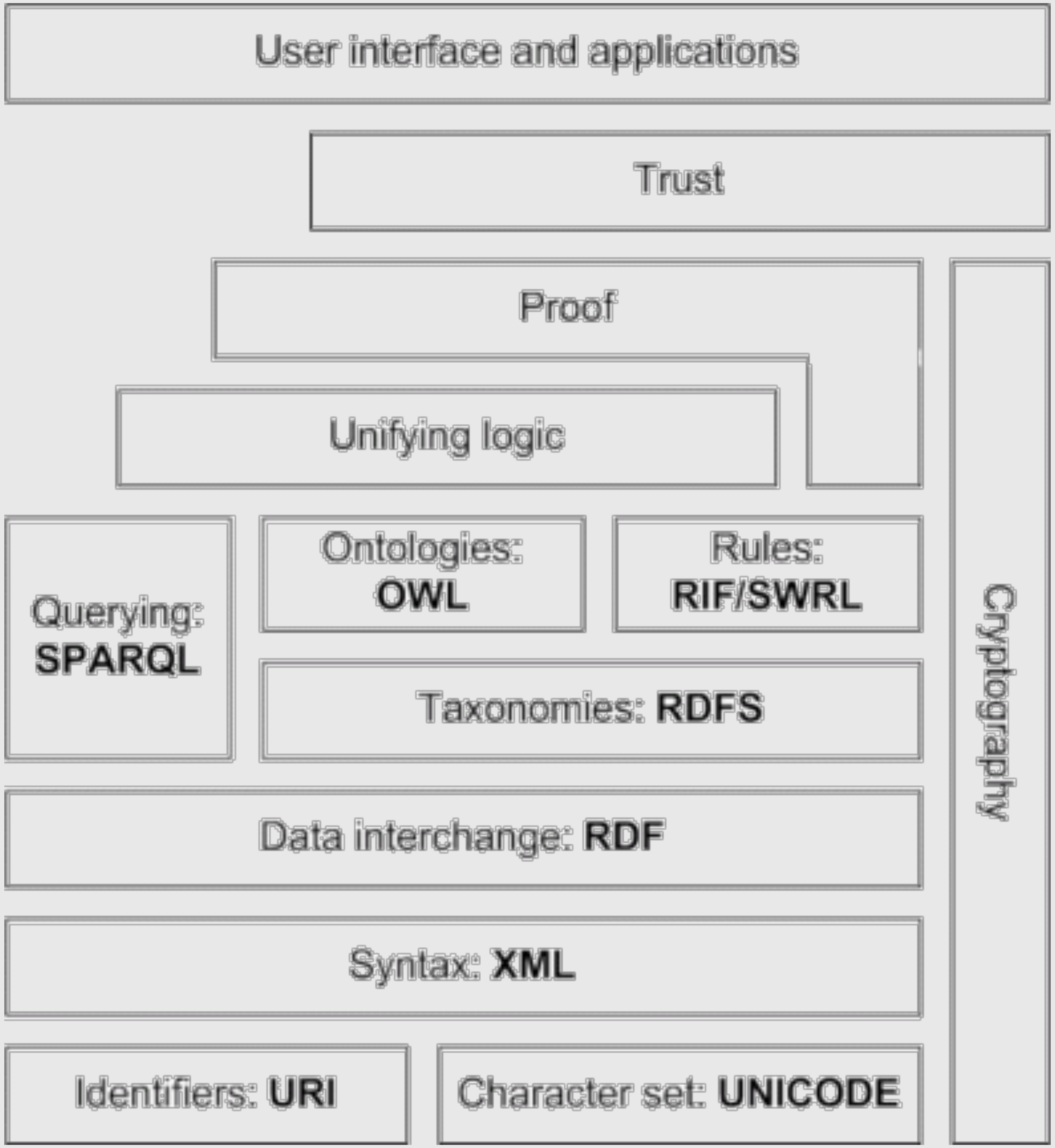
Languages

Aldo Gangemi
Valentina Presutti

The Semantic Web Layers

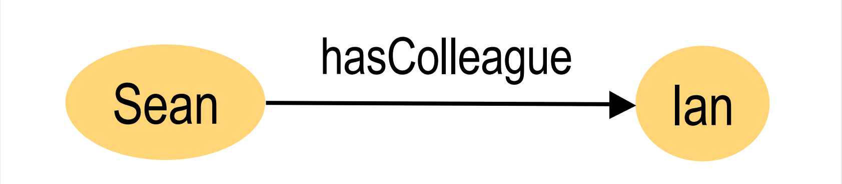


RDF

- RDF stands for Resource Description Framework
- It is a W3C Recommendation
  - http://www.w3.org/RDF
- RDF is a graphical formalism (+ XML syntax + semantics)
  - for representing metadata
  - for describing the semantics of information in a machine-accessible way
- Provides a simple data model based on triples.

RDF Data Model

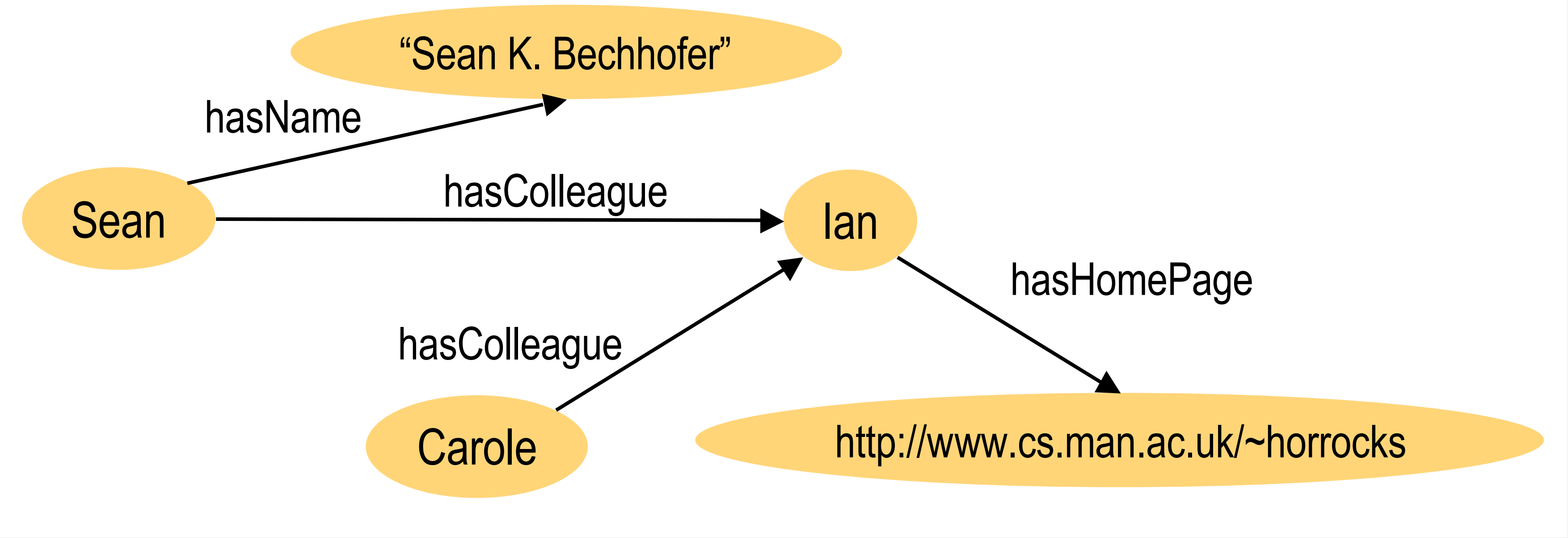
- Statements are <subject, predicate, object> triples:
  - <Sean,hasColleague,Ian>
- Can be represented as a graph:



- Statements describe properties of resources
- A resource is any object that can be pointed to by a URI:
  - The generic set of all names/addresses that are short strings that refer to resources
  - a document, a picture, a paragraph on the Web, http://www.cs.man.ac.uk/index.html, a book in the library, a real person (?), isbn://0141184280
- Properties themselves are also resources (URIs)

Linking Statements

- The subject of one statement can be the object of another
- Such collections of statements form a directed, labeled graph



- Note that the object of a triple can also be a "literal" (a string)

What does RDF give us?

- A mechanism for annotating data and resources.
- Single (simple) data model.
- Syntactic consistency between names (URIs).
- Low level integration of data.

RDF(S): RDF Schema

- RDF gives a formalism for meta data annotation, and a way to write it down in XML, but it does not give any special meaning to vocabulary such as subClassOf or type

- Interpretation is an arbitrary binary relation

- RDF Schema extends RDF with a schema vocabulary that allows you to define basic vocabulary terms and the relations between those terms
  - Class, type, subClassOf,
  - Property, subPropertyOf, range, domain
  - it gives “extra meaning” to particular RDF predicates and resources
  - this “extra meaning”, or semantics, specifies how a term should be interpreted

RDF/RDF(S) “Liberality”

- No distinction between classes and instances (individuals)
- Properties can themselves have properties
- No distinction between language constructors and ontology vocabulary, so constructors can be applied to themselves/each other

What does RDF(S) give us?

- Ability to use simple schema/vocabularies when describing our resources.
- Consistent vocabulary use and sharing.
- Simple inference

Need for a web ontology language

- But RDFS not a suitable foundation for Semantic Web
  - Too weak to describe resources in sufficient detail
- Requirements for web ontology language:
  - Compatible with existing Web standards (XML, RDF, RDFS)
  - Easy to understand and use (based on familiar KR idioms)
  - Formally specified and of “adequate” expressive power
  - Possible to provide automated reasoning support

Some premises to web ontology design

- Some recall of logics: propositional logics, first order logic, and description logics
- Web languages: RDF and OWL

- References:
  - Tutorial on Description Logics - Enrico Franconi:
  - Tutorial on Ontology Languages for the Semantic Web - Ian Horrocks and Sean Bechhofer:
  - Tutorial on Semantic Web Best Practices - Alan Rector:

About Logic

- It allows us to represent information about a domain in a very straight-forward way then deduce additional facts using one general domain-independent ”algorithm”: deduction.
- It lends itself to large-scale, distributed-design problems.
- Each logic is made up of a syntax, a semantics, a definition of the reasoning problems and the computational properties, and inference procedures for the reasoning problems (possibly sound and complete).
- The syntax describes how to write correct sentences in the language.
- The semantics tells us what sentences mean according to an interpretation function over a “domain”.
- The inference procedure derives results logically implied by a set of premises.

Formal languages: logics

- *Logics* are formal languages for representing information such that conclusions can be drawn.
- *Syntax* defines the sentences in the language.
- *Semantics* defines the “meaning” of sentences; i.e., defines truth of a sentence in a world.

- E.g., the language of arithmetic

  - $x + 2 \geq y$ is a sentence; $x2 + y >$ is not a sentence
  - $x + 2 \geq y$ is true iff the number $x + 2$ is no less than the number y
  - $x + 2 \geq y$ is true in a world where $x = 7$; $y = 1$
  - $x + 2 \geq y$ is false in a world where $x = 0$; $y = 6$
  - $x + 2 \geq x + 1$ is true in every world

- Logics differ in terms of their representation power and computational complexity of inference.
- The more restricted the representational power, the faster the inference in general.

The *one and only* logic?

- Logics of higher order
- Modal logics
  - epistemic
  - temporal and spatial
  - ...
- Description logic
- Non-monotonic logic
- Intuitionistic logic
- ...

But: there are "standard approaches":

⇒ **propositional and predicate logic**

Types of logic

- Logics are characterized by what they commit to as “primitives”
- Ontological commitment: what exists—facts? objects? time? beliefs
- Epistemological commitment: what states of knowledge?

Language	Ontological Commitment (What exists in the world)	Epistemological Commitment (What an agent believes about facts)
Propositional logic	Facts	True/False/Unknown
First-order logic	Facts, objects, relations	True/False/Unknown
Temporal logic	Facts, objects, relations, times	True/False/Unknown
Probability theory	Facts	Degree of beliefs 0...1
Fuzzy logic	Degree of truth	Degree of beliefs 0...1

Classical logics are based on the notion of TRUTH

Entailment - Logical implication

- Knowledge Base KB *entails* sentence *a*

  if and only if

  a is true in all worlds where KB is true

  - E.g., the KB containing “Roma won” and “Lazio won” *entails* “Either Roma won or Lazio won”

Propositional Logic

- We can only talk about facts and whether or not they are true.
- In the worst case, we can use the brute force truth-table method to do inference.
- Proof methods such as tableaux are generally more efficient, easier to implement, and easier to understand.

Propositional Logics: basic ideas

- Statements
  - The elementary building blocks of propositional logic are atomic statements that cannot be decomposed any further: propositions. E.g.,
    - “The block is red”
    - “The proof of the pudding is in the eating”
    - “It is raining”
  - and logical connectives “and”, “or”, “not”, by which we can build **propositional formulas**.
- Reasoning
  - when is a statement **logically implied** by a set of statements?
- Semantics: intuition
  - Atomic statements can be *true* T or *false* F
  - The truth value of formulas is determined by the truth value of the atoms

First Order Logic

- We can already do a lot with propositional logic.
- But it is unpleasant that we cannot access the structure of atomic sentences.
- Atomic formulas of propositional logic are too atomic – they are just statement which my be true or false but which have no internal structure.
- In First Order Logic (FOL) the atomic formulas are interpreted as statements about relationships between objects.
- We can talk about objects and relations between them, and we can quantify over objects.
- Good for representing most interesting domains, but inference is not only expensive, but may not terminate.

Predicate and Constants

- Let's consider the statements:
  - Mary is female
  - John is male
  - Mary and John are siblings
- In propositional logic the above statements are atomic propositions:
  - Mary-is-female
  - John-is-male
  - Mary-and-John-are-siblings
- In FOL atomic statements use predicates, with constants as argument:
  - Female(mary)
  - Male(john)
  - Siblings(mary,john)

Variables and Quantifiers

- Let's consider the statements:
  - Everybody is male or female
  - A male is not a female
- In FOL predicates may have variables as arguments, whose value is bounded by quantifers:
  - $\forall$ x. Male(x) $\vee$ Female(x)
  - $\forall$ x. Male(x) $\Rightarrow$ $\neg$ Female(x)
- Deduction (why?):
- Mary is not male
- *Not* Male(mary)

Functions

- Let's consider the statement:
  - The father of a person is male
- In FOL objects of the domain may be denoted by functions applied to (other)objects:
  - $\forall$ x. Male(father(x))

Semantics of FOL: intuition

- Just like in propositional logic, a (complex) FOL formula may be true (or false)with respect to a given interpretation.

- An interpretation specifies referents for
  - constant symbols --> objects
  - predicate symbols --> relations
  - function symbols --> functional relations

- An atomic sentence $P(t_1;....;t_n)$ is true in a given interpretation
  iff the objects referred to by $t_1;....;t_n$
  are in the relation referred to by the predicate P.

- An interpretation in which a formula is true is called a model for the formula.

Universal quantification

- Everyone in England is smart:
  - $\forall x(In(x, england) \Rightarrow Smart(x))$
- $(\forall x(\phi))$ is equivalent to the conjunction of all possible instantiations in x of ϕ:
  - $In(kingJohn, england) \Rightarrow Smart(kingJohn)$
  - $\wedge\ In(richard, england) \Rightarrow Smart(richard)$
  - $\wedge\ In(england, england) \Rightarrow Smart(england)$
  - $\wedge$ ...
- Typically, $\Rightarrow$ is the main connective with $\forall$.
- Common mistake: using $\wedge$ as the main connective with $\forall$:
  $\forall x(In(x, england) \wedge Smart(x))$
  means "Everyone is in England and everyone is smart"

Existential quantification

- Someone in France is smart:
  - $\exists x(In(x, france) \wedge Smart(x))$
- ($\exists x(\phi)$) is equivalent to the disjunction of all possible instantiations in x of ϕ
  - $In(kingJohn, france) \wedge Smart(kingJohn)$
  - $\vee\ In(richard, france) \wedge Smart(richard)$
  - $\vee\ In(france, france) \wedge Smart(france)$
  - $\vee$ ...
- Typically, $\wedge$ is the main connective with $\exists$.
- Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

  $\exists x(In(x, france)) \Rightarrow Smart(x)$

  is true if there is anyone who is not in France!

Introduction to Description Logics

Why Description Logics?

- If predicate logic is directly used without some kind of restriction, then the expressive power is too high for having good computational properties and efficient procedures.

What are Description Logics

- A family of logic based Knowledge Representation formalisms
  - Descendants of Semantic Networks, Minsky's frames, and KL-ONE
  - Describe domain in terms of **concepts** (classes), **roles**(relationships) and **individuals**

- Distinguished by
  - **Formal semantics** (model theoretic)
    - Decidable fragments of FOL
    - Closely related to Propositional Modal & Dynamic Logics
  - Provision of **inference services**
    - Sound and complete decision procedures for key problems
    - Implemented systems (highly optimized)

A pragmatist’s view of the history of Description Logics

- Informal Semantic Networks and Frames (pre 1980)
  - Wood: What’s in a Link; Brachman What IS-A is and IS-A isn’t.
- First Formalisation (1980)
  - Bobrow KRL, Brachman: KL-ONE
- All useful systems are intractable (1983)
  - Brachman & Levesque: A fundamental tradeoff
    - Hybrid systems: T-Box and A-Box
- All tractable systems are useless (1987-1990)
  - Doyle and Patel: Two dogmas of Knowledge Representation

Short history of Description Logics

- Phase 1
  - Incomplete systems (Back, Classic, Loom, . . .)
  - Based on **structural algorithms**
- Phase 2
  - Development of **tableau algorithms** and **complexity results**
  - Tableau-based systems (Kris, Crack)
  - Investigation of optimization techniques
- Phase 3
  - Tableau algorithms for **very expressive** DLs
  - **Highly optimised** tableau systems (FaCT, DLP, Racer)
  - Relationship to modal logic and decidable fragments of FOL

Latest developments

- Phase 4
  - Mature **implementations**
  - Mainstream **applications** and Tools
    - Databases
    - Consistency of conceptual schemata (EER, UML etc.)
    - Schema integration
    - Query subsumption (w.r.t. a conceptual schema)
  - Ontologies and **Semantic Web** (and **Grid**)
    - Ontology engineering (design, maintenance, integration)
    - Reasoning with ontology-based markup (meta-data)
    - Service description and discovery
  - **Commercial** implementations
    - Cerebra system from Network Inference Ltd

DL Semantics

- **Model theoretic** semantics. An interpretation consists of
  - A domain of discourse (a collection of objects)
  - Functions mapping
    - classes to set of objects
    - properties to sets of pairs of objects
  - Rules describe how to interpret the constructors and tell us when an interpretation is a model.
- In DL, a class description is thus a characterization of the individuals that are members of that class.

OWL Layering

- Three species of OWL
  - OWL full is union of OWL syntax and RDF
  - OWL DL restricted to FOL fragment (~ DAML+OIL)
    - Corresponds to *SHOIN*(D_n) Description Logicc
  - OWL Lite is “simpler” subset of OWL DL

- Semantic layering
  - OWL DL semantics = OWL full semantics within DL fragment
  - OWL Lite semantics = OWL DL semantics within Lite fragment

- DL semantics are **definitive**
  - In principle: correspondence proof
  - But: if Full disagrees with DL (in DL fragment), then Full is wrong

OWL Full

- No restriction on use of OWL vocabulary (as long as legal RDF)
  - Classes as instances (and much more)

- RDF style model theory
  - Reasoning using FOL engines
    - via axiomatization
  - Semantics should correspond with OWL DL for suitably restricted KBs

OWL DL

- Use of OWL vocabulary restricted
  - Cannot be used to do “nasty things” (i.e., modify OWL)
  - No classes as instances
  - Defined by abstract syntax + mapping to RDF
- Standard DL/FOL model theory (definitive)
  - Direct correspondence with (first order) logic
- Benefits from many years of DL research
  - Well defined semantics
  - Formal properties well understood (complexity, decidability)
  - Known reasoning algorithms
  - Implemented systems (highly optimized)

OWL Lite

- Like DL, but fewer constructs
  - No explicit negation or union
  - Restricted cardinality (zero or one)
  - No nominals (oneOf)
- Semantics as per DL
  - Reasoning via standard DL engines (+datatypes)
    - E.g., FaCT, RACER, Cerebra, Pellet
- In practice, not really used.
  - Possible alternative: ”tractable fragments”

OWL syntaxes

- Abstract syntax
  - Used in the definition of the language and the DL/Lite semantics
- OWL in RDF (the “official” concrete syntax)
  - RDF/XML presentation
- XML presentation syntax
  - XML Schema definition

OWL DL Semantics

- Semantics defined by interpretations: $I = (\Delta^I, \cdot^I)$
  - *I:concepts → subset of* Δ^I
  - *I:properties → binary relations over* Δ^I *(subsets of* $\Delta^I \times \Delta^I$*)*
  - *I:individuals → elements of* Δ^I
- *Interpretation function* $\cdot^I$ *extended* to concept expressions

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \quad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}} \quad (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$$

$$\{x_n, \ldots, x_n\}^{\mathcal{I}} = \{x_n^{\mathcal{I}}, \ldots, x_n^{\mathcal{I}}\}$$

$$(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y. \langle x, y \rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$$

$$(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y. (x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$$

$$(\leqslant nR)^{\mathcal{I}} = \{x \mid \#\{y \mid \langle x, y \rangle \in R^{\mathcal{I}}\} \leqslant n\}$$

$$(\geqslant nR)^{\mathcal{I}} = \{x \mid \#\{y \mid \langle x, y \rangle \in R^{\mathcal{I}}\} \geqslant n\}$$

OWL Class Constructors

Constructor	Example	Interpretation
Classes	Human	*I(Human)*
intersectionOf	intersectionOf(Human Male)	*I(Human)* ∩ *I(Male)*
unionOf	unionOf(Doctor Lawyer)	*I(Doctor)* ∪ *I(Lawyer)*
complementOf	complementOf/Male)	*Δ \ I(Male)*
oneOf	oneOf(john mary)	*{I(john), I(mary)}*

OWL CLass Constructors

Constructor	Example	Interpretation
someValuesFrom	restriction(hasChild someValuesFrom Lawyer)	$\{x \mid \exists y.\ \langle x,y \rangle \in \mathit{I(hasChild)} \wedge y \in \mathit{I(Lawyer)}\}\}$
allValuesFrom	restriction(hasChild allValuesFrom Doctor)	$\{x \mid \forall y.\ \langle x,y \rangle \in \mathit{I(hasChild)} \Rightarrow y \in \mathit{I(Doctor)}\}\}$
minCardinality	restriction(hasChild minCardinality(2)))	$\{x \mid \#\ \langle x,y \rangle \in \mathit{I(hasChild)} \geq 2\}$
maxCardinality	restriction(hasChild maxCardinality (2))	$\{x \mid \#\ \langle x,y \rangle \in \mathit{I(hasChild)} \leq 2$

OWL Axioms

- Axioms allow us to add further statements about arbitrary concept expressions and properties
  - Subclasses, Disjointness, Equivalence, transitivity of properties, etc.
- An interpretation is then a model of the axiom iff it satisfies every axiom in the model.

Axiom	Example	Interpretation
SubClassOf	SubClassOf(Human Animal)	$I(Human) \subseteq I(Animal)$
EquivalentClasses	EquivalentClass(Man intersectionOf(Human Male))	$I(Man) = I(Human) \cap I(Male)$
DisjointClasses	DisjointClass(Animal Plant)	$I(Animal) \cap I(Plant) = \emptyset$

OWL Individual Axioms

Axiom	Example	Interpretation
Individual	Individual(Valentina Type(Human))	*I(Valentina)* ∈ *I(Human)*
Fact	Individual(Valentina value(worksWith Aldo))	*I* 〈*Valentina,Aldo*〉 ∈ *I(worksWith)*
DifferentIndividuals	DifferentIndividuals(Valentina Aldo)	*I(Valentina)* ≠ *I(Aldo)*
SameIndividualAs	SameIndividualAs(AldoGangemi GangemiAsTutor)	*I(*AldoGangemi*)* = *I(*GangemiAsTutor*)*

OWL Property Axioms

Axiom	Example	Interpretation
SubPropertyOf	SubPropertyOf(hasMother hasParent)	$I(hasMother) \subseteq I(hasParent)$
domain	ObjectProperty(owns domain(Person))	$\forall x.\ \langle x,y \rangle \in I(owns) \Rightarrow x \in I(Person)$
range	ObjectProperty(employs range(Person))	$\forall x.\ \langle x,y \rangle \in I(employs) \Rightarrow y \in I(Person)$
symmetric	ObjectProperty(connects Symmetric)	$\forall x,y.\ \langle x,y \rangle \in I(connects) \Rightarrow \langle y,x \rangle \in I(connects)$
transitive	ObjectProperty(hasPart Transitive)	$\forall x,y,z.\ \langle x,y \rangle \in I(hasPart) \wedge \langle y,z \rangle \in I(hasPart) \Rightarrow \langle x,z \rangle \in I(hasPart)$
inverseOf	ObjectProperty(hasChild inverseOf(hasParent))	$I(hasChild) = I(hasParent^{-})$

XML Datatypes in OWL

- OWL supports XML Schema primitive datatypes
- Clean separation between "object" classes and datatypes
  - Disjoint interpretation domain: $d^I \subseteq \Delta_D$, and $\Delta_D \cap \Delta^I = \varnothing$
  - Disjoint datatype properties: $P^I_D \subseteq \Delta^I \times \Delta_D$
- Philosophical reasons:
  - Datatypes structured by built-in predicates
  - Not appropriate to form new datatypes using ontology language
- Practical reasons:
  - Ontology language remains simple and compact
  - Semantic integrity of ontology language not compromised
  - Implementability not compromised — can use hybrid reasoner
    - Only need sound and complete decision procedure for $d^I_1 \cap \ldots \cap d^I_n$, where d^I_i is a (possibly negated) datatype

Semantics

- An interpretation *I* satisfies an axiom if the interpretation of the axiom is true.
- *I* satisfies ontology or is a model of an ontology *O* (is a model of *O*) iff it satisfies every axiom in *O*
- *C* subsumes *D* w.r.t. an ontology *O* iff for every model *I* OF *O*, $I(D) \subseteq I(C)$
- *C* is equivalent to *D* w.r.t an ontology *O* iff for every model *I* of *O*, $I(C) = I(D)$
- *C* is satisfiable w.r.t. *O* iff there exists some model *I* of *O* s.t. $I(C) \neq O$
- An ontology *O* is consistent iff there exists some model *I* of *O*