph **G'**
  3. If **G=G'**, then **G** is OWL DL
    - ▶ Otherwise, go to step (1)

5/59

## OWL DL in RDF

- ▶ Mapping from OWL Abstract Syntax to RDF Graphs: given OWL ontology  $O$ , return RDF graph  $T(O)$
- ▶ OWL DL
- ▶ Reuses RDF(S) vocabulary as much as possible
  - e.g. `rdfs:subClassOf`, `rdf:type`
- ▶ Complete translation online at:
  - <http://www.w3.org/TR/owl-semantics/mapping.html>
- ▶ We skip annotations here

7/59

## Translating Ontologies

### Abstract Syntax

Ontology( $O$  axiom<sub>1</sub> ... axiom<sub>n</sub>)

### RDF graph

$O$  rdf:type owl:Ontology .  
T(axiom<sub>1</sub>) ... T(axiom<sub>n</sub>)

In case  $O$  is not present, a blank node is introduced.

8/59

## Translating partial Class definition

### Abstract Syntax

Class(classID partial *description*<sub>1</sub> ... *description*<sub>n</sub>)

### RDF graph

```
classID rdf:type owl:Class .
classID rdfs:subClassOf T(description1) . ...
classID rdfs:subClassOf T(descriptionn) .
```

9/59

## Translating complete Class definition

### Abstract Syntax

Class(classID complete *description*<sub>1</sub> ... *description*<sub>n</sub>)

### RDF graph

```
classID rdf:type owl:Class .
classID owl:intersectionOf T(SEQ description1 ...
descriptionn) .
```

10/59

## Translating complete Class definition

T(SEQ *description*<sub>1</sub> ... *description*<sub>n</sub>)

### RDF graph

```
:l1 rdf:type rdf:List . [opt]
:l1 rdf:first T(item1) . :l1 rdf:rest :l2 .
...
:ln rdf:type rdf:List . [opt]
:ln rdf:first T(itemn) . :ln rdf:rest rdf:nil .
```

11/59

## Enumerated class

### Abstract Syntax

EnumeratedClass(classID iID1 ... iIDn)

### RDF graph

```
classID rdf:type owl:Class .
classID owl:oneOf T(SEQ iID1 ... iIDn) .
```

12/59

## Disjoint classes

### Abstract Syntax

DisjointClasses(*description*<sub>1</sub> ... *description*<sub>n</sub>)

### RDF graph

```
T(descriptionj) owl:disjointWith T(descriptioni) .
1 ≤ i < j ≤ n
```

13/59

## Translating Union

### Abstract Syntax

unionOf(*description*<sub>1</sub> ... *description*<sub>n</sub>)

### RDF graph

```
:x rdf:type owl:Class .
:x owl:unionOf T(SEQ description1 ...
descriptionn) .
```

14/59

## Translating Intersection

### Abstract Syntax

intersectionOf(description1 ... descriptionn)

### RDF graph

```
:x rdf:type owl:Class .
:x owl:intersectionOf T(SEQ description1 ...
descriptionn) .
```

15/59

## Translating Restrictions

### Abstract Syntax

restriction(ID component1 ... componentn)  
(With at least two components)

### RDF graph

```
:x rdf:type owl:Class .
:x owl:intersectionOf
T(SEQ(restriction(ID component1) ...
restriction(ID componentn))) .
```

16/59

## Translating Restrictions

### Abstract Syntax

restriction(ID allValuesFrom(range))

### RDF graph

```
:x rdf:type owl:Restriction .
:x owl:onProperty T(ID) .
:x owl:allValuesFrom T(range) .
```

17/59

## Translating Restrictions

### Abstract Syntax

restriction(ID minCardinality(min))

### RDF graph

```
:x rdf:type owl:Restriction .
:x owl:onProperty T(ID) .
:x owl:minCardinality "min"^^xsd:nonNegativeInteger .
```

18/59

## Translating Individuals

### Abstract Syntax

Individual(iID  
type(type1) ... type(typhen)  
value(pID1 v1) ... value(pIDk vk))

### RDF graph

```
iID rdf:type T(type1) . ... iID rdf:type T(typhen) .
iID T(pID1) T(v1) . ... iID T(pIDk) T(vk) .
```

19/59

## RDF Rules: N3

- ▶ RDF Rules: extension of Turtle to **rules**
- ▶ `rdf:type` may be abbreviated to `a`
- ▶ Syntax:  
 $\{ (triple\ pattern\ .)^* \} \Rightarrow \{ (triple\ pattern\ .)^* \} .$
- ▶ Rule safety:
  - ▶ Variables on right-hand side of  $\Rightarrow$  (**head**) **must** occur on left-hand side **body**
- ▶ Variables implicitly universally quantified over rule
- ▶  $\Rightarrow$  as standard implication in Logic Programming
- ▶ Rule with empty head is **integrity constraint**
- ▶ Rule with empty body is **fact**

22/59

## RDF Rules: examples

```
{?A rdfs:subClassOf ?B. ?B rdfs:subClassOf ?A} =>
  {?A owl:equivalentClass ?B}.
```

```
{?D owl:complementOf ?C.
  ?D owl:equivalentClass owl:Nothing} =>
  {?C owl:equivalentClass owl:Thing}.
```

```
{?A owl:differentFrom ?A} => {}.
```

23/59

## Rule-based OWL reasoning

- ▶ Cannot **exactly** capture OWL DL semantics
- ▶ OWL DL semantics can be approximated to large extent, e.g., equality:
  - ▶ Equality not really in the (rule) language
  - ▶ Can be simulated with `owl:sameAs`
- ▶ Hard to characterize which part of Semantics is captured
- ▶ Can capture part of OWL Full semantics
- ▶ Rule-based RDF reasoning popular on Semantic Web (e.g. Jena, CWM, Euler, ...)

24/59

## Basic Ideas of DLs

- ▶ Basic syntactic building blocks
  - ▶ Atomic concepts
  - ▶ Atomic roles
  - ▶ Individuals
- ▶ Limited constructs for building complex concepts, roles
- ▶ Implicit knowledge can be inferred automatically
  - ▶ Subsumption

26/59

## Inference of implicit knowledge

Consider the TBox:

$Woman \equiv Person \sqcap Female$

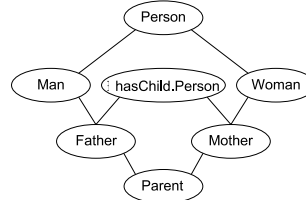
$Man \equiv Person \sqcap \neg Woman$

$Mother \equiv Woman \sqcap \exists hasChild.Person$

$Father \equiv Man \sqcap \exists hasChild.Person$

$Parent \equiv Mother \sqcup Father$

entails the following subsumption hierarchy:



27/59

## Review of DL Basics

- ▶ Set-based term descriptions
- ▶ Intentional knowledge
  - ▶ A DL ontology is not a database!
- ▶ Main inference procedure: subsumption reasoning
  - ▶ If a concept  $D$  is more general than a concept  $C$ , it **subsumes** that concept:  
 $C \sqsubseteq D$
- ▶ Efficient TBox reasoning
- ▶ ABox reasoning (query answering) not well developed

28/59

## OWL constructs

OWL Construct	DL	Example
intersectionOf	$C_1 \sqcap \dots \sqcap C_n$	$Human \sqcap Male$
unionOf	$C_1 \sqcup \dots \sqcup C_n$	$Doctor \sqcup Lawyer$
complementOf	$\neg C$	$\neg Male$
oneOf	$\{o_1, \dots, o_n\}$	$\{john, mary\}$
allValuesFrom	$\forall P.C$	$\forall hasChild.Doctor$
someValuesFrom	$\exists P.C$	$\exists hasChild.Lawyer$
value	$\exists P.\{o\}$	$\exists citizenOf.USA$
minCardinality	$\geq nP.C$	$\geq 2hasChild.Lawyer$
maxCardinality	$\leq nP.C$	$\leq 1hasChild.Male$
cardinality	$= nP.C$	$= 1hasParent.Female$

+ XML Schema datatypes: int, string, real, etc...

29/59

## OWL axioms

OWL Axiom	DL	Example
SubClassOf	$C_1 \sqsubseteq C_2$	<i>Human</i> $\sqsubseteq$ <i>Animal</i> $\sqcap$ <i>Biped</i>
EquivalentClasses	$C_1 \equiv \dots \equiv C_n$	<i>Man</i> $\equiv$ <i>Human</i> $\sqcap$ <i>Male</i>
SubPropertyOf	$P_1 \sqsubseteq P_2$	<i>hasDaughter</i> $\sqsubseteq$ <i>hasChild</i>
EquivalentProperties	$P_1 \equiv \dots \equiv P_n$	<i>cost</i> $\equiv$ <i>price</i>
SameIndividual	$o_1 = \dots = o_n$	<i>President_Bush</i> = <i>G.W_Bush</i>
DisjointClasses	$C_i \sqsubseteq \neg C_j$	<i>Male</i> $\sqsubseteq$ $\neg$ <i>Female</i>
DifferentIndividuals	$o_i \neq o_j$	<i>john</i> $\neq$ <i>peter</i>
inverseOf	$P_1 \equiv P_2^-$	<i>hasChild</i> $\equiv$ <i>hasParent</i> <sup>-</sup>
Transitive	$P^+ \sqsubseteq P$	<i>ancestor</i> <sup>+</sup> $\sqsubseteq$ <i>ancestor</i>
Symmetric	$P \equiv P^-$	<i>connectedTo</i> $\equiv$ <i>connectedTo</i> <sup>-</sup>

30/59

## More examples

```
Class( firstYearCourse partial restriction (isTaughtBy allValuesFrom
( Professor )))

Class(mathCourse partial restriction (isTaughtBy hasValue (949352)))

Class(academicStaffMember partial restriction (teaches someValuesFrom
( undergraduateCourse)))

Class(course partial restriction (isTaughtBy minCardinality(1)))

Class(department partial restriction (hasMember minCardinality(10))
restriction (hasMember maxCardinality(30)))
```

31/59

## More examples

In DL syntax:

```
firstYearCourse  $\sqsubseteq$   $\forall$  isTaughtBy. Professor
mathCourse  $\sqsubseteq$   $\exists$  isTaughtBy. {949352}
academicStaffMember  $\sqsubseteq$   $\exists$  teaches. undergraduateCourse
course  $\sqsubseteq$   $\geq$  1 isTaughtBy
department  $\sqsubseteq$   $\geq$  10 hasMember  $\sqcap$   $\leq$  30 hasMember
```

32/59

## More examples

```
Class(course partial complementOf(staffMember))

Class(peopleAtUni complete unionOf(staffMember student))

Class(facultyInCS complete intersectionOf( faculty
restriction (belongsTo hasValue (CSDepartment))))

Class(adminStaff complete intersectionOf( staffMember
complementOf(unionOf(faculty techSupportStaff))))
```

33/59

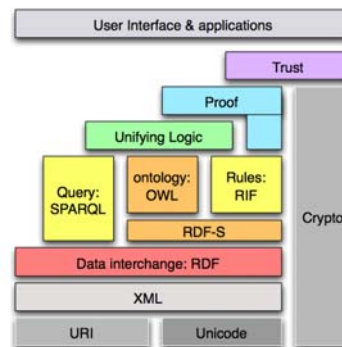
## More examples

In DL syntax:

```
course  $\sqsubseteq$   $\neg$  staffMember
peopleAtUni  $\equiv$  staffMember  $\sqcup$  student
facultyInCS  $\equiv$  faculty  $\sqcap$   $\exists$  belongsTo. {CSDepartment}
adminStaff  $\equiv$  staffMember  $\sqcap$   $\neg$ (faculty  $\sqcup$  techSupportStaff)
```

34/59

## Semantic Web Languages



36/59

## Why relate DL and LP?

Why not use DL or LP for everything?

- ▶ LP: Efficient algorithms and implementations for query answering
- ▶ DL: Efficient algorithms and implementations for subsumption reasoning
- ▶ Thus: it depends on the reasoning task which language to use
- ▶ Many existing implementations of rules systems (e.g., SQL1999  $\approx$  Datalog with linear recursion)
- ▶ Web Ontology Language (OWL) based on DL
- ▶ Thus: in order to leverage existing rule implementations for the Semantic Web, we need to translate!

37/59

## Recap: Logic Programming

- ▶ Any FOL term is a term in LP
- ▶ Any FOL atomic formula is an atomic formula in LP
- ▶ Any Horn formula is a rule in LP (quantification usually omitted)
  - ▶  $H \leftarrow B_1 \wedge \dots \wedge B_n$
- ▶ Logic programming is a syntactic subset of FOL
- ▶ **Note!** Negation-as-failure in LP is an **extension** of Horn rules
  - ▶  $\neg \neq \text{not}$

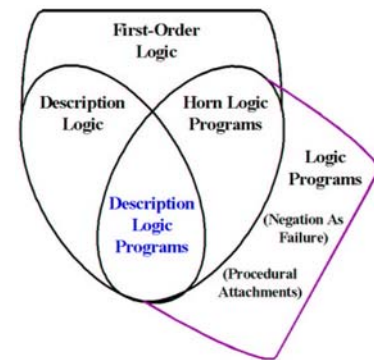
38/59

## Entailment

- ▶ General First-Order entailment:
  - ▶  $\phi \models \psi$  iff for every interpretation  $\mathcal{I}$ : if  $\mathcal{I} \models \phi$  then  $\mathcal{I} \models \psi$
  - ▶ Thus, the set of models of  $\phi$   $M(\phi)$  is a subset of  $M(\psi)$ :  $M(\phi) \subseteq M(\psi)$
  - ▶ e.g.,  $p(x) \wedge q(x) \models p(x)$
- ▶ Ground entailment:
  - ▶  $\phi \models \psi_{ground}$  iff for every interpretation  $\mathcal{I}$ : if  $\mathcal{I} \models \phi$  then  $\mathcal{I} \models \psi_{ground}$  and  $\psi_{ground}$  **does not** contain variables
  - ▶ e.g.,  $(p(x) \rightarrow q(x)) \wedge p(a) \models q(a)$
- ▶ Logic Programming only defines ground entailment
- ▶ Horn Logic (i.e., Horn subset of FOL) is equivalent to Logic Programming wrt. ground entailment
  - ▶ For any set of Horn formulas  $\phi$  and a ground Horn formula  $\psi_{ground}$ :  $\phi \models_{FOL} \psi_{ground}$  iff  $\phi \models_{LP} \psi_{ground}$
  - ▶  $\models_{FOL}$  is classical First-Order entailment;  $\models_{LP}$  is LP entailment

39/59

## Relation between DL and LP



41/59

## Description Logic Programs

- ▶ "Intersection" of Description Logics and Logic Programming
- ▶ That part of Description Logics (OWL in particular) which can be translated to a Logic Program
- ▶ Horn Logic subset of *SHOIN*, **reduced** to a Logic Program: Description Logic Program: DLP
- ▶ General idea:
  1. Translate *SHOIN* axiom to First-Order Logic
  2. Rewrite to Horn Logic
    - ▶ If rewriting not possible: formula not in DLP
  3. Reduce to Logic Program

42/59

## Recap: *SHOIN*

Concept descriptions

$C, D \longrightarrow A$		(atomic concept)
$\top$		(universal concept)
$\perp$		(bottom concept)
$C \sqcap D$		(intersection)
$C \sqcup D$		(disjunction)
$\neg C$		(negation)
$\forall R.C$		(value restriction)
$\exists R.C$		(existential quantification)
$\{o_1, \dots, o_n\}$		(enumeration)
$\exists R.\{o\}$		(hasValue)
$\geq nR$		(minimal cardinality)
$\leq nR$		(maximal cardinality)

43/59

## Recap: SHOIN

Individual assertions

$$a \in C$$

$$\langle a, b \rangle \in R$$

44/59

## Recap: SHOIN

Axioms

$$C \sqsubseteq D \quad (\text{class subsumption})$$

$$C \equiv D \quad (\text{equivalence})$$

$$Q \sqsubseteq R \quad (\text{property subsumption})$$

$$R \equiv Q^- \quad (\text{inverse roles})$$

$$R \equiv R^- \quad (\text{symmetric roles})$$

$$R^+ \sqsubseteq R \quad (\text{transitive properties})$$

45/59

## Mapping SHOIN to FOL

$A$ (atomic concept)	$A(x)$
$\top$	$\top$
$\perp$	$\perp$
$C \sqcap D$	$tr(C) \wedge tr(D)$
$C \sqcup D$	$tr(C) \vee tr(D)$
$\neg C$	$\neg tr(C)$
$\forall R.C$	$\forall y : R(x, y) \rightarrow tr(C, y)$
$\exists R.C$	$\exists y : R(x, y) \wedge tr(C, y)$
$\{o_1, \dots, o_n\}$	$x = o_1 \vee \dots \vee x = o_n$
$\exists R.\{o\}$	$R(x, o)$
$\geq nR$	$\exists y_1, \dots, y_n : \bigwedge R(x, y_i) \wedge \bigwedge y_i \neq y_j$
$\leq nR$	$\forall y_1, \dots, y_{n+1} : \bigwedge R(x, y_i) \rightarrow \bigvee y_i = y_j$

46/59

## Mapping SHOIN to FOL

$a \in A$	$A(a)$
$\langle a, b \rangle \in R$	$R(a, b)$
$C \sqsubseteq D$	$\forall x : tr(C, x) \rightarrow tr(D, x)$
$C \equiv D$	$\forall x : tr(C, x) \leftrightarrow tr(D, x)$
$Q \sqsubseteq R$	$\forall x, y : Q(r, y) \rightarrow R(x, y)$
$R \equiv Q^-$	$\forall x, y : R(x, y) \leftrightarrow Q(y, x)$
$R^+ \sqsubseteq R$	$\forall x, y, z : R(x, y) \wedge R(y, z) \rightarrow R(x, z)$

47/59

## Identifying DLP (1)

- ▶ Next step: identify DLP-fragment of SHOIN
- ▶ Start with the axioms
  - ▶ Easy case: properties
 

$Q \sqsubseteq R$	$\forall x, y : Q(r, y) \rightarrow R(x, y)$
$R \equiv Q^-$	$\forall x, y : R(x, y) \leftrightarrow Q(y, x)$
$R^+ \sqsubseteq R$	$\forall x, y, z : R(x, y) \wedge R(y, z) \rightarrow R(x, z)$

 All Horn Logic!
  - ▶ Equivalence axioms:  $C \equiv D$   
 Reduce to:  $C \sqsubseteq D, D \sqsubseteq C$
  - ▶ Subsumption axioms:  $C \sqsubseteq D \mapsto \forall x : tr(C, x) \rightarrow tr(D, x)$   
 $tr(C, x)$  is the **body** of the rule  
 $tr(D, x)$  is the **head** of the rule

48/59

## Identifying DLP (2)

- ▶  $C \sqsubseteq D$ 
  - ▶ **Remember:** the **body** is a conjunction of literals; the **head** is a single literal
  - ▶ Thus:  $C$  and  $D$  look different
  - ▶  $C$  is called the **left-hand side** (lhs)
  - ▶  $D$  is called the **right-hand side** (rhs)
- ▶ We distinguish between:
  - ▶ Descriptions allowed on the left-hand side
  - ▶ Descriptions allowed on the right-hand side

49/59

## Identifying DLP (3)

SHOIN	FOL	lhs	rhs
$A$ (atomic concept)	$A(x)$	+	+
$\top$	$\top$	+/-	+
$\perp$	$\perp$	+	-
$C \sqcap D$	$tr(C) \wedge tr(D)$	+	+
$C \sqcup D$	$tr(C) \vee tr(D)$	+	-
$\neg C$	$\neg tr(C)$	-	-
$\forall R.C$	$\forall y : R(x, y) \rightarrow tr(C, y)$	-	+
$\exists R.C$	$\exists y : R(x, y) \wedge tr(C, y)$	+	-
$\{o_1, \dots, o_n\}$	$x = o_1 \vee \dots \vee x = o_n$	+	-
$\exists R.\{o\}$	$R(x, o)$	+	+
$\geq nR$	$\exists y_1, \dots, y_n : \bigwedge R(x, y_i) \wedge \bigwedge y_i \neq y_j$	+	-
$\leq nR$	$\forall y_1, \dots, y_{n+1} : \bigwedge R(x, y_i) \rightarrow \bigvee y_i = y_j$	-	-

50/59

## Identifying DLP (4)

- Individual assertions:

$$a \in A \quad \Bigg| \quad A(a)$$

$$\langle a, b \rangle \in R \quad \Bigg| \quad R(a, b)$$

Trivially in DLP!

- Example:  $\top \sqcap \exists P.A \sqcap \{o_1, o_2\} \sqcap B \sqsubseteq \forall P.B$

Step 1: rewrite to FOL:

$$\forall x : \top \wedge (\exists y : P(x, y) \wedge A(y)) \wedge (x = o_1 \vee x = o_2) \wedge B(x) \rightarrow (\forall y : P(x, y) \wedge B(y))$$

Step 2: rewrite to Horn Logic:

$$1: \forall x : \top \wedge (\exists y : P(x, y) \wedge A(y)) \wedge (x = o_1 \vee x = o_2) \wedge B(x) \rightarrow (\forall y : P(x, y) \wedge B(y)) \Leftrightarrow$$

51/59

## Example (cont'd)

$$2: \forall x : \neg \top \vee \neg (\exists y : P(x, y) \wedge A(y)) \vee \neg (x = o_1 \vee x = o_2) \vee \neg B(x) \vee (\forall y : P(x, y) \wedge B(y)) \Leftrightarrow$$

$$3: \forall x : \neg (\exists y : P(x, y) \wedge A(y)) \vee \neg (x = o_1 \vee x = o_2) \vee \neg B(x) \vee (\forall y : P(x, y) \wedge B(y)) \Leftrightarrow$$

$$4: \forall x : (\forall y : \neg (P(x, y) \wedge A(y))) \vee \neg (x = o_1 \vee x = o_2) \vee \neg B(x) \vee (\forall y : P(x, y) \wedge B(y)) \Leftrightarrow$$

$$5: \forall x, y : \neg P(x, y) \vee \neg A(y) \vee \neg (x = o_1 \vee x = o_2) \vee \neg B(x) \vee (\forall y : P(x, y) \wedge B(y)) \Leftrightarrow$$

52/59

## Example (cont'd)

$$6: \forall x, y : \neg P(x, y) \vee \neg A(y) \vee (\neg x = o_1 \wedge \neg x = o_2) \vee \neg B(x) \vee (\forall y : P(x, y) \wedge B(y)) \Leftrightarrow$$

$$7: \forall x, y, z : \neg P(x, y) \vee \neg A(y) \vee (\neg x = o_1 \wedge \neg x = o_2) \vee \neg B(x) \vee (P(x, z) \wedge B(z)) \Leftrightarrow$$

$$8: \forall x, y, z : \neg P(x, y) \vee \neg A(y) \vee \neg x = o_1 \vee \neg B(x) \vee (P(x, z) \wedge B(z)) \Leftrightarrow$$

$$\forall x, y, z : \neg P(x, y) \vee \neg A(y) \vee \neg x = o_2 \vee \neg B(x) \vee (P(x, z) \wedge B(z)) \Leftrightarrow$$

53/59

## Example (cont'd)

$$9: \forall x, y, z : \neg P(x, y) \vee \neg A(y) \vee \neg x = o_1 \vee \neg B(x) \vee P(x, z)$$

$$\forall x, y, z : \neg P(x, y) \vee \neg A(y) \vee \neg x = o_1 \vee \neg B(x) \vee B(z)$$

$$\forall x, y, z : \neg P(x, y) \vee \neg A(y) \vee \neg x = o_2 \vee \neg B(x) \vee P(x, z)$$

$$\forall x, y, z : \neg P(x, y) \vee \neg A(y) \vee \neg x = o_2 \vee \neg B(x) \vee B(z)$$

Step 3: Reduce to Logic Program

$$P(x, z) :- P(x, y), A(y), x=o1, B(x).$$

$$B(z) :- P(x, y), A(y), x=o1, B(x).$$

$$P(x, z) :- P(x, y), A(y), x=o2, B(x).$$

$$B(z) :- P(x, y), A(y), x=o2, B(x).$$

Semantic reduction: only entailment of ground atoms

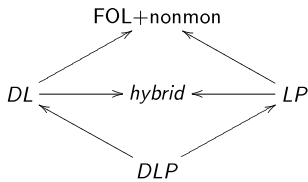
54/59

## Integration of DL and LP beyond DLP

- DLP has limited expressiveness
  - No disjunction
  - No existentials
  - No negation
  - No chaining variables over predicates
- DLP allows **extension** to DL or LP, no real **interoperation**

56/59

## Outlook: Integration beyond DLP



- ▶ Extension of DLP
  - ▶ Interoperation only for inexpressive ontologies
- ▶ Unified language
  - ▶ Obviously undecidable
  - ▶ Little research has been done
- ▶ Hybrid integration
  - ▶ Full DL and LP
  - ▶ Interaction limited to retain decidability
  - ▶ Several existing approaches (e.g. [Eiter, 2004; Rosati, 2006])

57/59

## Summary

### OWL in RDF

Mapping OWL DL to RDF

### Reasoning with OWL

RDF-based Rule Reasoning  
DL-based Reasoning

### OWL DL and Logic Programming

Description Logic Programs  
Beyond DLP

58/59

## Required reading

- ▶ OWL Guide: <http://www.w3.org/TR/owl-guide/>
- ▶ Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. **Journal of Web Semantics**, 1(1):7, 2003.

## Further reading

- ▶ Grosz et al. Description logic programs: Combining logic programs with description logic. In **WWW-2003**.
- ▶ OWL Reference: <http://www.w3.org/TR/owl-ref/>
- ▶ OWL Abstract syntax and Semantics: <http://www.w3.org/TR/owl-semantics/>
- ▶ T. Eiter, T. Lukasiewicz, R. Schindlauer, H. Tompits: Combining Answer Set Programming with Description Logics for the Semantic Web. In **KR 2004**.

59/59