

# Intelligent Systems on the World Wide Web

## Ontologies

Lecture Slides  
Steffen Staab  
Institute for Applied Computer Science and Formal  
Description Methods (AIFB)  
Karlsruhe University

## Ontology – Philosophical Definition

- Ontology ....  
defined in a philosophical sense, is the study of the nature of being.
- Aristotle, Metaphysics, IV, 1
- Questions:

*What is the meaning of „to be“?*

*Which properties are shared by all things?*

## Ontology – Definition in Computer Science

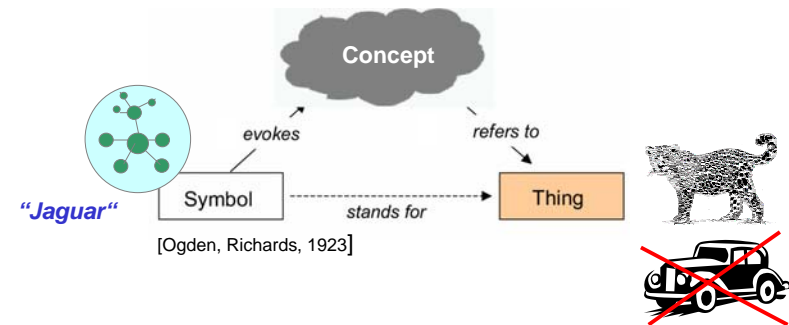
Gruber 93:

An Ontology is a

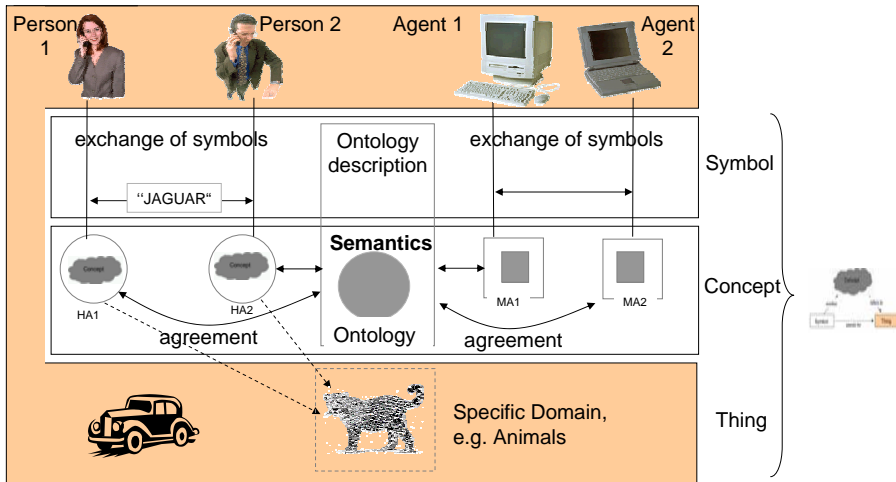
formal specification  
of a shared  
conceptualization  
of a domain of interest

- ⇒ machine-understandable
- ⇒ group of people
- ⇒ about concepts
- ⇒ between general description and individual use

## Triangle of Meaning

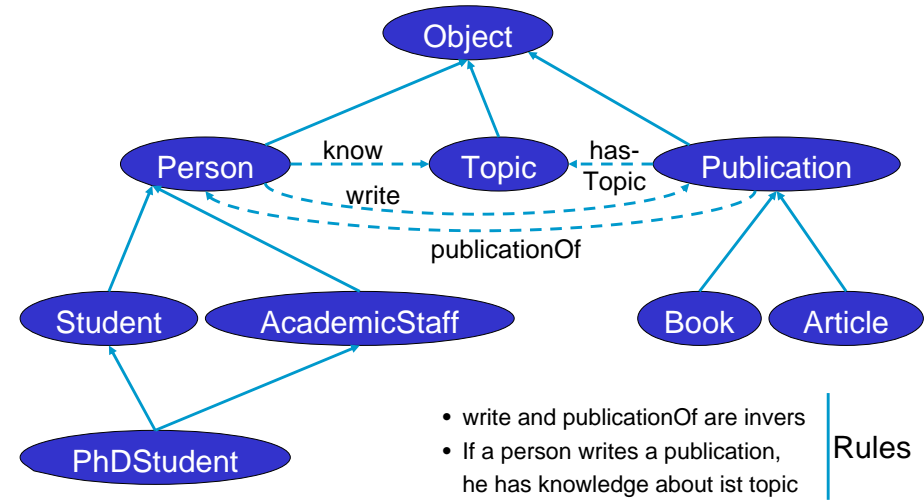


## Communication



Slide 5

## An Example Ontology



Slide 6

## Ontology Representation

- **Predicate Logic**
- Semantic Web Languages
- Description Logic
- Frame Logic



## FOL as an ontology language

- Propositional Logic is too limited.
  - The world only consists of facts.
  - I.e., we cannot express the following:
    - All men are mortal*
    - Socrates is a man*
    - Therefore, Socrates is mortal*
- The world consists of **objects**, and **properties** that distinguish one object from the other. Among objects, **relations** hold. Some relations are **functions**.

Slide 8

## FOL as an ontology language II

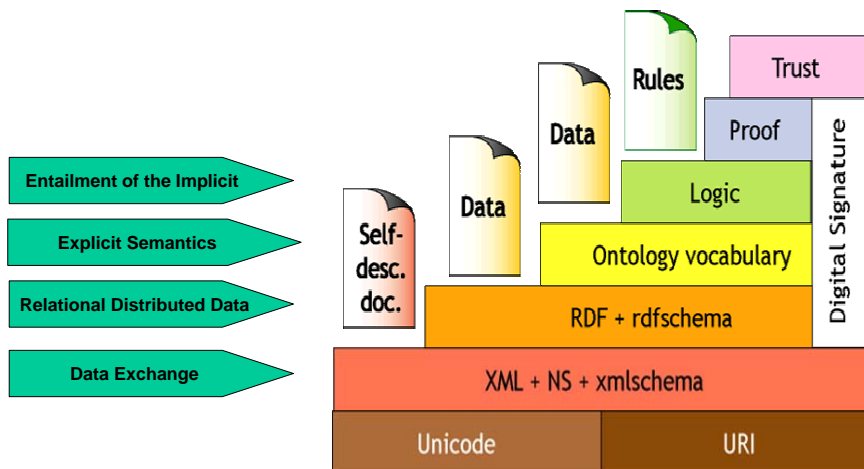
- Problems
  - Expressiveness vs. Tractability
  - Domain of Discourse unstructured
    - Class-based formalisms
    - Frames

## Ontology Representation

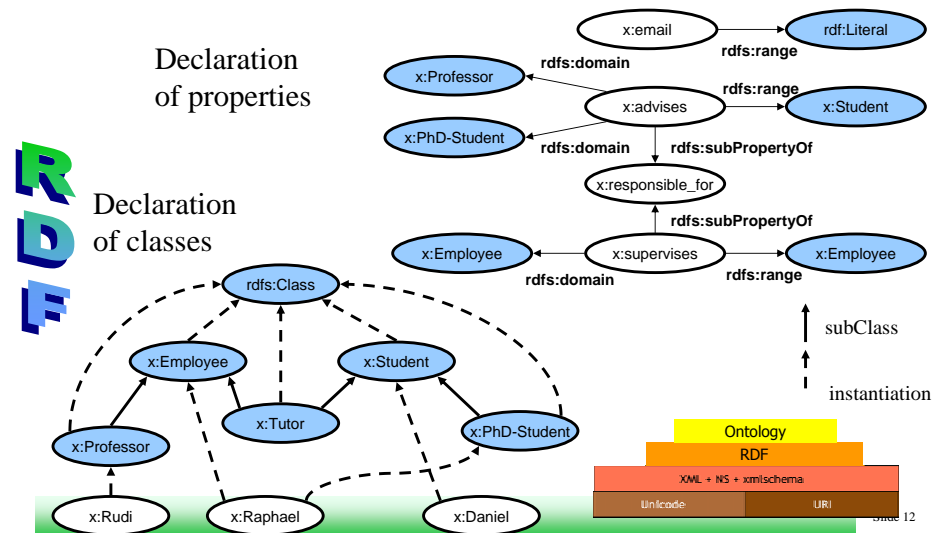
- Predicate Logic
- **Semantic Web Languages**
- Description Logic
- Frame Logic



## Semantic Web Architecture

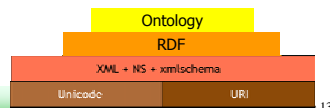


## RDF Schema – simple ontologies



## RDF Schema Elements

- Only named classes
- Primitive classes
- SubClassOf (partial order)
- SubPropertyOf (partial order)
- Multiple instantiation of classes
- Global domain/range
- (Un)named instances



Slide 13

## Formal Models of RDF (1/3)

- **Official Semantics: RDF Model Theory**  
[ → <http://www.w3.org/TR/rdf-mt/> ]

*Specification of a precise semantics for RDF (and RDFS), and of corresponding entailment and inference rules which are sanctioned by the semantics.*

- **Other Proposals:**
  - RDFS(FA)  
[ → <http://dl-web.man.ac.uk/rdfsfa/> ]  
UML-Like Stratification
  - RDF in First-Order Logic  
[ → [http://nestroy.wi-inf.uni-essen.de/rdf/logical\\_interpretation/](http://nestroy.wi-inf.uni-essen.de/rdf/logical_interpretation/) ]  
( Outdated RDF semantics in FOL )

Slide 14

## Formal Models of RDF (2/3)

- RDF-MT is based on classical Tarski-style Model Theory
- Some Entailment rules for RDFS:

	If E contains:	then add:
rdf1	xxx aaa yyy .	aaa rdf:type rdf:Property .
rdfs2	xxx aaa yyy . aaa rdfs:domain zzz .	xxx rdf:type zzz .
rdfs3	xxx aaa uuu . aaa rdfs:range zzz .	uuu rdf:type zzz .
rdfs4a	xxx aaa yyy .	xxx rdf:type rdfs:Resource
rdfs4b	xxx aaa uuu .	uuu rdf:type rdfs:Resource
rdfs5a	aaa rdfs:subPropertyOf bbb . bbb rdfs:subPropertyOf ccc .	aaa rdfs:subPropertyOf ccc
rdfs5b	xxx rdf:type rdf:Property . xxx aaa yyy .	xxx rdfs:subPropertyOf xxx

Slide 15

## Formal Model of RDF (3/3)

- The entailment process terminates on any finite RDF graph  
→ only finitely many possible triples can be formed from a given finite vocabulary.
- Example Graph (Single Triple):  
[foo bar baz ].
- Closure (for mentioned rules only !):
  1. foo bar baz . Source
  2. foo rdf:type rdfs:Resource . Rule 4a on (1)
  3. baz rdf:type rdfs:Resource . Rule 4a on (1)
  4. bar rdf:type rdf:Property . Rule 1 on (1)
  5. rdf:type rdf:type rdf:Property . Rule 1 on (4)
  6. rdf:type rdfs:subPropertyOf rdf:type. Rule 5b on (5)
  7. bar rdfs:subPropertyOf bar. Rule 5b on (4)
  8. rdfs:subPropertyOf rdf:type rdf:Property Rule 1 on (6)

Slide 16

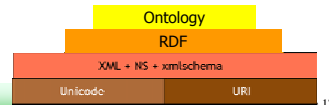
## Need for Higher Expressivity I

Work on Web Ontology Language (OWL)

- „WebOnt“ working group at W3C

[ → <http://www.w3.org/2001/sw/WebOnt/> ]

- Started in November 2001
- Currently 53 Members
- Input / Starting point
  - OIL, DAML-ONT, DAML+OIL
  - Work of EU/US research projects
    - OnToKnowledge [ → [www.ontoknowledge.org](http://www.ontoknowledge.org) ]
    - DARPA/DAML [ → [www.daml.org](http://www.daml.org) ]

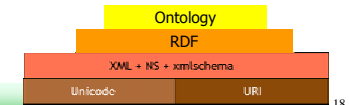


Slide 17

## Need for Higher Expressivity II

- (In)equality of instances
  - Property characteristics
  - Complex class constructors
  - Elements of Description Logic
- Instances
  - Property restrictions
  - Class constructors
  - Rules

→ Elements of F-Logic



Slide 18

Ausnahmsweise heute: F-Logic zuerst!

Für die Veranstalter des eOrganisation Kick-offs

-> Lese: [http://www.ontoprise.de/documents/tutorial\\_flogic.pdf](http://www.ontoprise.de/documents/tutorial_flogic.pdf)

Notiz: der Vorgänger

<http://www.informatik.uni-freiburg.de/~dbis/florid/tutorial.ps.gz>

Ist sehr ähnlich, bis auf geringe syntaktische Unterschiede

Slide 19

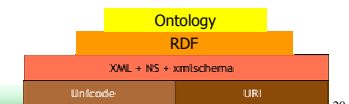
## OWL Documentation

Available via

[→ <http://www.w3.org/2001/sw/WebOnt/> ]

Since 2004: **Recommendation**

- Requirements for a Web Ontology Language
- OWL Guide
- OWL Reference Description
- OWL Formal specification
- OWL Feature Synopsis



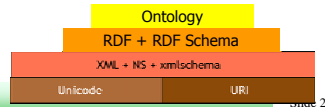
Slide 20



# (In)Equality I

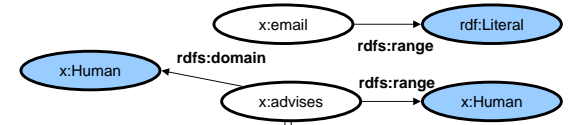
- sameClassAs
- samePropertyAs
- sameIndividualAs
- differentIndividualAs  
*DL-like, since OWL does not take Unique Names Assumption (UNA)*

→ Very useful for information integration, facilitates content interoperability

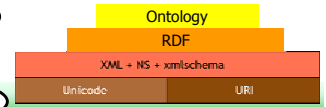
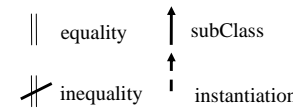
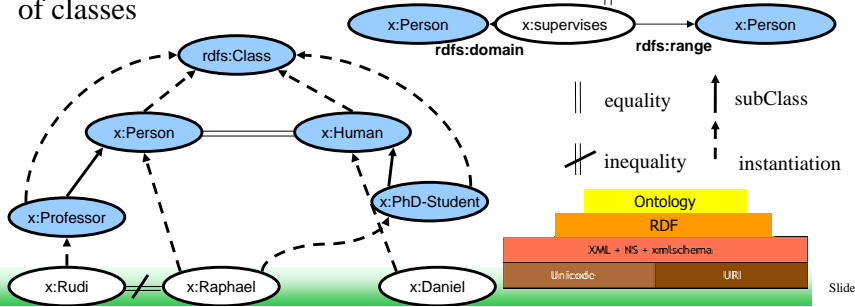


# (In)Equality II

Declaration of properties



Declaration of classes

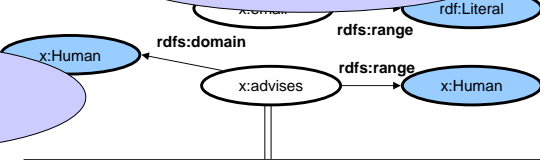


# (In)Equality II

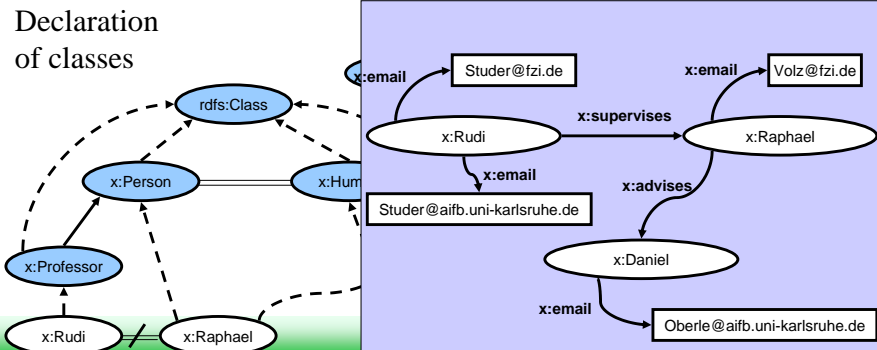


Declaration

Raphael, Rudi, Daniel



Declaration of classes

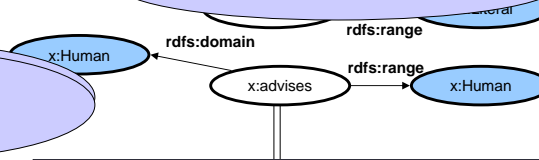


# (In)Equality II

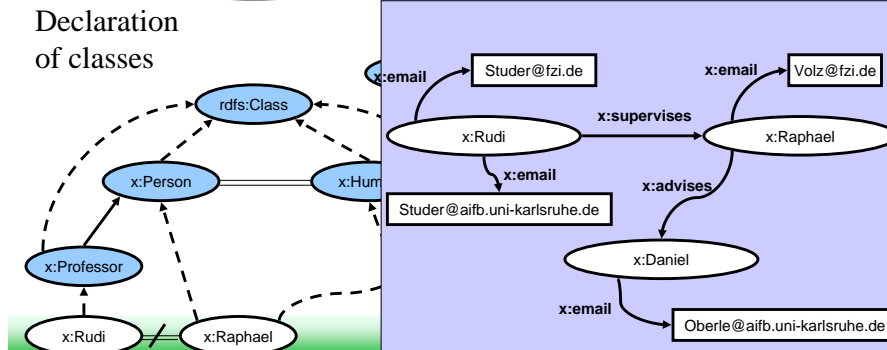


Declaration

Raphael



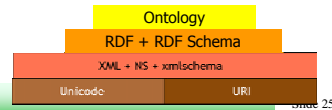
Declaration of classes





## Property characteristics I

- inverseOf
- transitive
- Symmetric



Slide 25



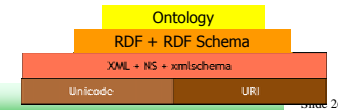
## Property characteristics II

Cardinality constraints

- MinCardinality
- MaxCardinality

Several short hand notations, e.g.:

- Functional
  - a short hand for assigning min. Cardinality 0 and max. Cardinality 1
  - e.g. property „first degree“ of a Person
  - cannot be combined with transitivity to remain decidable (See Horrocks, Sattler, Tobies)

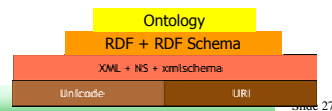


Slide 26



## Property characteristics III

- InverseFunctional
  - instances can be distinctly identified via the property value
  - (~ primary keys)
  - E.g. no two people can hold the same passport.

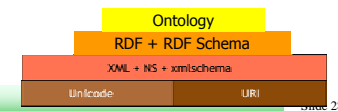


Slide 27



## Property characteristics IV

- Local Range Constraints
  - Restricts property ranges locally
- allValuesFrom
  - universal quantification (Allquantor)
  - E.g. Humans can only give birth to humans, whereas Animals give birth to animals
- someValuesFrom
  - existential quantification
  - E.g. EngineeringStudents must take at least one MathClass whereas Students can take arbitrary classes

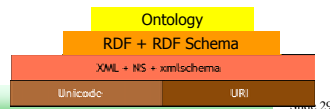


Slide 28



## Extended class definitions

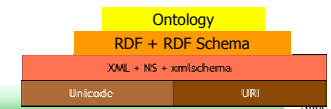
- Defined Classes
  - Class extension is (partly) virtual, also made up by instances that additionally meet class definition constraints
  - Class Descriptions can be arbitrarily complex and combine all available class definition constructs
- Enumerated Classes
  - Set of instances is closed*
  - e.g. Continents*



## Defined Classes

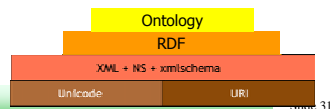
### Class Definition Constraints

- Property fillers
  - Selection on instances
  - *e.g. DINKs are those Couples that have dual income and no kids*
- Boolean combinations
  - Union, Intersection, Complement
  - *e.g. AcademicStaff is the union of Lecturers, Researchers and Professors*
- Disjoint Classes
  - E.g. Male and Female are disjoint*



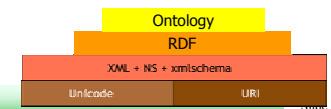
## OWL „Extra-Logical“ features

- Inclusion of other ontologies
- Assignment of metadata to ontology definitions
  - AnnotationProperties



## Layering in OWL

- 3 Layers
  - OWL Lite
    - **easy to implement** (thus encouraging toolbuilders)
    - **Sufficiently more expressive than RDF Schema** (to justify existence)
  - OWL DL
    - **Corresponds to very expressive DL**
  - OWL Full
    - **Fully compatible with RDF semantics**



## An elaborated example

Slide 33

## People & Pets - 1

[Namespaces:

```
rdf = http://www.w3.org/1999/02/22-rdf-syntax-ns#
xsd = http://www.w3.org/2001/XMLSchema#
rdfs = http://www.w3.org/2000/01/rdf-schema#
owl = http://www.w3.org/2002/07/owl#
pp = http://cohse.semanticweb.org/ontologies/people#
```

]

Ontology(  
Class(pp:bone partial)  
Class(pp:brain partial)  
Class(pp:white+thing partial)

```
Class(pp:plant partial)
Class(pp:grass partial pp:plant)
Class(pp:tree partial pp:plant)
Class(pp:leaf partial restriction(pp:part_of
  someValuesFrom(pp:tree)))
```

Slide 34

## People & Pets - 2

```
Class(pp:vehicle partial)
Class(pp:bicycle partial pp:vehicle)
Class(pp:bus partial pp:vehicle)
Class(pp:car partial pp:vehicle)
Class(pp:lorry partial pp:vehicle)
Class(pp:truck partial pp:vehicle)
Class(pp:van partial pp:vehicle)
Class(pp:company partial)
Class(pp:bus+company partial pp:company)
Class(pp:haulage+company partial pp:company)
```

```
Class(pp:publication partial)
Class(pp:magazine partial pp:publication)
Class(pp:broadsheet partial pp:newspaper)
Class(pp:quality+broadsheet partial pp:broadsheet)
Class(pp:tabloid partial pp:newspaper)
Class(pp:red+top partial pp:tabloid)
Class(pp:newspaper partial unionOf(pp:broadsheet pp:tabloid)
  pp:publication)
```

Slide 35

## People & Pets - 3

```
Class(pp:male partial)
Class(pp:female partial)
Class(pp:young partial)
Class(pp:adult partial)
Class(pp:elderly partial pp:adult)
```

```
Class(pp:pet complete restriction(pp:is_pet_of
  someValuesFrom(owl:Thing)))
```

```
Class(pp:animal partial restriction(pp:eats
  someValuesFrom(owl:Thing)))
Class(pp:vegetarian complete
  intersectionOf(pp:animal
    restriction(pp:eats allValuesFrom(complementOf(pp:animal)))
    restriction(pp:eats
      allValuesFrom(complementOf(restriction(pp:part_of
        someValuesFrom(pp:animal)))))))
```

Slide 36

## People & Pets - 4

```

Class(pp:duck partial pp:animal)
Class(pp:cat partial pp:animal)
Class(pp:tiger partial pp:animal)
Class(pp:dog partial restriction(pp:eats
  someValuesFrom(pp:bone)))
Class(pp:sheep partial pp:animal restriction(pp:eats allValuesFrom
  pp:grass))
Class(pp:giraffe partial pp:animal restriction(pp:eats allValuesFrom
  pp:leaf))
Class(pp:cow partial pp:vegetarian)
Class(pp:mad+cow complete
  intersectionOf(pp:cow restriction(pp:eats
  someValuesFrom(intersectionOf(pp:brain
  restriction(pp:part_of someValuesFrom pp:sheep))))))
  
```

## People & Pets - 6

```

Class(pp:person partial pp:animal)
Class(pp:kid complete intersectionOf(pp:young pp:person))
Class(pp:man complete intersectionOf(pp:person pp:male pp:adult))
Class(pp:woman complete intersectionOf(pp:female pp:person pp:adult))
Class(pp:old+lady complete intersectionOf(pp:elderly pp:female pp:person))
Class(pp:old+lady partial
  intersectionOf(restriction(pp:has_pet allValuesFrom(pp:cat))
  restriction(pp:has_pet someValuesFrom(pp:animal))))
Class(pp:grownup complete intersectionOf(pp:person pp:adult))

Class(pp:animal+lover complete
  intersectionOf(pp:person restriction(pp:has_pet minCardinality(3))))
Class(pp:pet+owner complete
  intersectionOf(restriction(pp:has_pet someValuesFrom(pp:animal)) pp:person))
Class(pp:cat+liker complete
  intersectionOf(restriction(pp:likes someValuesFrom(pp:cat)) pp:person))
Class(pp:cat+owner complete
  intersectionOf(pp:person restriction(pp:has_pet someValuesFrom(pp:cat))))
Class(pp:dog+liker complete
  intersectionOf(restriction(pp:likes someValuesFrom(pp:dog)) pp:person))
Class(pp:dog+owner complete
  intersectionOf(restriction(pp:has_pet someValuesFrom(pp:dog)) pp:person))
  
```

```

Class(pp:driver partial pp:adult)
Class(pp:driver complete
  intersectionOf(restriction(pp:drives someValuesFrom(pp:vehicle)) pp:person))
Class(pp:lorry+driver complete
  intersectionOf(restriction(pp:drives someValuesFrom(pp:lorry)) pp:person))
Class(pp:haulage+worker complete
  restriction(pp:works_for
  someValuesFrom(unionOf(restriction(pp:part_of
  someValuesFrom(pp:haulage+company))
  pp:haulage+company))))
  
```

## People & Pets - 7

```

Class(pp:haulage+truck+driver complete
  intersectionOf(pp:person
  restriction(pp:drives someValuesFrom(pp:truck))
  restriction(pp:works_for
  someValuesFrom(
  restriction(pp:part_of someValuesFrom(pp:haulage+company))))))
Class(pp:van+driver complete
  intersectionOf(pp:person restriction(pp:drives someValuesFrom(pp:van))))
Class(pp:bus+driver complete
  intersectionOf(pp:person restriction(pp:drives someValuesFrom(pp:bus))))
Class(pp:white+van+man complete
  intersectionOf(pp:man
  restriction(pp:drives
  someValuesFrom(intersectionOf(pp:white+thing pp:van))))))
Class(pp:white+van+man partial restriction(pp:reads allValuesFrom pp:tabloid))
  
```

## People & Pets - 6

```

DisjointClasses(pp:broadsheet pp:tabloid)
DisjointClasses(pp:dog pp:cat)
DisjointClasses(pp:young pp:adult)
DisjointClasses(unionOf(pp:animal restriction(pp:part_of
  someValuesFrom pp:animal))
  unionOf(pp:plant restriction(pp:part_of
  someValuesFrom pp:plant)))
  
```

## People & Pets - 9

ObjectProperty(pp:likes)  
 ObjectProperty(pp:drives)  
 ObjectProperty(pp:eaten\_by)  
 ObjectProperty(pp:eats inverseOf(pp:eaten\_by) domain(pp:animal))  
 ObjectProperty(pp:works\_for)  
 ObjectProperty(pp:reads range(pp:publication))

ObjectProperty(pp:has\_parent)  
 ObjectProperty(pp:has\_father range(pp:man))  
 ObjectProperty(pp:has\_mother range(pp:woman))  
 ObjectProperty(pp:has\_child)

ObjectProperty(pp:has\_pet domain(pp:person) range(pp:animal))  
 ObjectProperty(pp:is\_pet\_of inverseOf(pp:has\_pet))  
 ObjectProperty(pp:part\_of)  
 ObjectProperty(pp:has\_part inverseOf(pp:part\_of))

DataProperty(pp:service\_number range(xsd:integer))

SubPropertyOf(pp:has\_mother pp:has\_parent)  
 SubPropertyOf(pp:has\_pet pp:likes)  
 SubPropertyOf(pp:has\_father pp:has\_parent)

Slide 41

## People & Pets - 10

Individual(pp:Tom type(owl:Thing))  
 Individual(pp:Dewey type(pp:duck))  
 Individual(pp:Huey type(pp:duck))  
 Individual(pp:Louie type(pp:duck))  
 Individual(pp:Fido type(pp:dog))  
 Individual(pp:Flossie type(pp:cow))  
 Individual(pp:Fluffy type(pp:tiger))  
 Individual(pp:Rex type(pp:dog) value(pp:is\_pet\_of pp:Mick))  
 Individual(pp:Q123+ABC type(pp:white+thing) type(pp:van))  
 Individual(pp:Tibbs type(pp:cat))

Individual(pp:Daily+Mirror type(owl:Thing))  
 Individual(pp:The+Guardian type(pp:broadsheet))  
 Individual(pp:The+Sun type(pp:tabloid))  
 Individual(pp:The+Times type(pp:broadsheet))

Individual(pp:The42 type(pp:bus)  
 value(pp:service\_number  
 [42^^http://www.w3.org/2001/XMLSchema#integer]))

Slide 42

## People & Pets - 11

Individual(pp:Kevin type(pp:person))  
 Individual(pp:Fred type(pp:person) value(pp:has\_pet pp:Tibbs))  
 Individual(pp:Joe type(pp:person)  
 type(restriction(pp:has\_pet maxCardinality(1)))  
 value(pp:has\_pet pp:Fido))  
 Individual(pp:Mick type(pp:male)  
 value(pp:reads pp:Daily+Mirror)  
 value(pp:drives pp:Q123+ABC))  
 Individual(pp:Minnie type(pp:elderly) type(pp:female)  
 value(pp:has\_pet pp:Tom))  
 Individual(pp:Walt type(pp:person)  
 value(pp:has\_pet pp:Huey)  
 value(pp:has\_pet pp:Louie)  
 value(pp:has\_pet pp:Dewey))  
 )

Slide 43

## Ontology Representation

- Predicate Logic
- Semantic Web Languages
- **Description Logic**
- Frame Logic



## Description Logic (I)

- OWL equivalent to very expressive Description Logic
- Description Logic
  - Family of languages rather than one language
  - Leading Language in Knowledge Representation (KR) [ → www.kr.org ]
  - ... 25 years of research ...  
→ only shallow introduction possible
- Core distinction between
  - **class definitions** (T-Box ≈ Ontology)
  - **instance definitions** (A-Box ≈ Knowledge Base)

Slide 45

## Description Logic (II)

- Benefits for OWL
  - Well-defined semantics
  - Formal properties well understood (complexity & decidability)
  - Known reasoning algorithms
  - Implemented systems
- Expressive Power determined by
  - Kinds of constructors provided
  - Kinds of axioms allowed

Slide 46

## DL Class Constructors

Constructor	DL Syntax	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	$\{x_1 \dots x_n\}$	{ Germany, France }
allValuesFrom	$\forall P.C$	$\forall$ hasParent.Human
someValuesFrom	$\exists P.C$	$\exists$ hasChild.Lawyer
hasValue	$\exists P.\{x\}$	$\exists$ citizenOf.{USA}
minCardinality	$\exists \geq n P$	$\exists \geq 1$ passport
maxCardinality	$\exists \leq n P$	$\exists \leq 2$ hasArm
cardinality	$\exists = n P$	$\exists = n$ head

Arbitrarily complex nesting of constructors, e.g.  
 Person  $\sqcap \exists$  hasChild.(Doctor  $\sqcup \forall$  hasChild.Doctor)

Slide 47

## DL Axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \subseteq C_2$	Human $\subseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \subseteq P_2$	hasDaughter $\subseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
sameAs	$\{x_1\} \equiv \{x_2\}$	{PresidentBush} $\equiv$ {GWBush}
disjointWith	$C_1 \subseteq \neg C_2$	Male $\subseteq \neg$ Female
differentIndividualAs	$\{x_1\} \subseteq \neg \{x_2\}$	{marc} $\subseteq \neg$ {hans}
inverseOf	$P_1 \equiv P_2^{-1}$	hasChild $\equiv$ hasParent <sup>-1</sup>
TransitiveProperty	$P^+ \subseteq P$	ancestor <sup>+</sup> $\subseteq$ ancestor
FunctionalProperty	Top $\subseteq \exists \leq 1 P$	Top $\subseteq \exists \leq 1$ has Mother
InverseFunctionalProperty	Top $\subseteq \exists \leq 1 P^{-1}$	Top $\subseteq \exists \leq 1$ isMotherOf <sup>-1</sup>

Slide 48

## Open World Assumption I

hasChild(Jocasta, Oedipus)      hasChild(Jocasta, Polyneikes)  
 hasChild(Oedipus, Polyneikes)    hasChild(Polyneikes, Thersandros)  
 Patricide(Oedipus)                ¬ Patricide(Thersandros)

- Has Jocasta a child that is a patricide, which itself has a child that is **not** a patricide ?
- Entailment Problem:

$A_0 := (\exists \text{ hasChild.}(\text{Patricide} \sqcup \exists \text{ hasChild.} \neg \text{Patricide}))(Jocasta)$

Missing information:

- Polyneikes is a patricide
- Polyneikes is not a patricide

## Open World Assumption II

- Model falls in two classes
  - Polyneikes is patricide
    - is child of Polyneikes
    - has child Thersandros that is no patricide
    - **JOCASTA is answer**
  - Polyneikes is not patricide
    - Polyneikes has patricide child Oedipus
    - Oedipus has not patricide child Polyn.
    - **JOCASTA is answer**
- Open world assumption requires case analyses
- Makes reasoning more complex than query answering in databases

## DL Reasoning Services

- Semantic Web „Understanding“ closely related to reasoning
- Reasoning useful at all stages of ontology life-cycle
  - Ontology Design and maintenance
    - **Check class consistency and (unexpected) implied relationships**
    - **Particularly important with large ontologies**
  - Ontology integration
    - **Assert inter-ontology relationships**
    - **Reasoner computes integrated class hierarchy / consistency**
  - Ontology deployment
    - **Determine whether A-Box is consistent with ontology**
    - **Determine if individuals are instances of ontology classes**

## DL = Decidable Reasoning

- DL constructors/axioms restricted so reasoning is decidable
- Facilitates provision of reasoning services
  - Known algorithms
  - Implemented systems
  - Evidence of empirical tractability
- Understanding dependent on reliable & consistent reasoning
  - Sound
  - Complete

## Basic Inference Problems

- Consistency
  - Check if knowledge is meaningful
  - Is there some model I of O
  - Is there some possible I of C
- Subsumption
  - Structure knowledge
  - Compute Taxonomy
- Equivalence
  - Check if two classes denote the same set of instances
- Instantiation
  - Check if individual i is an instance of class C
- Retrieval
  - Retrieve set of individuals that instantiate C
- Problems are reducible to consistency (satisfiability):
  - $C \subseteq D$  iff  $D \sqcap \neg C$  not consistent w.r.t. O
  - $i \in C$  iff O with  $\{i\} \sqcap \neg C$  is not consistent

Slide 53

## DL Modeling Tool: OilEd

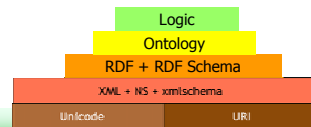
- [ [→ http://oiled.man.ac.uk/](http://oiled.man.ac.uk/) ]
- Open-source, GPL licensed
- Tutorial
  - [ [→ http://oiled.man.ac.uk/tutorial/](http://oiled.man.ac.uk/tutorial/) ]
- Single-user Modeling Support for
  - OIL
  - DAML+OIL
  - OWL

Slide 54



## Future developments: Rules

- Not in scope of WebOnt WG
- Most likely development
  - Focus on decidability
  - Ontology language must be restricted (i.e. without cardinalities)
  - „Horn fragment“-type rules must be used
  - Starting point
    - Paper by [ Grosz et al., WWW 2003 ]
- Research question
  - Prototypes exist, e.g. CARIN



Slide 55

## DL Research Challenges

- Increasing expressive power
- Scalability
  - Very Large KB
  - Reasoning with Individuals
- Other reasoning tasks
  - Querying
  - Matching
  - Least Common Subsumer (LCS)
  - Explanation
  - ...



Slide 56

## DL Resources

- DL Tutorials, Lectures etc.  
[ → [dl.kr.org](http://dl.kr.org) ]
- DL Implementations
  - Racer [ → <http://www.fh-wedel.de/~mo/racer/> ]
  - Fact [ → <http://www.cs.man.ac.uk/FaCT/> ]
  - More... [ → <http://www.ida.liu.se/labs/iislab/people/patla/DL/systems.html> ]

## Evaluation of description logics

### Advantages

- Decidable (if chosen carefully as in OWL)
- Reasoning over conceptual specification (e.g. subsumption)

### Disadvantages

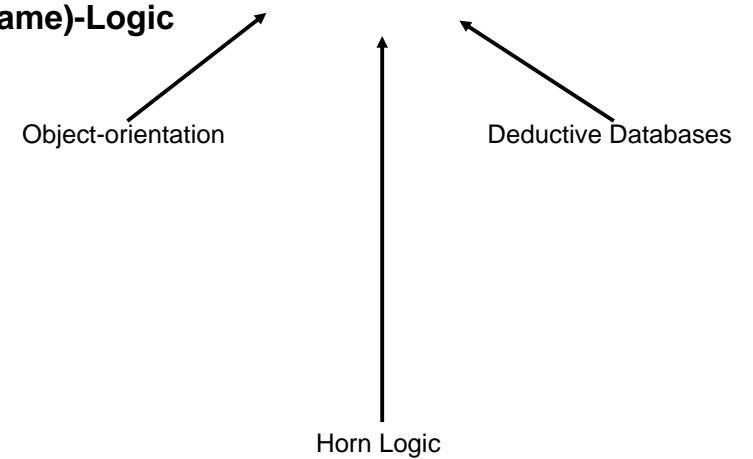
- Unusual modeling style
- Little support for computing with objects (i.e. A-Box Reasoning)

## Ontology Representation

- Predicate Logic
- Semantic Web Languages
- Description Logic
- **Frame Logic**



## F(rame)-Logic



Environment for computing with objects

## F – Logic II

- object oriented (frame based) logics
- well defined semantics (well-founded semantics)
- efficient evaluation strategy

Slide 61

## A First Example

```

/* facts */
abraham:man.
sarah:woman.
isaac:man[father->abraham; mother->sarah].
ishmael:man[father->abraham; mother->hagar:woman].
jacob:man[father->isaac; mother->rebekah:woman].
esau:man[father->isaac; mother->rebekah].
/* rules consisting of a rule head and a rule body */
FORALL X,Y X[son->>Y] <- Y:man[father->X].
FORALL X,Y X[son->>Y] <- Y:man[mother->X].
FORALL X,Y X[daughter->>Y] <- Y:woman[father->X].
FORALL X,Y X[daughter->>Y] <- Y:woman[mother->X].
/* query */
FORALL X,Y <- X:woman[son->>Y[father->abraham]].
    
```

Slide 62

## Objects, Methods

- objects are identified by unique ids

```
isaac:man.
```

- single- and multivalued methods

```
isaac:man[father->abraham; mother->sarah].
jacob[son->>{reuben, simeon, levi}].
```

- parametrised methods

```
jacob[son@(leah)->>{reuben, simeon};
      son@(rachel)->>{joseph, benjamin}].
```

Slide 63

## Class Membership and Subclass Relationship

```
abraham:man.
isaac:man.
sarah:woman.
```

```
woman::person.
man::person.
```

Slide 64

## F-Molecule

- an F-Molecule:  
isaac:man[father->abraham; son->>{jacob,esau}].
- may be split into several F-atoms:  
isaac:man.  
isaac[father->abraham].  
isaac[son->>jacob].  
isaac[son->>esau].

Slide 65

## Signatures

- define ranges of the methods

```
person[father=>man].  
person[daughter=>>woman].  
man[son@(woman)=>>man].  
person[father=>{man, person}].
```

Slide 66

## Nesting

```
isaac[father->abraham:man  
[son@(hagar:woman)->>ishmael];  
mother->sarah:woman].
```

```
jacob:(man::person).  
jacob[(father:method)->isaac].
```

Slide 67

## Predicates

```
married(isaac,rebekah).
```

```
male(jacob).
```

```
sonof(isaac,rebekah,jacob).
```

```
true.
```

Slide 68

## Path Expressions

```
isaac:man[son->> {jacob,esau}].
jacob[son@(rachel,11)->>{joseph, benjamin};
  son@(zilpah,12)->>{gad, asher};
  son@(bilhah,13)->>{dan, naphtali}].
```

```
isaac..son                {jacob,esau}
jacob..son@(rachel,11)    {joseph,benjamin}
benjamin.father.father.mother
```

Slide 69

## Rules

```
FORALL X,Y X[ancestor->>Y] <- X[father->Y].

FORALL X,Y X[ancestor->>Y] <- X[mother->Y].

FORALL X,Y,Z X[ancestor->>Y]
  <- X[father->Z] AND Z[ancestor->>Y].

FORALL X,Y,Z X[ancestor->>Y]
  <- X[mother->Z] AND Z[ancestor->>Y].
```

Slide 70

## Queries

Queries are rules with an empty head  
(remember Prolog ?!...)

The answer to a query consists of all variable bindings such that the corresponding groundinstance of the rule body is true in the object base.

→ Unlike DL: Closed-world assumption !

Example:

```
FORALL Y <- jacob[ancestor->>Y:woman].
```

Result bindings:

**Y = rebekah**

**Y = sarah**

Slide 71

## Quantification

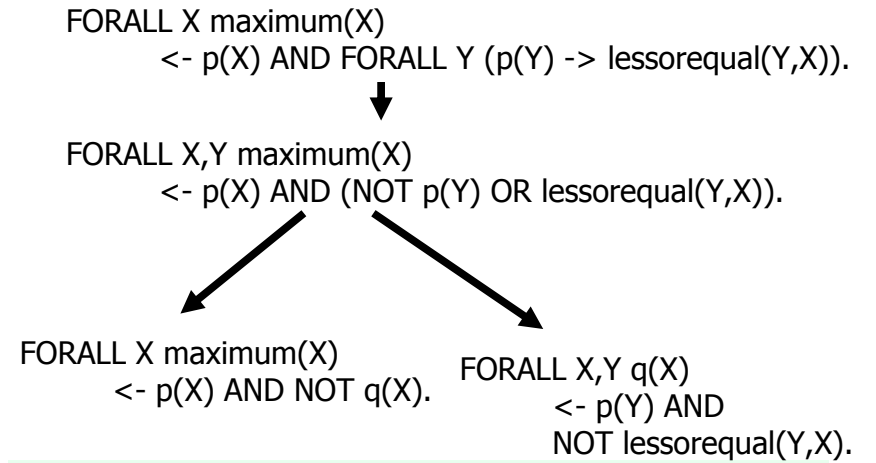
- **quantification may range over classes::**
  - FORALL X,Y <- X::Y.
  - FORALL X,R <- X[name=>R].
  - FORALL A,R <- Person[A=>R].

Slide 72

## Negation

```
FORALL X,Y
  X[notrelated->>Y] <-
    X:person AND
    Y:person AND
    NOT X[ancestor->>Y] AND
    NOT Y[ancestor->>X].
```

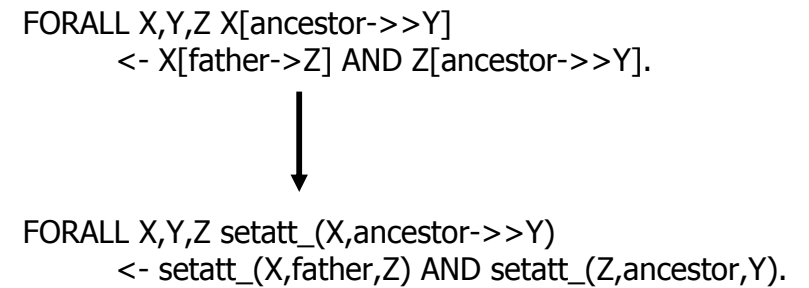
## Lloyd-Topor Transformation



## Transforming F-Logics to Horn logics

a:C	→	isa_(a,C)
A::B	→	sub_(A,B)
A[B=>C]	→	atttype_(A,B,C)
A[B=>>C]	→	setatttype_(A,B,C)
a[B->c]	→	att_(a,B,c)
a[B->>c]	→	setatt_(a,B,c)

## Transforming F-Logics to Horn logics



## Extensions of F-Logic

### General

- For path expressions
- FOL-style predicate symbols

### Proprietary

- For built-ins
- For database connectors
- For namespaces

## Naive Evaluation

C-name	P-name
Mary Fox	Linda Fox
June Fox	Linda Fox
Bill Fox	Linda Fox
Mary Fox	John Fox
June Fox	John Fox
Bill Fox	John Fox
John Hunt	Mary Fox
Jack Hunt	Mary Fox
Helen Kent	June Fox
Dean Kent	June Fox

ancestor(X, Y) :- parent(X, Y).  
 ancestor(X, Y) :-  
     ancestor(X, Z)  
     ancestor(Z, Y)

## Naive Evaluation

ancestor(Y, X) :- ancestor(Z, X), ancestor(Y, Z)

C-name	P-name
Mary Fox	Linda Fox
John Fox	Peter Fox
Bill Fox	Linda Fox
Mary Fox	John Fox
Peter Fox	Mike Fox
Bill Fox	John Fox
John Hunt	Mary Fox
Jack Hunt	Mary Fox
Helen Kent	June Fox
Dean Kent	June Fox
Jack Hunt	Linda Fox
John Hunt	Linda Fox
Bill Fox	Peter Fox
Mary Fox	Peter Fox

=

C-name	P-name
Mary Fox	Linda Fox
John Fox	Peter Fox
Bill Fox	Linda Fox
Mary Fox	John Fox
Peter Fox	Mike Fox
Bill Fox	John Fox
John Hunt	Mary Fox
Jack Hunt	Mary Fox
Helen Kent	June Fox
Dean Kent	June Fox



C-name	P-name
Mary Fox	Linda Fox
John Fox	Peter Fox
Bill Fox	Linda Fox
Mary Fox	John Fox
Peter Fox	Mike Fox
Bill Fox	John Fox
John Hunt	Mary Fox
Jack Hunt	Mary Fox
Helen Kent	June Fox
Dean Kent	June Fox

## Naive Evaluation

ancestor(Y, X) :- ancestor(Z, X), ancestor(Y, Z)

C-name	P-name
Mary Fox	Linda Fox
John Fox	Peter Fox
Bill Fox	Linda Fox
Mary Fox	John Fox
Peter Fox	Mike Fox
Bill Fox	John Fox
John Hunt	Mary Fox
Jack Hunt	Mary Fox
Helen Kent	June Fox
Dean Kent	June Fox
Jack Hunt	Linda Fox
John Hunt	Linda Fox
Bill Fox	Peter Fox
Mary Fox	Peter Fox
Bill Fox	Mike Fox
Mary Fox	Mike Fox
John Fox	Mike Fox

C-name	P-name
Mary Fox	Linda Fox
John Fox	Peter Fox
Bill Fox	Linda Fox
Mary Fox	John Fox
Peter Fox	Mike Fox
Bill Fox	John Fox
John Hunt	Mary Fox
Jack Hunt	Mary Fox
Helen Kent	June Fox
Dean Kent	June Fox
Jack Hunt	Linda Fox
John Hunt	Linda Fox
Bill Fox	Peter Fox
Mary Fox	Peter Fox
John Fox	Peter Fox

## Naive Evaluation

$\text{ancestor}(Y, X) :- \text{ancestor}(Z, X), \text{ancestor}(Y, Z)$

C-name	P-name
Mary Fox	Linda Fox
John Fox	Peter Fox
Bill Fox	Linda Fox
Mary Fox	John Fox
Peter Fox	Mike Fox
Bill Fox	John Fox
John Hunt	Mary Fox
Jack Hunt	Mary Fox
Helen Kent	June Fox
Dean Kent	June Fox
Jack Hunt	Linda Fox
John Hunt	Linda Fox
Bill Fox	Peter Fox
Mary Fox	Peter Fox
Bill Fox	Mike Fox
Mary Fox	Mike Fox
John Fox	Mike Fox

C-name	P-name
Mary Fox	Linda Fox
John Fox	Peter Fox
Bill Fox	Linda Fox
Mary Fox	John Fox
Peter Fox	Mike Fox
Bill Fox	John Fox
John Hunt	Mary Fox
Jack Hunt	Mary Fox
John Hunt	Mary Fox
Jack Hunt	Mary Fox
Helen Kent	June Fox
Dean Kent	June Fox
Jack Hunt	Linda Fox
John Hunt	Linda Fox
Jack Hunt	Linda Fox
John Hunt	Linda Fox
Bill Fox	Peter Fox
Mary Fox	Peter Fox
John Hunt	Linda Fox
Bill Fox	Peter Fox
Mary Fox	Peter Fox

$:-\text{ancestor}(\text{"Mary Fox"}, \text{"Mike Fox"}, ) \longrightarrow \text{yes}$

Slide 81

## Naive Evaluation

- computes minimal model
- fast join operations
- computes the entire model  
does not consider constants in query

Slide 82

## Problem Negation

$\text{roman}(\text{caesar}) \wedge$   
 $(\forall x) (\text{roman}(x) \wedge \neg \text{pomp}(x) \Rightarrow \text{enemy}(x))$

≡

$\text{roman}(\text{caesar}) \wedge$   
 $(\forall x) (\neg \text{roman}(x) \vee \text{pomp}(x) \vee \text{enemy}(x))$

more than one smallest model:

$M1 = \{\text{roman}(\text{cäsar}), \text{pomp}(\text{caesar})\}$

$M2 = \{\text{roman}(\text{cäsar}), \text{enemy}(\text{caesar})\}$

Slide 83

## Stratified Semantics

$\text{roman}(\text{caesar}) \wedge \text{livesin}(\text{marcus}, \text{Pompei}) \wedge$   
 $(\forall x) (\text{livesin}(x, \text{Pompei}) \Rightarrow \text{pomp}(x))$

Stratum 0

$(\forall x) (\text{roman}(x) \wedge \neg \text{pomp}(x) \Rightarrow \text{enemy}(x))$

Stratum 1

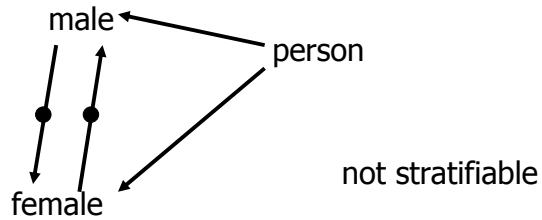
$M = \{\text{roman}(\text{caesar}), \text{livesin}(\text{marcus}, \text{Pompei}), \text{pomp}(\text{marcus}), \text{enemy}(\text{caesar})\}$

Slide 84

What's about programs like

```

person(somebody).
male(X) :- person(X),not female(X).
female(X) :- person(X), not male(X).
    
```



Operator T

$$T(S) = \{ \sigma(C) \mid C: -A_1, \dots, A_n \text{ is a rule and } A_i \text{ is positive and } \sigma(A_i) \text{ is in } S \text{ and } A_j \text{ is negative and } \sigma(A_j) \text{ is not in } S \}$$

Example

```

person(somebody).
male(X) :- person(X),not female(X).
female(X) :- person(X), not male(X).
    
```

$$T(\{\}) = \{ \text{person(somebody)} \}$$

Repeated Application of T

```

person(somebody).
male(X) :- person(X),not female(X).
female(X) :- person(X), not male(X).
    
```

$$T(\{\text{person(somebody)}\}) = \{ \text{person(somebody), male(somebody), female(somebody)} \}$$

$$T(\{\text{person(somebody), male(somebody), female(somebody)}\}) = \{ \text{person(somebody)} \}$$

Alternating Fix Point

```

person(somebody).
male(X) :- person(X),not female(X).
female(X) :- person(X), not male(X).
    
```

$$T(S_1) = S_2 \quad \text{with } S_1 \text{ subset of } S_2$$

$$T(S_2) = S_1$$

## Wellfounded Semantics

person(somebody).  
male(X) :- person(X), not female(X).  
female(X) :- person(X), not male(X).

$T(S1) = S2$

with  $S1$  subset of  $S2$

$T(S2) = S1$

$S1$  are the true facts  
 $S2-S1$  are unknown facts  
all other facts of HB are false

Slide 89

## Properties Wellfounded Model

- in case of pure Horn logics the wellfounded model equals the minimal model
- in case of stratified programs the wellfounded model equals the stratified model (perfect model)
- it defines a model for all normal programs

Slide 90

## Evaluation of F-Logic

### Advantages

- Expressive mechanism for reasoning with instances
- Object-oriented view
- Smooth integration with relational and/or OO databases

### Disadvantages

- No classical model theory
- Different semantics may be applicable (well-founded, stable-model, stratified....)

Slide 91

## F-Logic Resources

- Documentation
  - Original Paper  
[ → <http://citeseer.nj.nec.com/kifer90logical.html> ]
  - Tutorial  
[ → <http://citeseer.nj.nec.com/frohn97how.html> ]
- Implementations
  - OntoBroker (has GUI Editor, commercial)  
[ → <http://www.ontoprise.com/> ]
  - Flora, XSB Pre-Compiler  
[ → <http://flora.sourceforge.net/> ]
  - Florid  
[ → <http://www.informatik.uni-freiburg.de/~dbis/florid/> ]

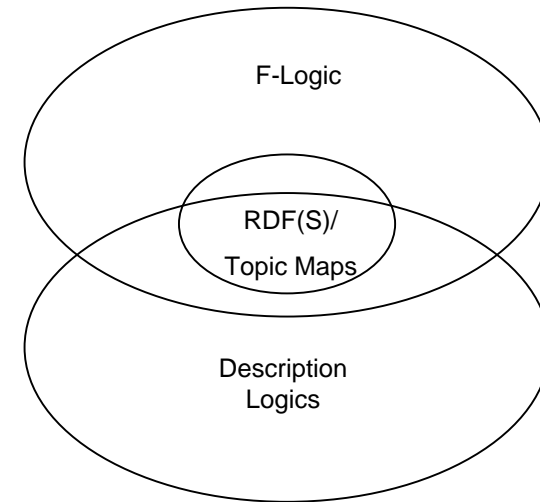
Slide 92

## Summary of Strengths

Web Orientation	RDF(S) OWL
Consistency Checking	Description Logics
Reasoning with Instances	F-Logic
Navigation	Topic Maps

Slide 93

## Inferential Expressivity



Slide 94

## Ontologies - Some Examples

- General purpose ontologies:
  - WordNet / EuroWordNet, <http://www.cogsci.princeton.edu/~wn>
  - The Upper Cyc Ontology, <http://www.cyc.com/cyc-2-1/index.html>
  - IEEE Standard Upper Ontology, <http://suo.ieee.org/>
- Domain and application-specific ontologies:
  - RDF Site Summary RSS, <http://groups.yahoo.com/group/rss-dev/files/schema.rdf>
  - UMLS, <http://www.nlm.nih.gov/research/umls/>
  - KA2 / Science Ontology, <http://ontobroker.semanticweb.org/ontos/ka2.html>
  - RETSINA Calendaring Agent, <http://ilrt.org/discovery/2001/06/schemas/ical-full/hybrid.rdf>
  - AIFB Web Page Ontology, <http://ontobroker.semanticweb.org/ontos/aifb.html>
  - Web-KB Ontology, <http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/>
  - Dublin Core, <http://dublincore.org/>
- Meta-Ontologies
  - Semantic Translation, <http://www.ecimf.org/contrib/onto/ST/index.html>
  - RDFT, <http://www.cs.vu.nl/~borys/RDFT/0.27/RDFT.rdfs>
  - Evolution Ontology, <http://kaon.semanticweb.org/examples/Evolution.rdfs>
- Ontologies in a wider sense
  - Agrovoc, <http://www.fao.org/agrovoc/>
  - Art and Architecture, <http://www.getty.edu/research/tools/vocabulary/aat/>
  - UNSPSC, <http://eccma.org/unspsc/>
  - DTD standardizations, e.g. HR-XML, <http://www.hr-xml.org/>



Slide 95