Semantic Technologies for Intelligence, Defense, and Security (STIDS) 2011 Tutorial:

Introduction to Ontologies and Semantic Technologies

Dr. Leo Obrst
Information Semantics
Information Discovery & Understanding
Command & Control Center
MITRE
Lobrst@mitre.org
November 15, 2011

Overview

- The initial segment of this course introduces Ontologies and Semantic Technologies. It first describes the difference between Syntax and Semantics, and then looks at various definitions of Ontology, and describes the Ontology Spectrum and the range of Semantic Models
- The second segment focuses on Logic, the foundation of ontologies and knowledge representation, and then describes logical Ontologies and the Semantic Web languages and technologies

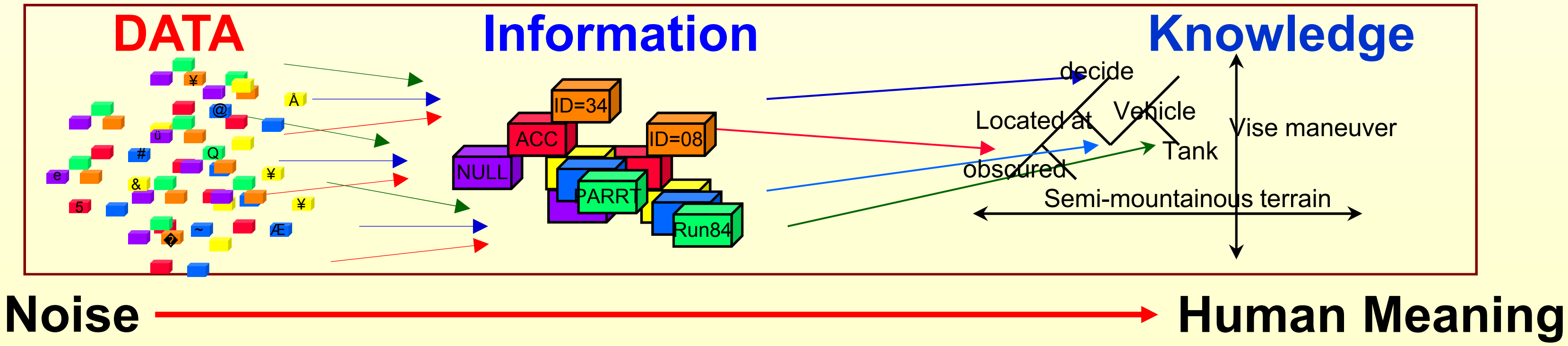
Brief Definitions:

- **Semantics:** **Meaning and the study of meaning**
- **Semantic Models:** **The *Ontology Spectrum:* Taxonomy, Thesaurus, Conceptual Model, Logical Theory, the range of models in increasing order of semantic expressiveness**
- **Ontology:** **An ontology defines the terms used to describe and represent an area of knowledge (subject matter)**
- **Knowledge Representation:** **A sub-discipline of AI addressing how to represent human knowledge (conceptions of the world) and what to represent, so that the knowledge is usable by machines**
- **Semantic Web:** **"T*he Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation*."**

  - T. Berners-Lee, J. Hendler, and O. Lassila. 2001. The Semantic Web. In *The Scientific American,* May, 2001.

The Problem

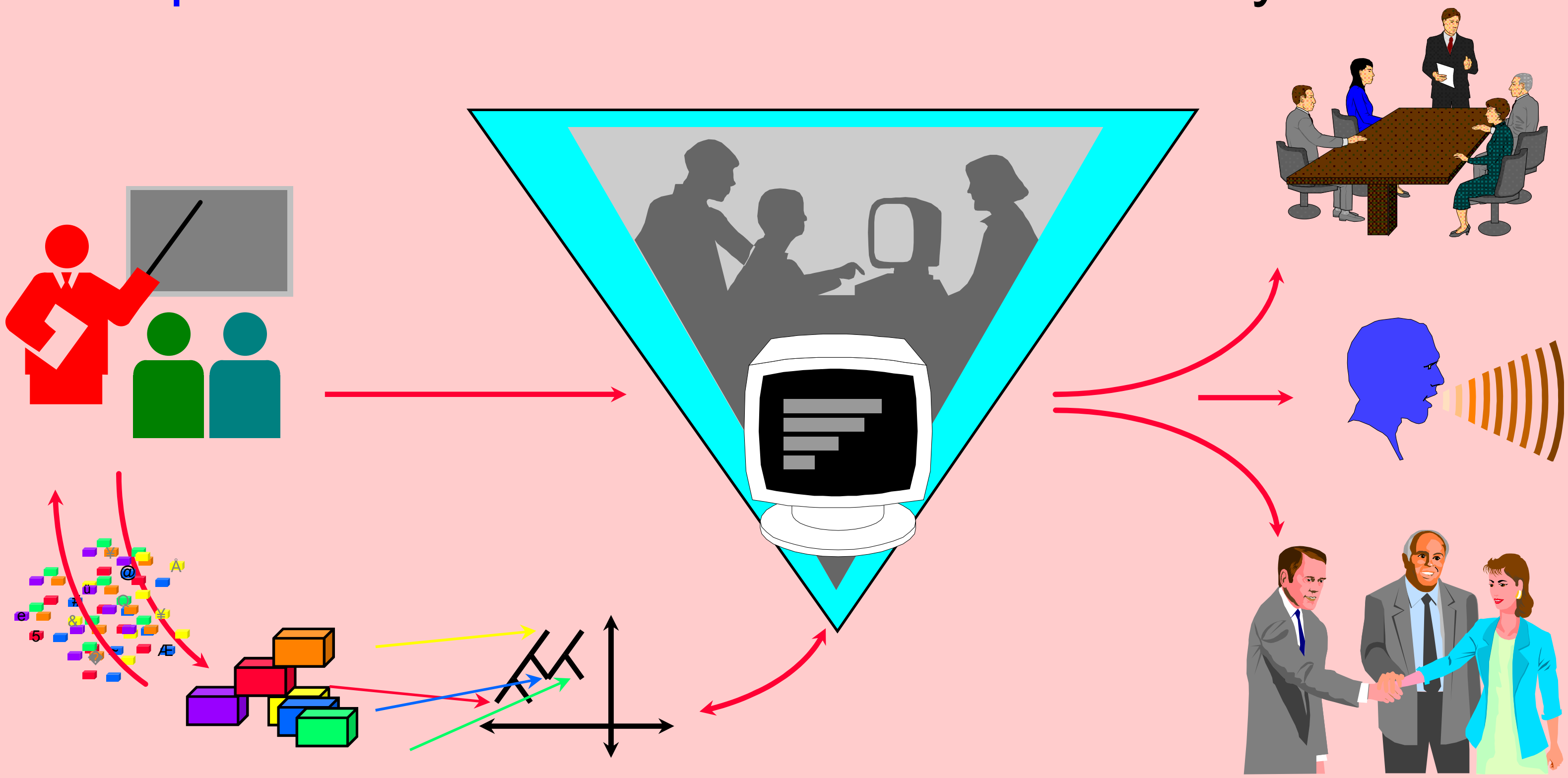
- With the increasing complexity of our systems and our IT needs, we need to go to human level interaction
- We need to maximize the amount of Semantics we can utilize
- From data and information level, we need to go to human *semantic* level interaction



- And represented semantics means multiply represented semantics, requiring *semantic integration*

The Solution

- We need to offload the very real, heavy cognitive interpretation burden from humans to our systems



- We need to represent human semantics using machine-interpretable ontologies

Advancing Along the Interpretation Continuum

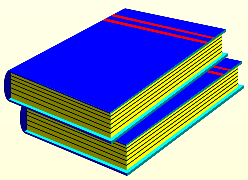
Simple Metadata: XML

Richer Metadata: RDF/S

Very Rich Metadata: OWL, CL, FOL

Human interpreted ⟵ **Interpretation Continuum** ⟶ ***Computer interpreted***

DATA
- **Relatively unstructured**
- **Random**

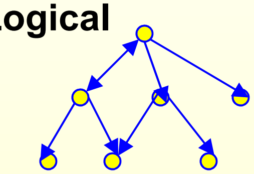


- **Info retrieval**
- **Web search**

- **Text summarization**
- **Content extraction**
- **Topic maps**

- **Reasoning services**
- **Ontology Induction**

KNOWLEDGE
- **Very structured**
- **Logical**



. . .

Display raw documents; All interpretation done by humans

Find and correlate patterns in raw docs; display matches only

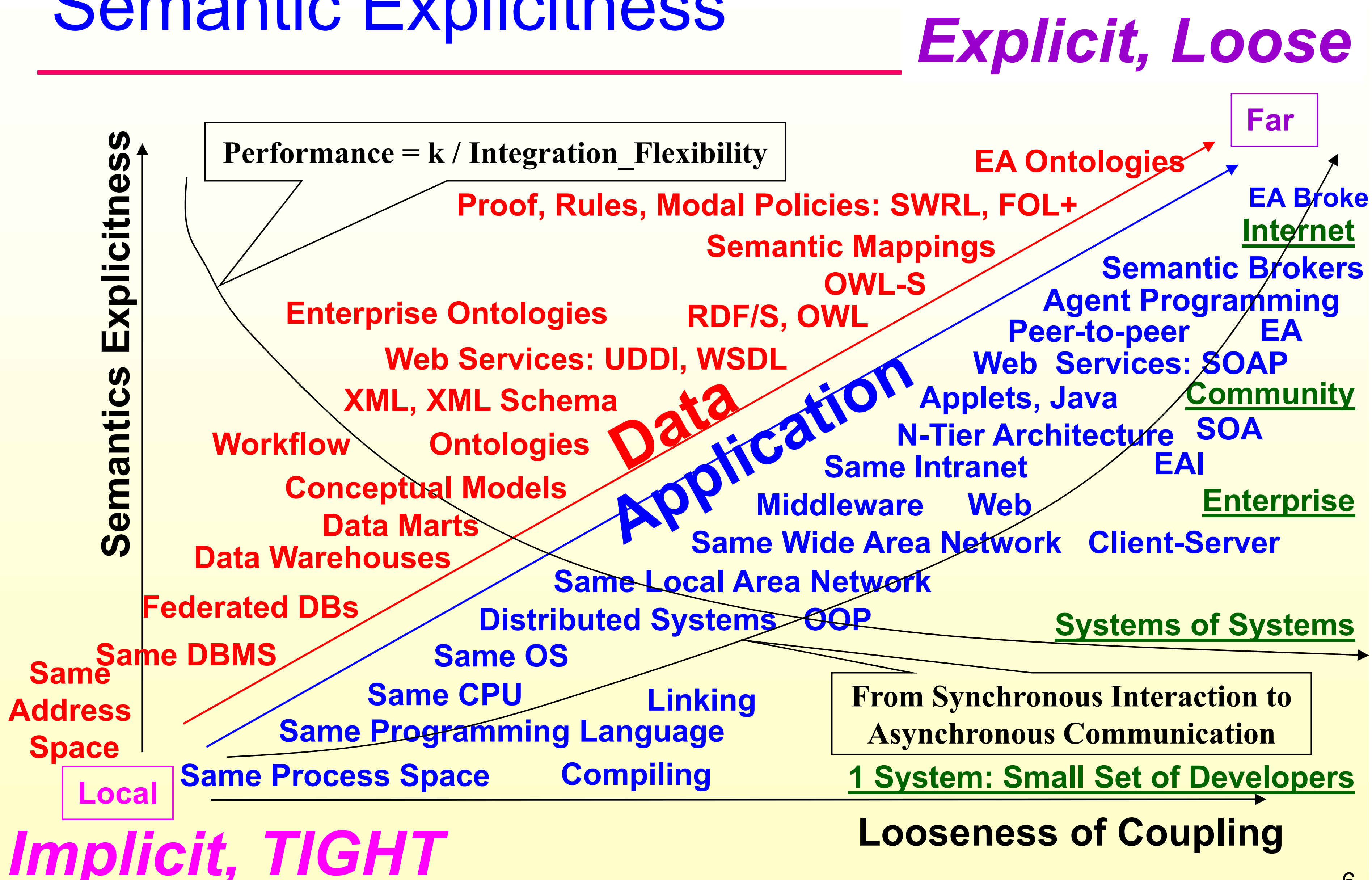
Store and connect patterns via conceptual model (i.e,. an ontology); link to docs to aid retrieval

Automatically acquire concepts; evolve ontologies into domain theories; link to institution repositories (e.g., MII)

Automatically span domain theories and institution repositories; inter-operate with fully interpreting computer

Moving to the right depends on increasing automated semantic interpretation

Motivation: Tightness of Coupling & Semantic Explicitness



Syntax

- A Language has a Syntax (set of symbols, & formation rules) & a Semantics (what the symbols, well-formed formulas mean)
- A formal language can be identified by its set of well-formed formulas; a natural language by its set of sentences (infinite)
- Syntax is form & structure
  - Symbols
  - Tokens/Types
    - Restricted words of a programming language
      - Do, While, Until, If, Then, Else, Declare
    - User defined constants & variables
      - A = 7 + 3; Y = A + 1; While Count < 5 Do
  - Order: how do words combine
    - To form a program?
    - To form a sentence?
    - Rules for combining: English grammar rules, BNF/EBNF rules
- Applies to Natural Languages, Programming Languages, Formal Languages, including Logics, Knowledge Representation/Ontology Languages!

Semantics: It All Depends on What ‘is’ is

- Semantics is meaning
- “Oh, it’s just semantics”: Wrong!
  - Implies that it’s quibbling about meaning, i.e., meaningless meaning, mincing words, not substantive or contentful distinctions
- “Real” semantics is about meaning
  - What meaning do we assign our squiggles on the page, pixels on the screen, ink on a map, sounds in a track, bits on a disk, flickering shades of dark & light on a film, squinting of an eye, a shrug?
  - What is the meaning of: ‘45-XG-92+@55’ ?
  - Is it the same or similar to ‘abk3#40’?
  - What is the meaning of ‘the man hit the ball’? ‘Green ideas sleep furiously’? ‘Hit man the the ball’? ‘Joe is a abk3#40’?
  - It’s the meaning of systems, services, data, documents, agents, humans

Semantics

- Semantics is meaning
  - Literal & figurative
  - Both context independent & context dependent
  - Meaning & use (intent of the meaning)
  - Natural language, programming & formal languages
  - Informal & formal
  - Express the meaning in a loose/strict, natural language definition or description
    - Semantics (Merriam-Webster, http://www.m-w.com/cgi-bin/dictionary)

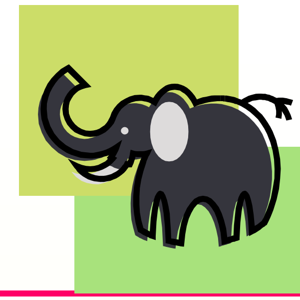
      1 : the study of meaning: a : the historical and psychological study and the classification of changes in the signification of words or forms viewed as factors in linguistic development b (1) : semiotic (2) : a branch of semiotics dealing with the relations between signs and what they refer to and including theories of denotation, extension, naming, and truth.
  - Express the meaning in a logical, mathematically rigorous manner
    - All students who took the test passed.

      $\forall x$: (student(x) $\wedge$ took_test(x) $\rightarrow$ passed_test(x))
- Syntax vs. Semantics: based on Language
  - A Language has a syntax and a semantics

Semantics: More

- Meta/Object levels: ‘p $\wedge$ q’ is a formula of Propositional Logic (PL)
- Use/Mention distinction:
  - Natural language can be “turned back on itself” (reflection): ‘The word *Socrates* has eight letters’
  - We use language to talk about language
  - ‘It depends on what the definition of *is* is’
- Type/Token distinction: related to Class/Instance
- Sense, Denotation, Reference: Triangle of Signification
- Extension vs. Intension: Triangle of Signification
- Lexical vs. Phrasal (Compositional) Meaning: words have their meanings, provide these to a compositional process of phrasal meaning
- **Semantics:** Using language or signage, ways to refer to the things of the world
- **Ontology:** The referents, the things of the world and their categories, properties

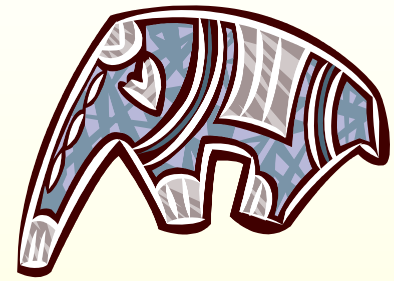
Ontology Elephants



There is no single real elephant

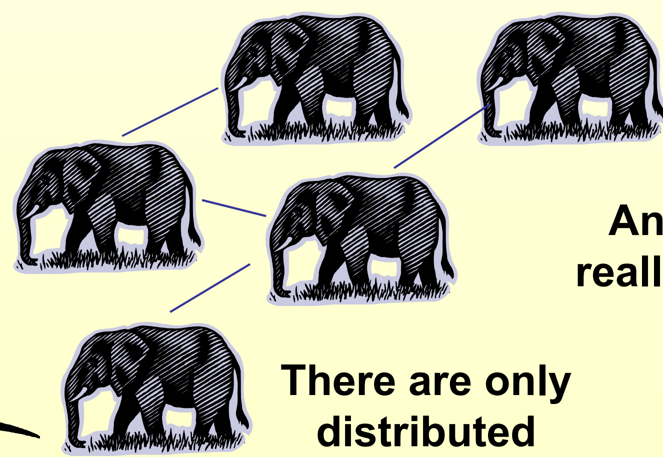
There must be a purpose for an elephant: use cases?

An elephant is abstract



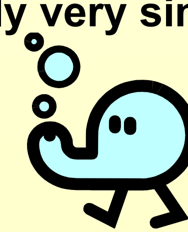
An elephant is **very** abstract

There must be an upper elephant



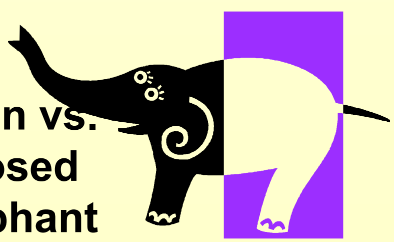
There are only distributed elephants & their mappings

An elephant is really very simple



An elephant is the result of consensus

Open vs. Closed Elephant



Some Issues

- We are like the blind men & the elephant: describing the ontology elephant from our own perspectives, which is of course what we most know about
- Multiple communities converging on semantics, with their own perspectives, concepts: see Ontology Spectrum
  - Logicians, formal ontologists, formal semanticists, some computer scientists
  - Librarian, information scientists
  - Object-oriented, development, programmers & software engineers
  - Classical AI knowledge representation folks
  - Database theorists & practitioners
  - Web community
  - Service Oriented Architecture (SOAs), Web services, enterprise architecture folks
  - Business & government analysts
- Problems:
  - Key distinctions are glossed over: term vs. concept, label vs. model, machine vs. human interpretablity, syntax vs. semantics-pragmatics (sense, reference, discourse, speech acts)

Ontology & Ontologies 1

- An ontology defines the terms used to describe and represent an area of knowledge (subject matter)
  - An ontology also is the model (set of concepts) for the meaning of those terms
  - An ontology thus defines the vocabulary and the meaning of that vocabulary
- Ontologies are used by people, databases, and applications that need to share domain information
  - Domain: a specific subject area or area of knowledge, like medicine, tool manufacturing, real estate, automobile repair, financial management, etc.
- Ontologies include computer-usable definitions of basic concepts in the domain and the relationships among them
  - They encode domain knowledge (modular)
  - Knowledge that spans domains (composable)
  - Make knowledge available (reusable)

Ontology & Ontologies 2

- The term ***ontology*** has been used to describe models with different degrees of structure (Ontology Spectrum)
  - **Less structure:** Taxonomies (Semio/Convera taxonomies, Yahoo hierarchy, biological taxonomy, UNSPSC), Database Schemas (many) and metadata schemes (ICML, ebXML, WSDL)
  - **More Structure:** Thesauri (WordNet, CALL, DTIC), Conceptual Models (OO models, UML)
  - **Most Structure:** Logical Theories (Ontolingua, TOVE, CYC, Semantic Web)
- Ontologies are usually expressed in a logic-based language
  - Enabling detailed, sound, meaningful distinctions to be made among the classes, properties, & relations
  - More expressive meaning but maintain "computability"
- Using ontologies, tomorrow's applications can be "intelligent"
  - Work at the human conceptual level
- Ontologies are usually developed using special tools that can model rich semantics

Big O: Ontology, Little o: ontology

- **Philosophy:** “a particular system of categories accounting for a certain vision of the world” or domain of discourse, a conceptualization (Big O)
- **Computer Science:** “an engineering product consisting of a specific vocabulary used to describe a part of reality, plus a set of explicit assumptions regarding the intended meaning of the vocabulary words”, “a specification of a conceptualization” (Little o)
- **Ontology Engineering:** towards a formal, logical theory, usually ‘concepts’ (i.e., the entities, usually classes hierarchically structured in a special subsumption relation), ‘relations’, ‘properties’, ‘values’, ‘constraints’, ‘rules’, ‘instances’, so:
- **Ontology (in our usage):**

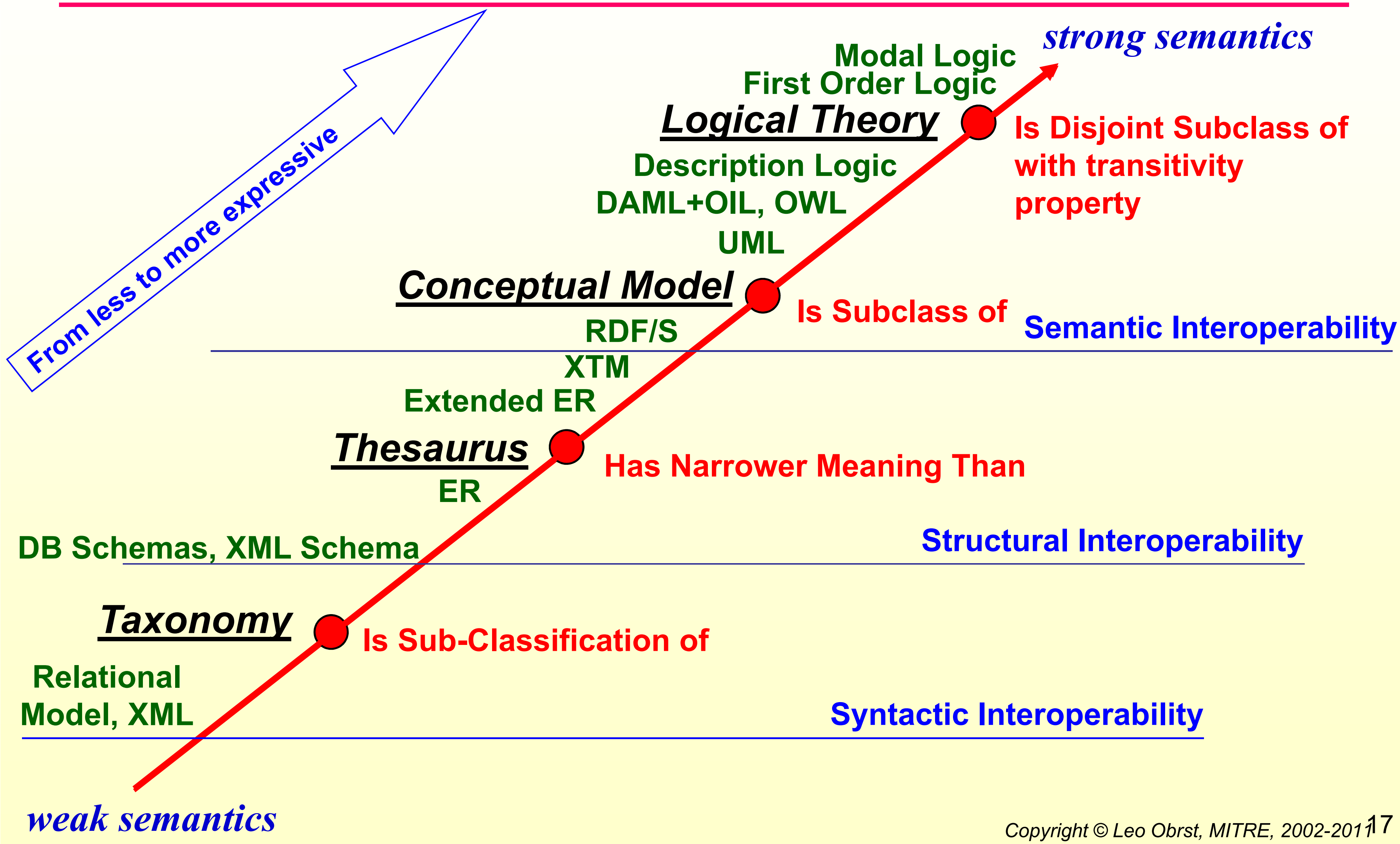
1) **A logical theory**
2) **About the world or some portion of the world**
3) **Represented in a form semantically interpretable by computer**
4) **Thus enabling automated reasoning comparable to a human’s**

* The first two definitions are derived from Guarino, 98; Guarino & Giaretta, 95; Gruber, 93, 94

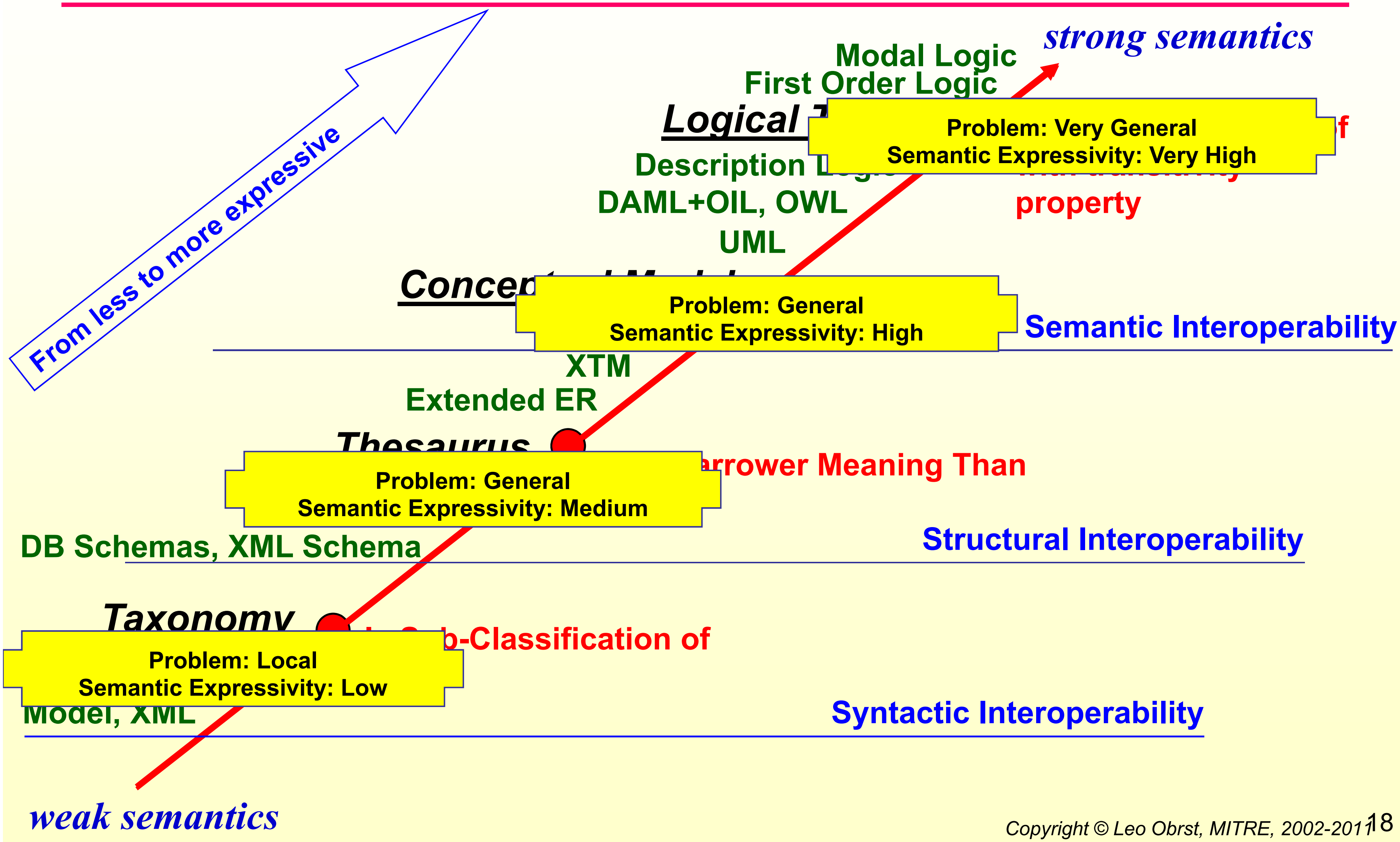
Ontology thus includes:

- **Objects** (things) in the many domains of interest
- The **relationships** between those things
- The **properties** (and property values) of those things
- The **functions and processes** involving those things
- **Constraints** on and **rules** about those things

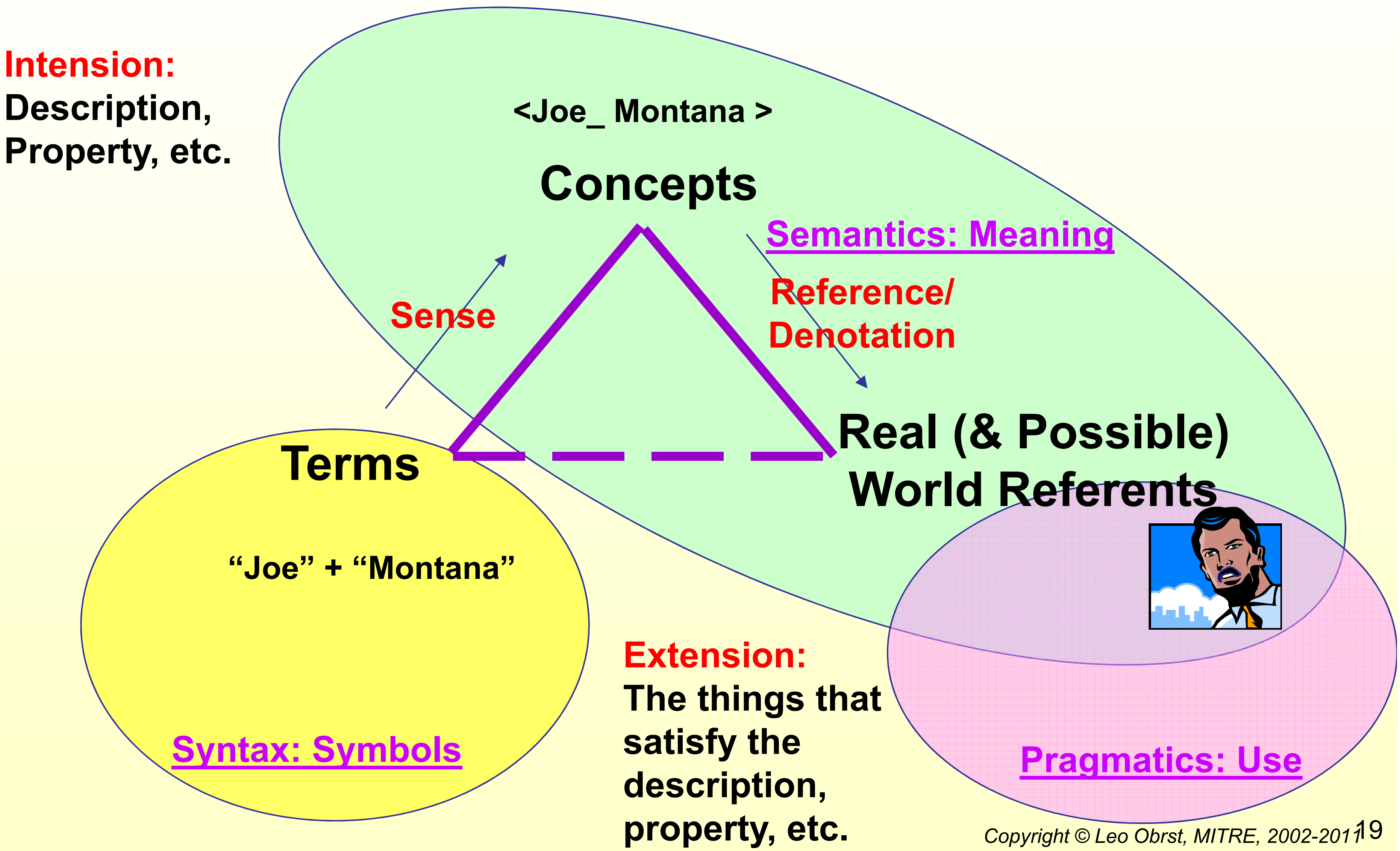
Ontology Spectrum: Range of Models



Ontology Spectrum: Generality & Expressiveness

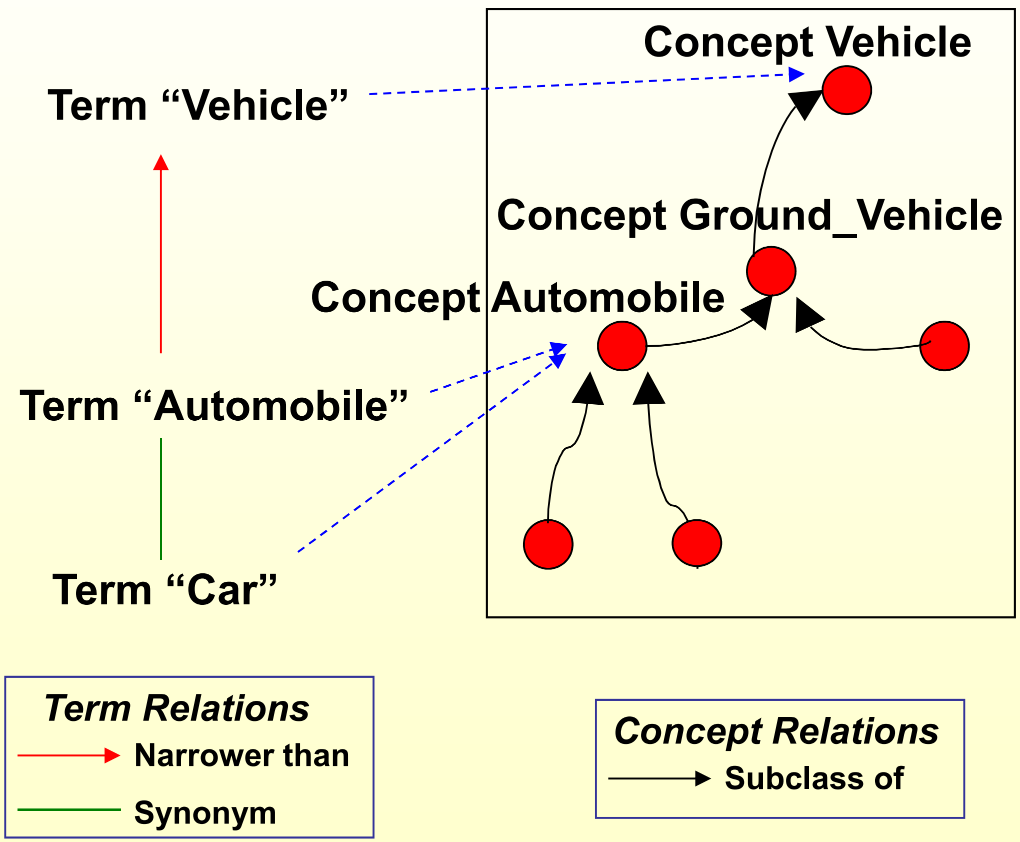


Triangle of Signification

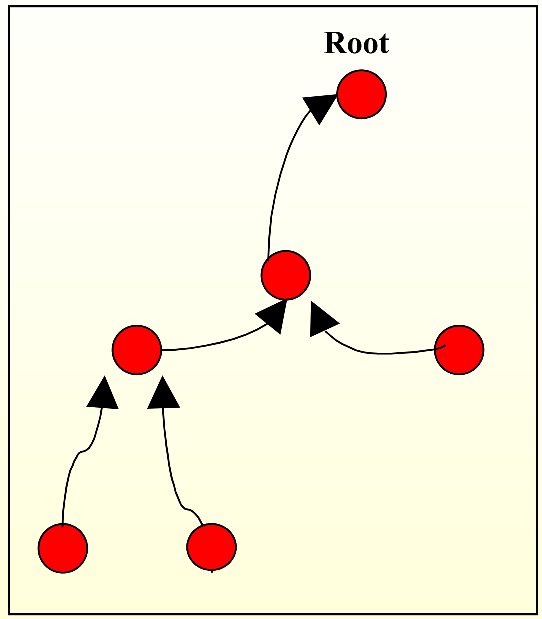


Term vs. Concept

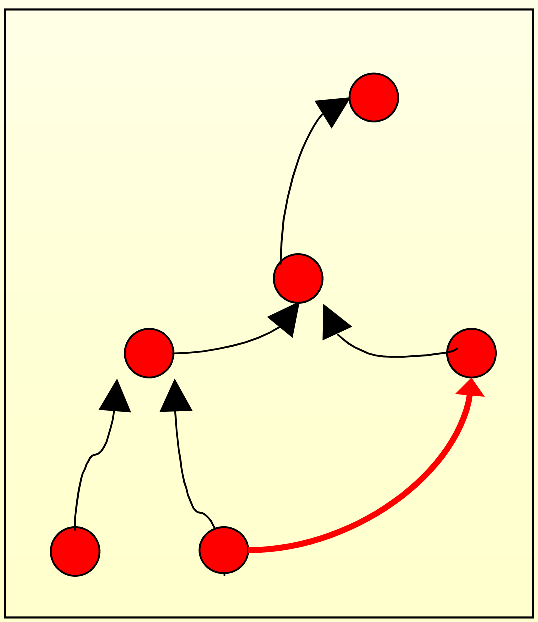
- **Term (terminology):**
  - Natural language words or phrases that act as indices to the underlying meaning, i.e., the concept (or composition of concepts)
  - The syntax (e.g., string) that stands in for or is used to indicate the semantics (meaning)
- **Concept:**
  - A unit of semantics (meaning), the node (entity) or link (relation) in the mental or knowledge representation model



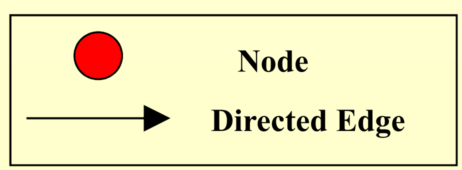
Tree vs. Graph

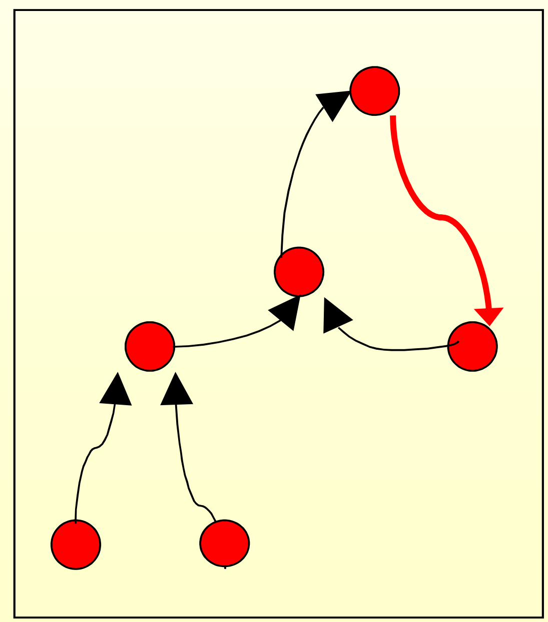


Tree



Directed Acyclic Graph





Directed Cyclic Graph

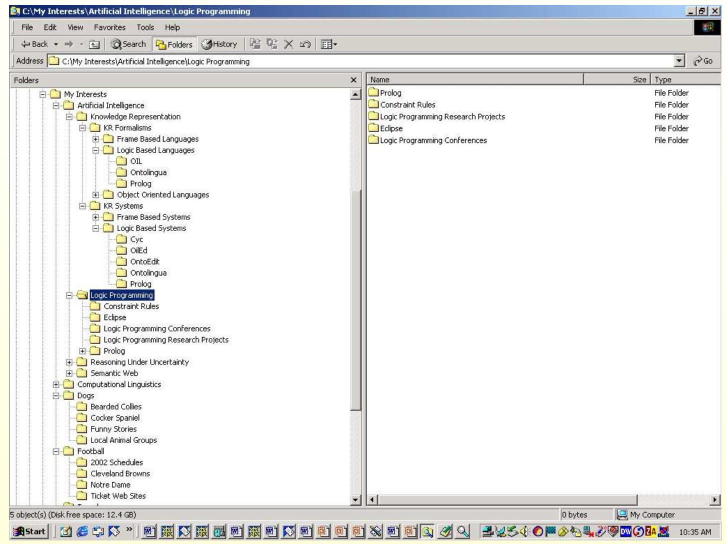
Taxonomy: Definition

- Taxonomy:
  - A way of classifying or categorizing a set of things, i.e., a classification in the form of a hierarchy (tree)
- IT Taxonomy:
  - The classification of information entities in the form of a hierarchy (tree), according to the presumed relationships of the real world entities which they represent
- Therefore: A taxonomy is a semantic (term or concept) hierarchy in which information entities are related by either:
  - The ***subclassification of*** relation (weak taxonomies) or
  - The ***subclass of*** relation (strong taxonomies) for concepts or the **narrower than** relation (thesauri) for terms
  - **Only the subclass/narrower than relation is a subsumption (generalization/specialization) relation**
  - **Subsumption (generalization/specialization) relation:** the mathematical subset relation
  - Mathematically, strong taxonomies, thesauri, conceptual models, and logical theories are minimally Partially Ordered Sets (posets), i.e., they are ordered by the subset relation
    - They may be mathematically something stronger (conceptual models and logical theories)

Taxonomies: Weak

- No consistent semantics for parent-child relationship: arbitrary ***Subclassification Relation***
- NOT a *generalization / specialization* taxonomy

Example: Your Folder/Directory Structure

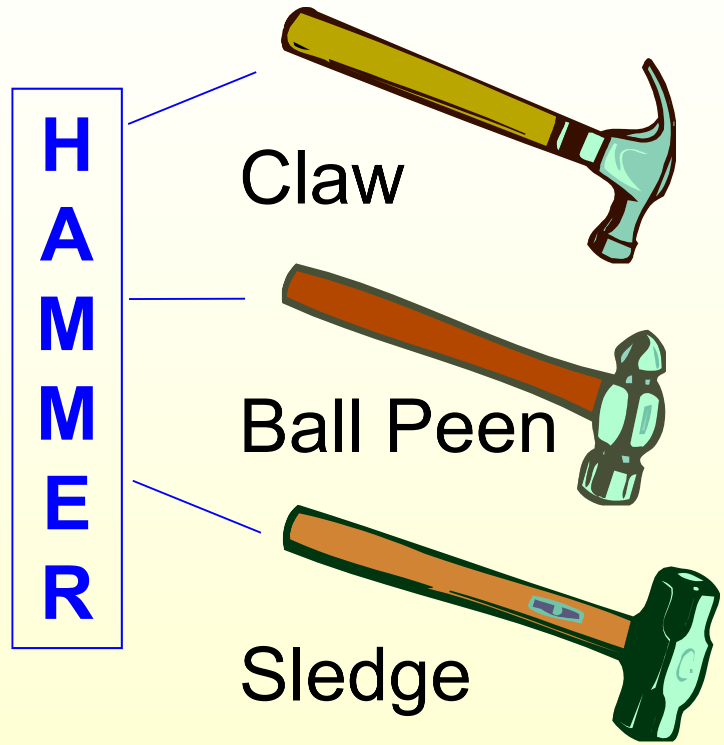


Example: UNSPSC

Segment	Family	Class	Commodity	Title
10	00	00	00	Live Plant and Animal Material and Accessories and Supplies
10	10	00	00	Live animals
10	10	15	00	Livestock
10	10	15	01	Cats
10	10	15	02	Dogs

Taxonomies: Strong

- Consistent semantics for parent-child relationship: ***Narrower than (terms) or Subclass (concepts) Relation***
- *A generalization/specialization* taxonomy
- **For concepts:** Each information entity is distinguished by a property of the entity that makes it unique as a subclass of its parent entity (a synonym for property is attribute or quality)
- **For terms:** each child term **implicitly** refers to a concept which is the subset of the concept referred to by its parent term



- **What are the *distinguishing properties* between these three hammers?**
  - **Form (physical property)**
  - **Function (functional property)**
- **“Purpose proposes property” (form follows function) – for human artifacts, at least**

Thesaurus: Definition

- From ANSI INISO 239.19-1993, (Revision of 239.194980):
  - A **thesaurus is a controlled vocabulary** arranged in a known order and structured so that equivalence, homographic, hierarchical, and associative relationships among terms are displayed clearly and identified by standardized relationship indicators
  - The **primary purposes of a thesaurus** are to facilitate retrieval of documents and to achieve consistency in the indexing of written or otherwise recorded documents and other items
- Four Term Semantic Relationships:
  - **Equivalence:** synonymous terms
  - **Homographic:** terms spelled the same
  - **Hierarchical:** a term which is broader or narrower than another term
  - **Associative:** related term
- A consistent semantics for the hierarchical parent-child relationship: broader than, narrower than
- This hierarchical ordering is a *Subsumption (i.e., generalization/specialization)* relation
- Can view just the *narrower-than* subsumption hierarchy as a **term taxonomy**
- Unlike Strong subclass-based Taxonomy, Conceptual Model, & Logical Theory: the relation is between Terms, NOT Concepts

Thesaural Term Relationships

Semantic Relation	*Definition*	*Example*
<u>Synonym</u> **Similar to** **Equivalent** **Used For**	A term X has nearly the same meaning as a term Y.	"Car" is a synonym for "automobile".
<u>Homonym</u> **Spelled the Same** **Homographic**	A term X is spelled the same way as a term Y, which has a different meaning	The "bank" which is a financial institution is a homonym for the "bank" which is the side of a river or stream.
<u>Broader Than</u> **(Hierarchic: parent of)**	A term X is broader in meaning than a term Y.	"Vehicle" has a broader meaning than "automobile".
<u>Narrower Than</u> **(Hierarchic: child of)**	A term X is narrower in meaning than a term Y.	"Automobile" has a narrower meaning than "vehicle".
<u>Associated</u> **Associative** **Related**	A term X is associated with a term Y, i.e., there is some unspecified relationship between the two.	A "comb" is associated with a "barber".

Thesaurus vs. Ontology

Controlled Vocabulary

Terms: Metal working machinery, equipment and supplies, metal-cutting machinery, metal-turning equipment, metal-milling equipment, milling insert, turning insert, etc.
Relations: use, used-for, broader-term, narrower-term, related-term

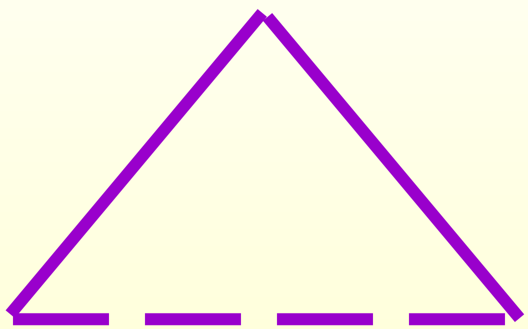
Thesaurus

Term Semantics (Weak)

Ontology

Logical-Conceptual Semantics (Strong)

Concepts



Terms

Real (& Possible) World Referents

'Semantic' Relations:

- Equivalent =
- Used For (Synonym) UF
- Broader Term BT
- Narrower Term NT
- Related Term RT

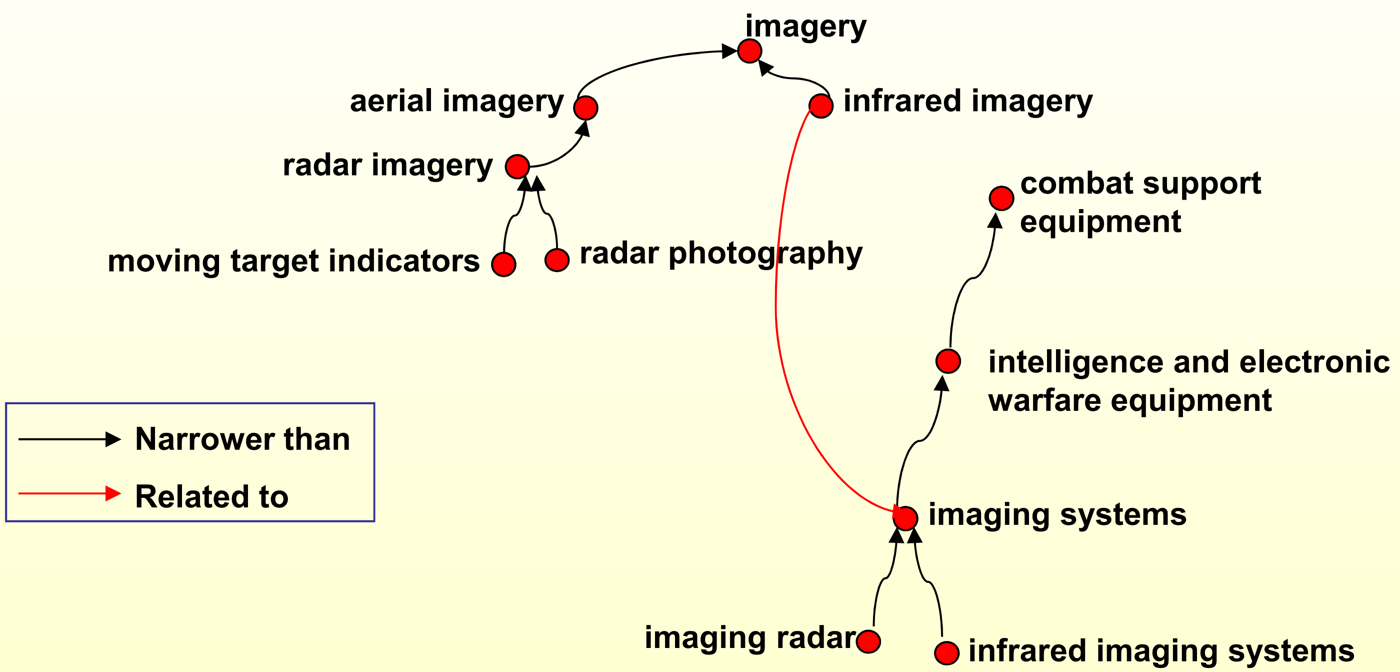
Logical Concepts

Entities: Metal working machinery, equipment and supplies, metal-cutting machinery, metal-turning equipment, metal-milling equipment, milling insert, turning insert, etc.
Relations: subclass-of; instance-of; part-of; has-geometry; performs, used-on;etc.
Properties: geometry; material; length; operation; UN/SPSC-code; ISO-code; etc.
Values: 1; 2; 3; "2.5 inches"; "85-degree-diamond"; "231716"; "boring"; "drilling"; etc.
Axioms/Rules: If milling-insert(X) & operation(Y) & material(Z)=HG_Steel & performs(X, Y, Z), then has-geometry(X, 85-degree-diamond).

Semantic Relations:

- Subclass Of
- Part Of
- Arbitrary Relations
- Meta-Properties on Relations

Center For Army Lessons Learned (CALL) Thesaurus Example

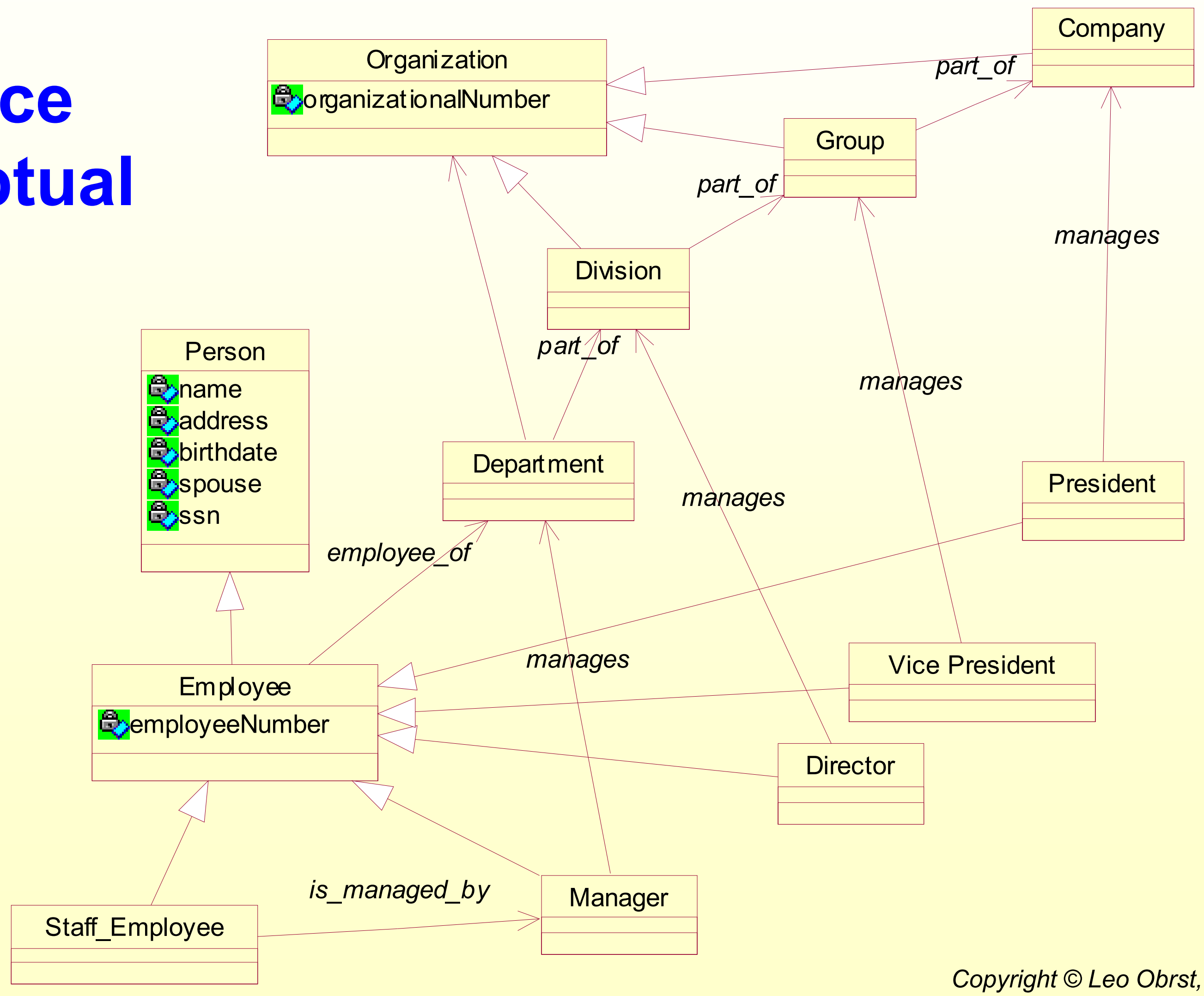


Conceptual Models: Weak Ontologies

- Many conceptual domains cannot be expressed adequately with a taxonomy (nor with a thesaurus, which models term relationships, as opposed to concept relationships)
- Conceptual models seek to model a portion of a domain that a database must contain data for or a system (or, recently, enterprise) must perform work for, by providing users with the type of functionality they require in that domain
- UML is paradigmatic modeling language
- Drawbacks:
  - Models mostly used for documentation, required human semantic interpretation
  - Limited machine usability because cannot directly interpret semantically
  - Primary reason: there is no Logic that UML is based on
- **You need more than a Conceptual Model** if you need machine-interpretability (more than machine-processing)
  - You need a logical theory (high-end ontology)

Conceptual Model: UML Example

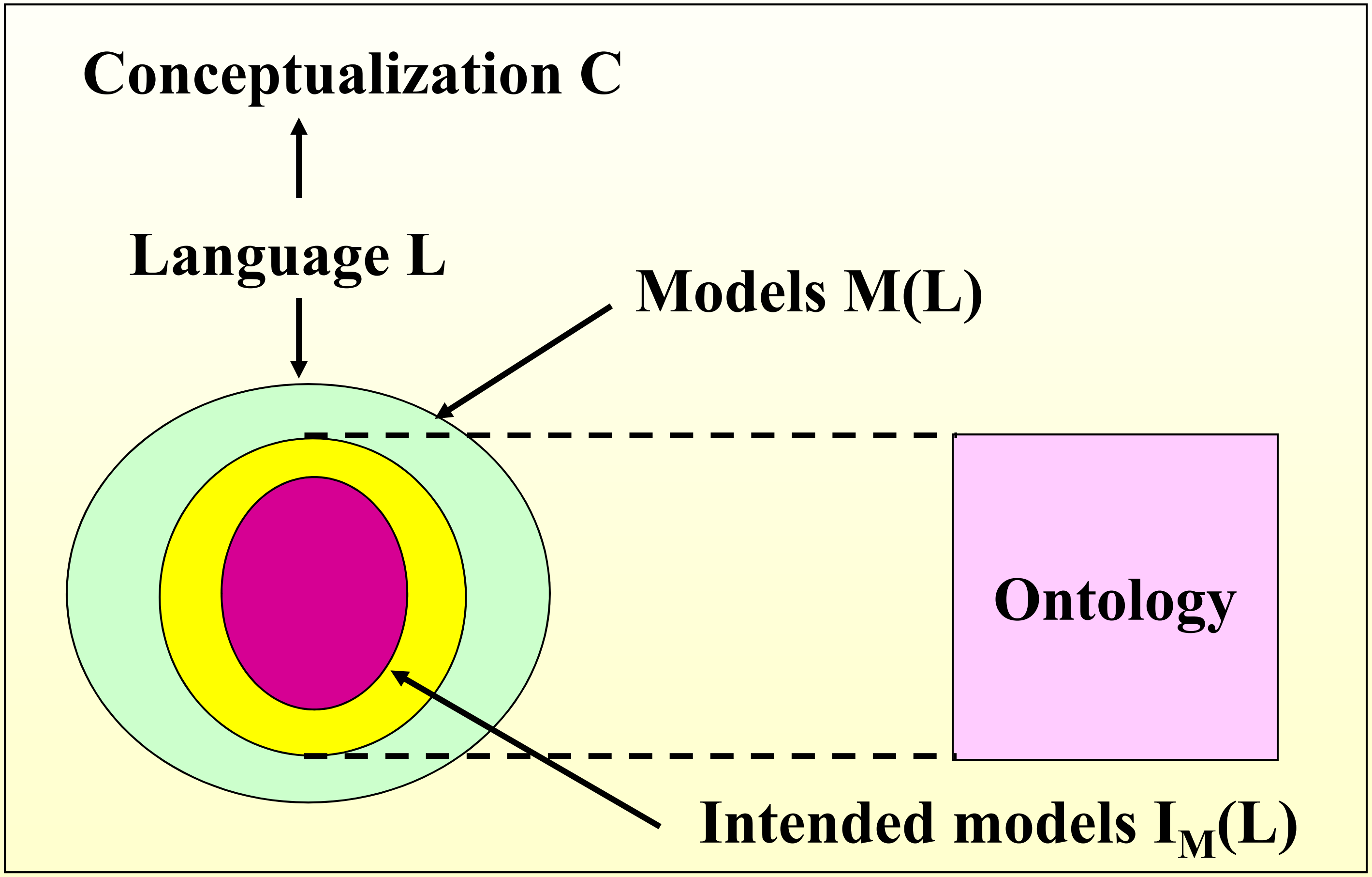
Human Resource Conceptual Model



Logical Theories: Strong Ontologies

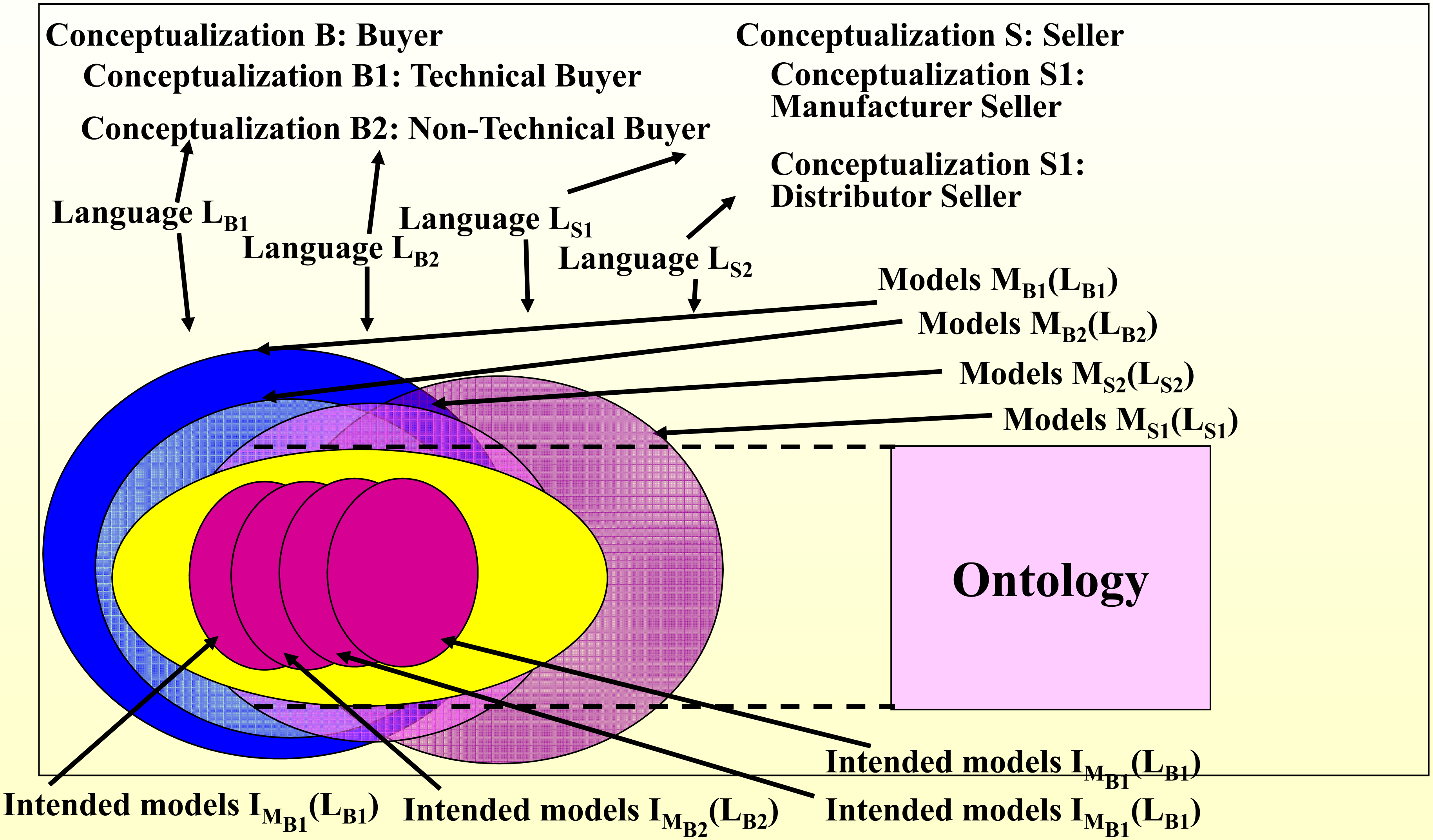
- Can be either Frame-based or Axiomatic
  - Frame-based: node-and-link structured in languages which hide the logical expressions, entity-centric, like object-oriented modeling, centering on the entity class, its attributes, properties, relations/associations, and constraints/rules
  - Axiomatic: axiom/rule-structured in languages which expose the logical expressions, non-entity-centric, so axioms that refer to entities (classes, instances, their attributes, properties, relations, constraint/rules) can be distributed

Logical Theories: More Formally

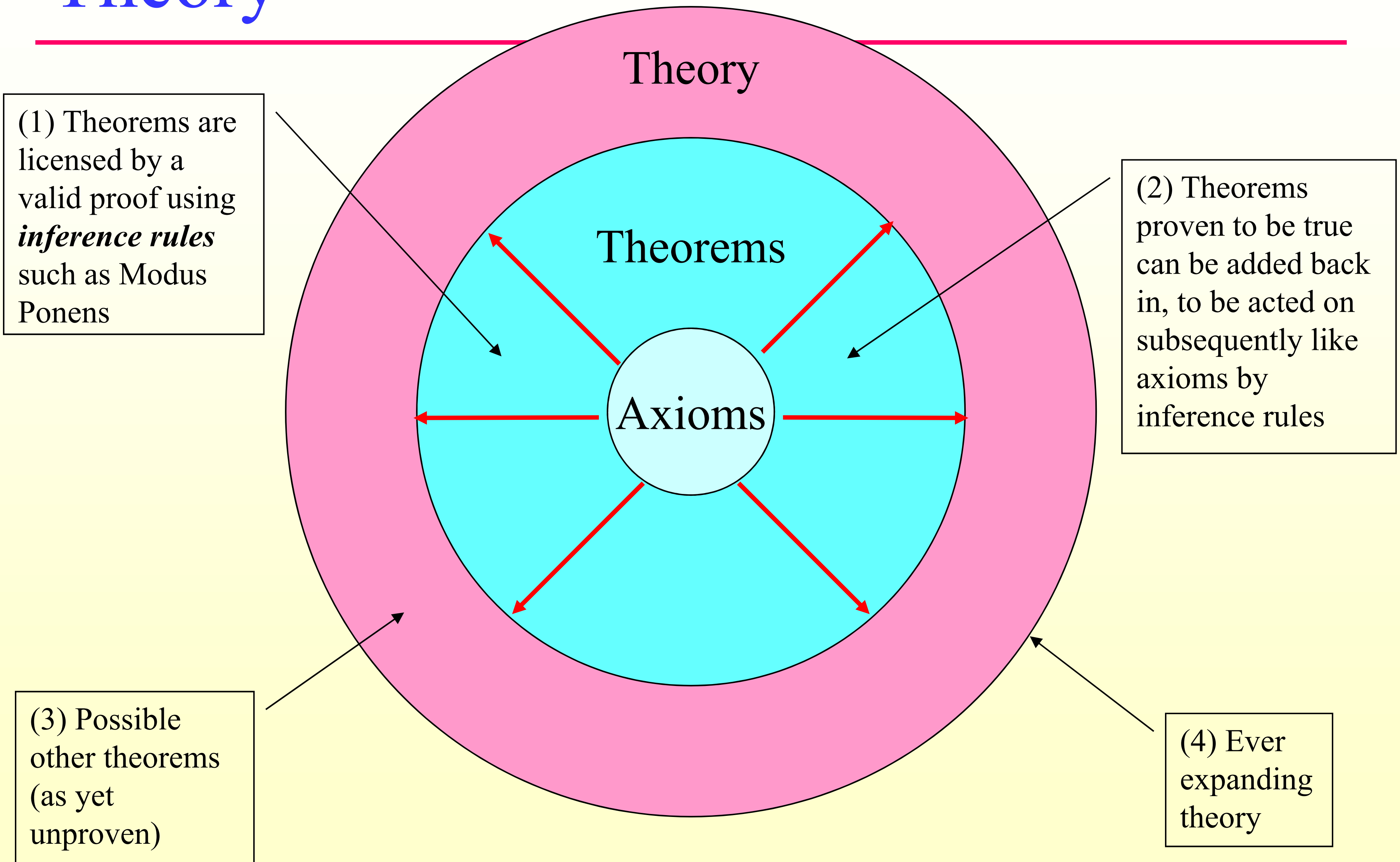


* N. Guarino. 1998. Formal ontology in information systems, pp. 3-15. In Formal Ontology in Information Systems, N. Guarino, ed., Amsterdam: IOS Press. Proceedings of the First International Conference (FOIS'98), June 6-8, Trent, Italy, p. 7

A More Complex Picture (from E-Commerce)



Axioms, Inference Rules, Theorems, Theory



Axioms

Class(Thing)

Class(Person)

Class(Parent)

Class(Child)

If SubClass(X, Y) then X is a subset of Y. This also means that if A is a member of Class(X), then A is a member of Class(Y)

SubClass(Person, Thing)

SubClass(Parent, Person)

SubClass(Child, Person)

ParentOf(Parent, Child)

NameOf(Person, String)

AgeOf(Person, Integer)

If X is a member of Class (Parent) and Y is a member of Class(Child), then $\neg$ (X =Y)

Inference Rules

And-introduction: given P, Q, it is valid to infer $P \wedge Q$.

Or-introduction: given P, it is valid to infer $P \vee Q$.

And-elimination: given $P \wedge Q$, it is valid to infer P.

Excluded middle: $P \vee \neg P$ (i.e., either something is true or its negation is true)

Modus Ponens: given $P \rightarrow Q$, P, it is valid to infer Q

Theorems

If $P \wedge Q$ are true, then so is $P \vee Q$.

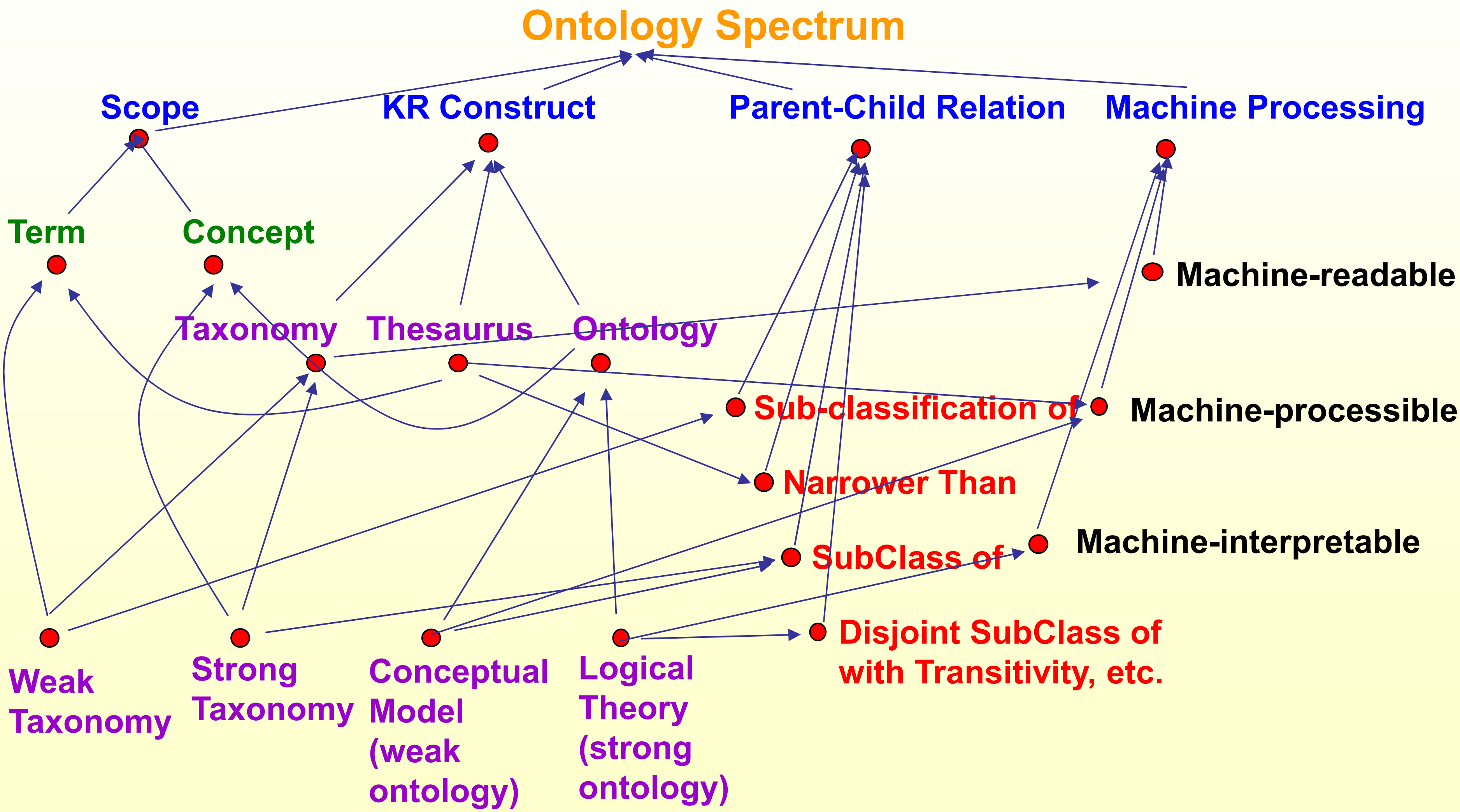
If X is a member of Class(Parent), then X is a member of Class(Person).

If X is a member of Class(Child), then X is a member of Class(Person).

If X is a member of Class(Child), then NameOf(X, Y) and Y is a String.

If Person(JohnSmith), then $\neg$ ParentOf(JohnSmith, JohnSmith).

Summary of Ontology Spectrum: Scope, KR Construct, Parent-Child Relation, Processing Capability



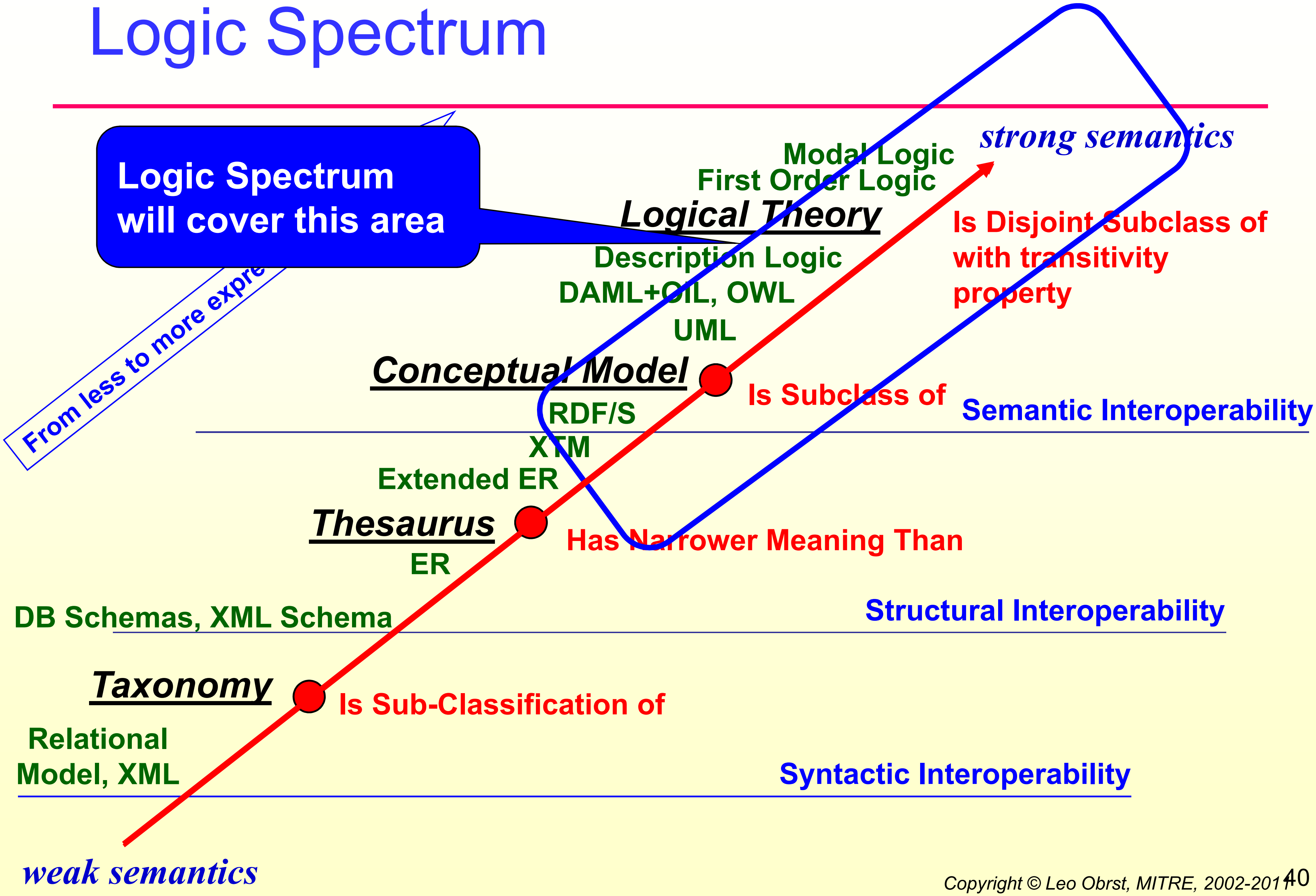
Part 1 Conclusions

- Ontology: a specification of a conceptualization, vocabulary + model, theory
- Informally, ontology and model are taken to be synonymous, i.e, a description of the structure and meaning of a domain, a conceptual model
- Bottom Line: *an Ontology* models *Concepts, i.e.,* the entities (usually structured in a class hierarchy with *multiple inheritance*), relations, properties (attributes), values, instances, constraints, and rules used to model one or more domains
  1) **A logical theory**
  2) **About the world or some portion of the world**
  3) **Represented in a form semantically interpretable by computer**
  4) **Thus enabling automated reasoning comparable to a human's**
- Logically, you can view an ontology as a set of *Axioms* (statements and constraints/rules) about some domain
- Using the axioms and some defined *Inference Rules* (example: Modus Ponens), you can derive (prove true) *Theorems* about that domain, and thus derive knew knowledge

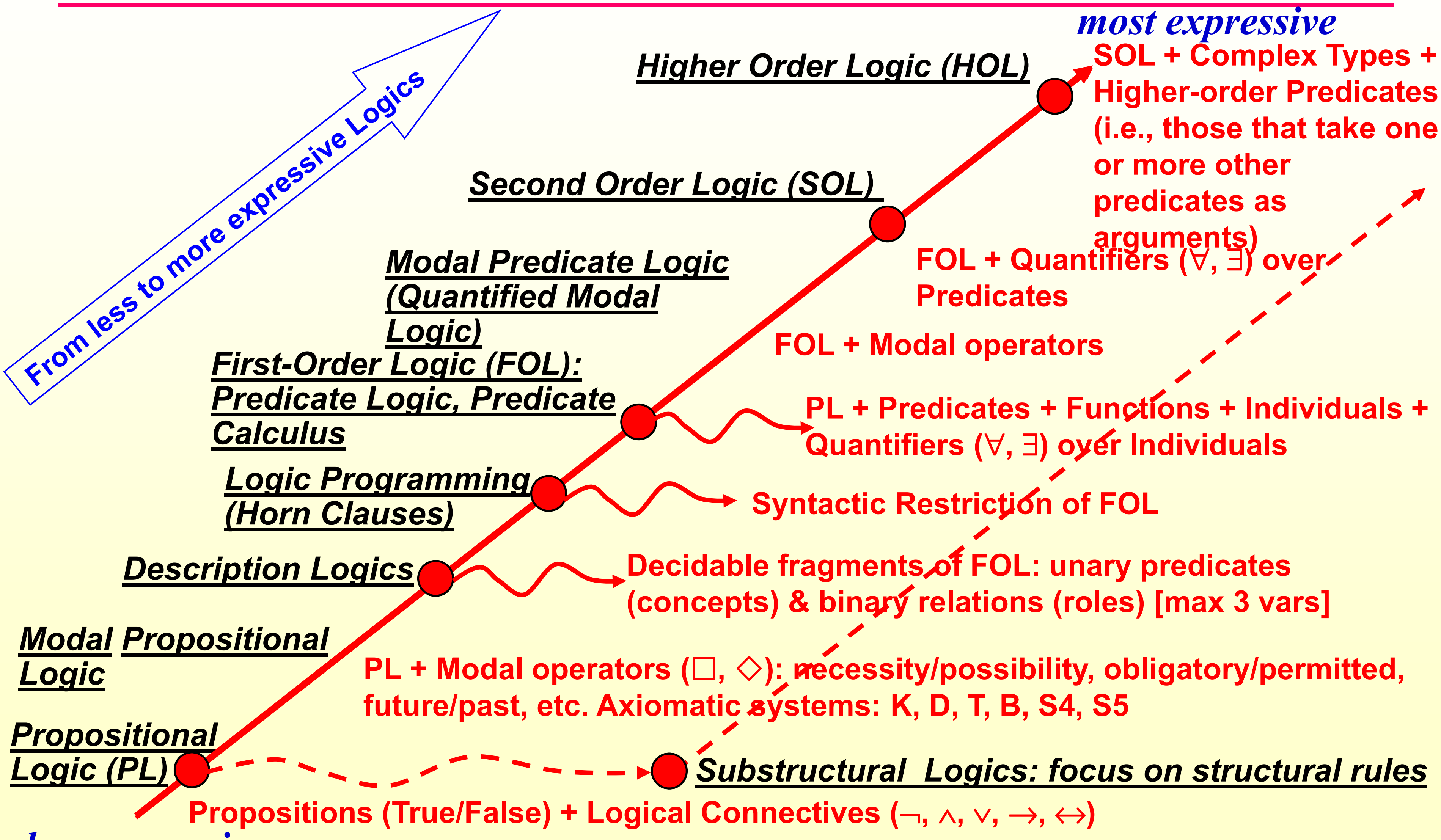
Take Break!

Agenda, Part 2:
Logic, Ontologies, Semantic Web

From Ontology Spectrum to Logic Spectrum



Logic Spectrum: Classical Logics: PL to HOL



Propositional & Predicate Logic

- Propositional Logic
  - Limitation: cannot speak about individuals (instances)
  - Granularity not fine enough
  - Propositions: truth-functions

If Plato is human, then Plato is mortal	$p \rightarrow q$	
Plato is human	p	
________________		
Plato is mortal	q	Modus Ponens

- Predicate Logic
  - Finer distinctions: can talk about individuals (instances)

If Plato is human, then Plato is mortal	$\forall x: p(x) \rightarrow q(x)$	
Plato is human	p(plato)	
________________		
Plato is mortal	q(plato)	Modus Ponens

  - An instantiated predicate is a proposition, e.g., human(plato) = true

First Order & Higher Order Logics: the basis of other Ontology Languages

- **FOL semi-decidable**
  - **Decidable: there is an effective method for telling whether or not each formula of a system is a theorem of that system or not**
  - **Semi-decidable: If a formula really is a theorem of a system, eventually will be able to prove it is, but not if it is not: may never terminate**
- **Second Order: sometimes used in linguistics**
  - **“Tall”, “Most”, etc.**
  - **Quantification over Individual & Predicate variables**
  - **$\exists\phi\ (\phi\ (a) \wedge F(\phi))$: “John has an unusual property”**
- **CYC: MELD, CYCL, has some constrained 2nd order reasoning**
- **Theorem-provers**
  - **HOL, Otter, etc.**
- **Prolog & Cousins**
  - **Restricted FOL: Horn Clauses (only 1 un-negated term in a formula, resolution method proves the contradiction of the negation of a term)**
  - **Non-standard negation: negation by finite failure**
  - **Closed World Assumption**
  - **Declarative + Operational Semantics: use of Cut**
- **Other: Conceptual Graphs, UML, Expert System Shells, Modal Logics**

Example: Inference and Proof

Proof Using Inference Rule of Modus Ponens

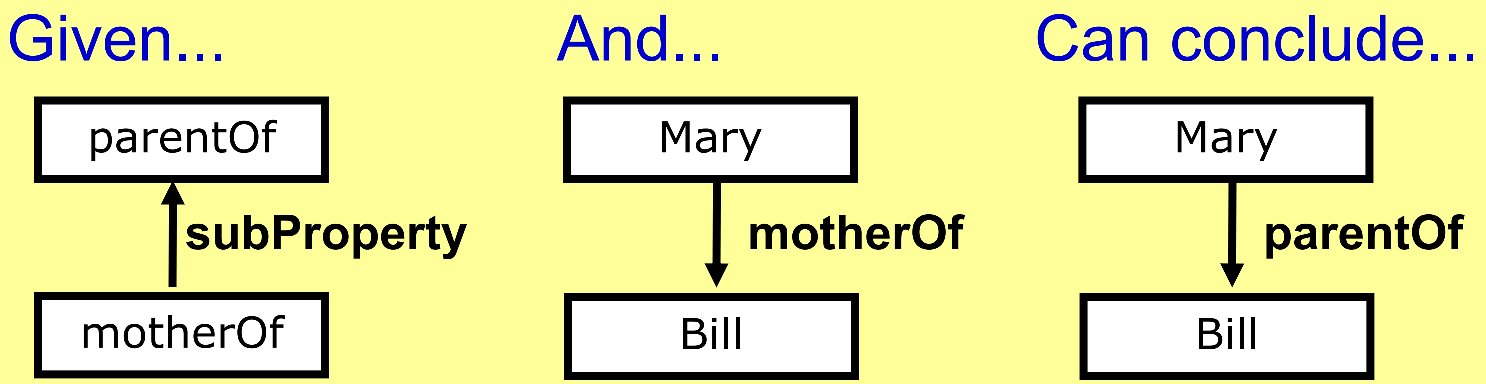
Given: **If motherOf is a subProperty of parentOf, and Mary is the mother of Bill, then Mary is the parentOf Bill**

motherOf is a subProperty of parentOf

Mary is the motherOf Bill

Infer: **Mary is the parentOf Bill**

Deduction A method of reasoning by which one infers a conclusion from a set of sentences by employing the axioms and rules of inference for a given logical system.



A simple inferencing example from “Why use OWL?” by Adam Pease, http://www.xfront.com/why-use-owl.html

Description Logic: Definitions

- What is a Description Logic? Terminological Logic, Concept Logic, based on: Concept Language, Term Subsumption Language
  - A declarative formalism for the representation and expression of knowledge and sound, tractable reasoning methods founded on a firm theoretical (logical) basis
    - **DL frame-based semantic network + logic (compositional syntax and model-theoretic semantics)**
    - **usual logical formulation of a concept would be as a single-variable predicate, i.e., in lambda calculus, as (MacGregor, 1991):**
    - **adult males: λx. Male(x) $\cup$ Adult(x)**
  - **Expressive, sound & complete, decidable, classical semantics, tractable reasoning**
  - **Function-free FOL using at most 3 variables (basic)**
- A *description*: an expression in a formal language that defines a set of instances or tuples
- DL: a syntax for constructing descriptions and a semantics that defines the meaning of each description

Description Logic: Components

- *T-box: Terminological box – concepts, classes, predicates*
  – One or more subsumption hierarchies/taxonomies of descriptions
  – Terminological axioms: introduce names of concepts, roles
  – Concepts: denote entities
  – Roles: denote properties (binary predicates, relations)
  – OO? No, but related. Why: no generally agreed upon formal basis to OO, though attempts (emerging UML)
    - **Isa generalization/specialization, Top/ Bottom**
    - **Part-of: mereology, mereotopology (parts+connections)**
    - **Other relations: aggregation, etc.**
  – Subsumption: comparable to matching or unification in other systems
- *A-box: Assertional box – individuals, constants*
  – Instances in the OO world, tuples in the DB world

Description Logic: Inference Methods & Properties

- Inference Methods (all based on subsumption)
  - **classification:** where do descriptions belong in hierarchies (subsumers, subsumees)
  - **detecting contradiction:** are descriptions coherent/satisfiable and is the KB consistent/satisfiable
  - **completion inference:** what are the logical consequences of axioms, inheritance
- Inference algorithms properties:
  - **soundness:** any expression that can be derived from the KB is logically implied by that KB
  - **completeness:** any expression that is logically implied by the KB can be derived
  - **decidability:** can a sound and complete algorithm be constructed?
  - **complexity:** is it tractable (worst-case polynomial time) or intractable?
  - **expressivity:**
    - roughly: expressivity and tractability are inversely proportional
    - some expressive formalisms may be intractable or even undecidable

Example: OIL, which became DAML+OIL, which became OWL

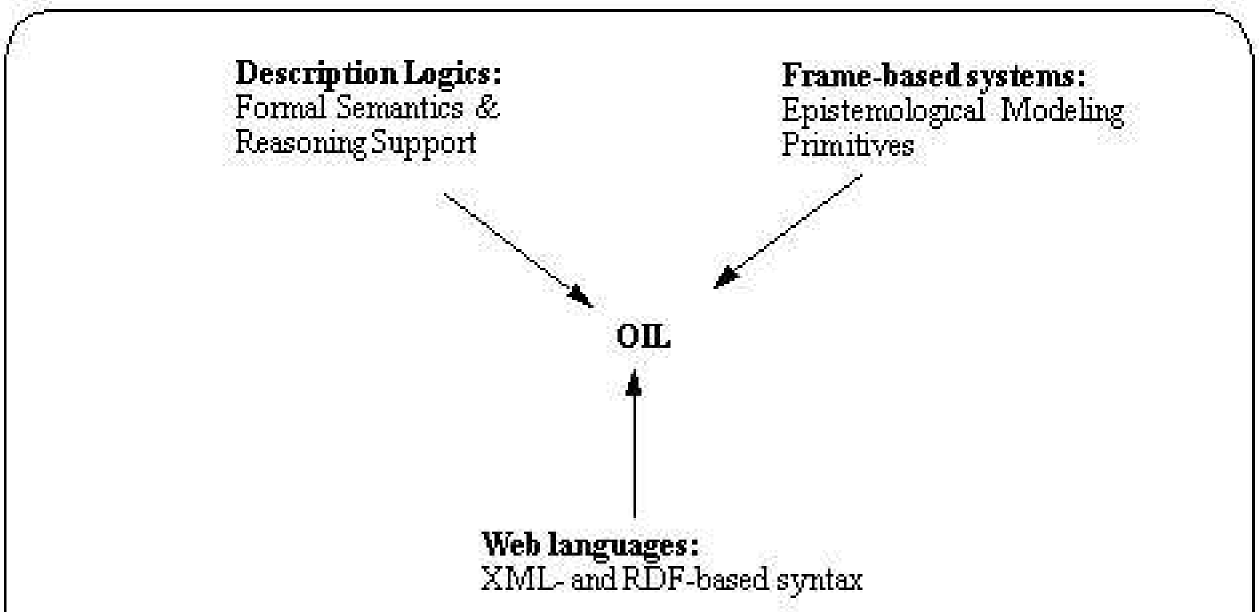


Fig. 1 The three roots of OIL.

Ontology Inference Layer/Language (OIL, merged to be DAML+OIL, now OWL)

Horrocks I. , D. Fensel, J. Broekstra, S. Decker, M. Erdmann, C. Goble, F. van Harmelen, M. Klein, S. Staab, R. Studer, and E. Motta. 2000. The Ontology Inference Layer OIL. http://www.ontoknowledge.org/oil/TR/oil.long.html

Back to Ontology: Ontology Representation Levels

Level	Example Constructs
Knowledge Representation (KR) Language (Ontology Language) Level: **Meta Level to the Ontology Concept Level**	**Class, Relation, Instance, Function, Attribute, Property, Constraint, Axiom, Rule**
Ontology Concept (OC) Level: **Object Level to the KR Language Level, Meta Level to the Instance Level**	**Person, Location, Event, Parent, Hammer, River, FinancialTransaction, BuyingAHouse, Automobile, TravelPlanning, etc.**
Ontology Instance (OI) Level: **Object Level to the Ontology Concept Level**	**Harry X. Landsford III, Ralph Waldo Emerson, Person560234, PurchaseOrderTransactionEvent6117090, 1995-96 V-6 Ford Taurus 244/4.0 Aerostar Automatic with Block Casting # 95TM-AB and Head Casting 95TM**

Meta-Level to Object-Level

Meta-Level to Object-Level

Language

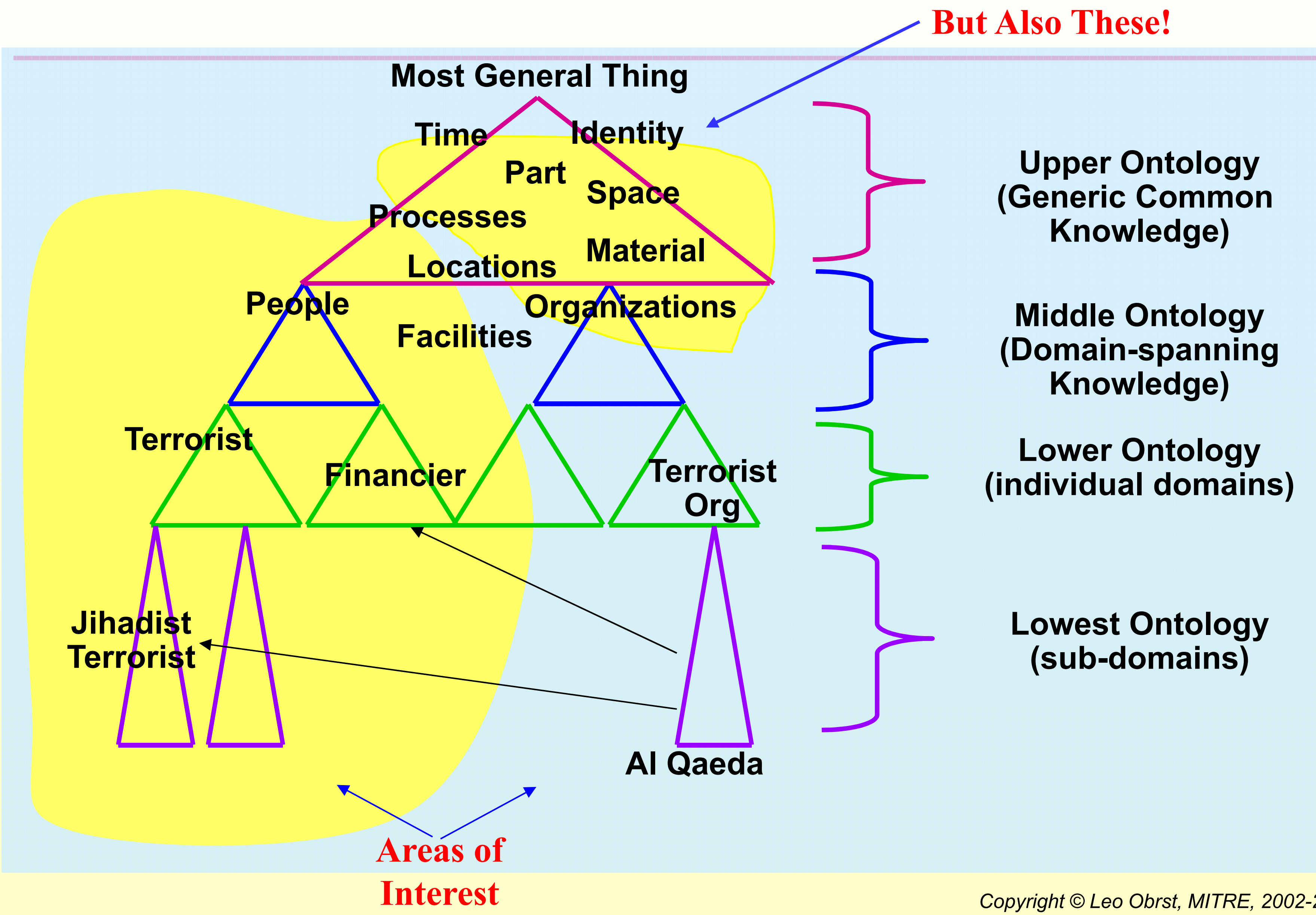
Ontology (General)

Knowledge Base (Particular)

Ontology Example from Electronic Commerce: the general domain of machine tooling & manufacturing; note that these are expressed in English, but usually would be in expressed in a logic-based language

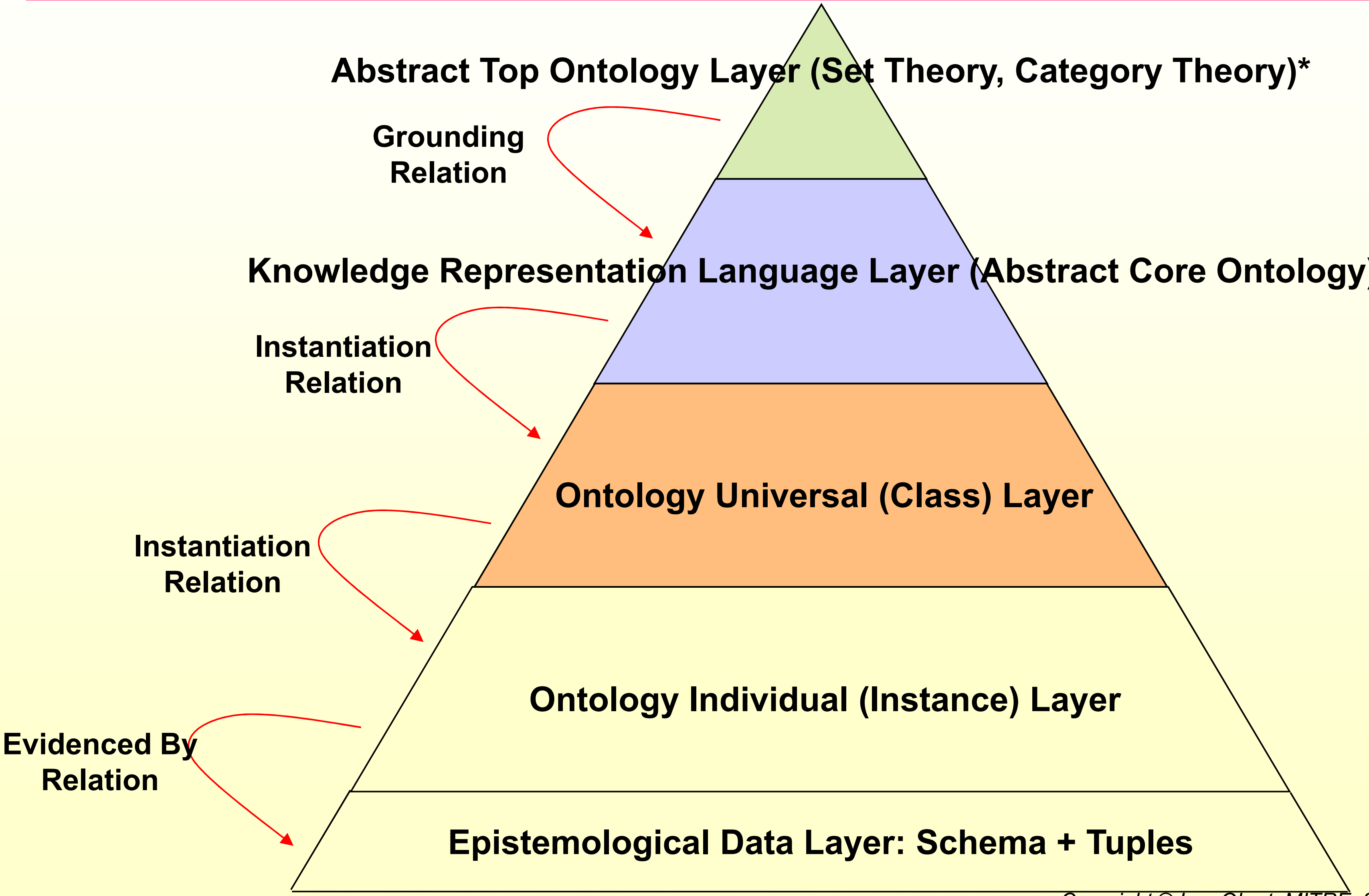
Concept	Example
Classes (general things)	Metal working machinery, equipment and supplies, metal-cutting machinery, metal-turning equipment, metal-milling equipment, milling insert, turning insert, etc.
Instances (particular things)	An instance of metal-cutting machinery is the "OKK KCV 600 15L Vertical Spindle Direction, 1530x640x640mm 60.24"x25.20"x25.20 X-Y-Z Travels Coordinates, 30 Magazine Capacity, 50 Spindle Taper, 20kg 44 lbs Max Tool Weight, 1500 kg 3307 lbs Max Loadable Weight on Table, 27,600 lbs Machine Weight, CNC Vertical Machining Center"
Relations: subclass-of, (kind_of), instance-of, part-of, has-geometry, performs, used-on, etc.	A kind of metal working machinery is metal cutting machinery, A kind of metal cutting machinery is milling insert.
Properties	Geometry, material, length, operation, ISO-code, etc.
Values:	1; 2; 3; "2.5", inches"; "85-degree-diamond"; "231716"; "boring"; "drilling"; etc.
Rules (constraints, axioms)	If milling-insert(X) & operation(Y) & material(Z)=HG_Steel & performs(X, Y, Z), then has-geometry(X, 85-degree-diamond). [Meaning: if you need to do milling on High Grade Steel, then you need to use a milling insert (blade) which has a 85-degree diamond shape.]

Upper, Middle, Domain Ontologies



Ontology Content Architecture: More Complex View

* Adapted from: Herre, Heinrich, and Frank Loebe. 2005. A Meta-ontological Architecture for Foundational Ontologies. In: R. Meersman and Z. Tari (Eds.): CoopIS/DOA/ODBASE 2005, LNCS 3761, pp. 1398–1415, 2005. Springer-Verlag Berlin Heidelberg.



Ontology Lifecycle

4) Analysis 3
- What are the resources available to harvest: vocabularies, schemas, taxonomies, conceptual models, ontologies?
- Are there domain standards, upper/middle ontologies to embed what we create within?

3) Analysis 2
- What are the referents, concepts: entities, relations, properties, rules?
- What are the terms that index the referents: terminology?

Requirements

2) Analysis 1 (Competency Questions)
- ***Bottom-Up***: What are semantics of current data sources?
- ***Top-Down:*** What would you like to ask?

1) Rationale: **Why do you need**

8) Analysis 4
- Refine with domain experts, end users

9) Design 3
- Refine conceptualization

10) Implement 2
- Refine ontology

11) Deploy 1
- Provide ontology application services

12) Deploy 2
- Correct problems

13) Analysis 5
- Interrogate users
- Refine reqs
- More resources?

14) Design 4
- How can changes needed be made?
- Refine reqs

5) Design 1
- What ontology architecture do we choose?
- How expressive is the ontology language we need?
- What conceptualization?
- How do we model these entities, relations, properties, rules?
- What are the instances of these?
- What data sources mappings can link to these? How?
- What kinds of ontology tools do we need?

6) Implement 1
- Implement the ontology server we will need: periodicity, granularity, configuration management
- Implement the infrastructure, services of our architecture: enhance the server with

7) Design 2
- Are we done with ontology development?
ns as queries against
nswers returned quickly
ts/end users?

Ontology Maturity Model

Most Mature

From less to more mature

OMM Level 5

Consistent, pervasive capture of real domain semantics embedded under common middle/upper semantics (axiomatized ontologies); extensive inference

OMM Level 4

Consistent & pervasive capture of real domain semantics, represented as persistent & maintained models (frame ontologies, some axioms); some linkage to upper/middle; some inference supported;

OMM Level 3

Focus is on capture of real domain semantics, mostly represented as persistent & maintained models (frame ontologies); term resources linked to models; database and information extraction routines use some domain ontologies

OMM Level 2

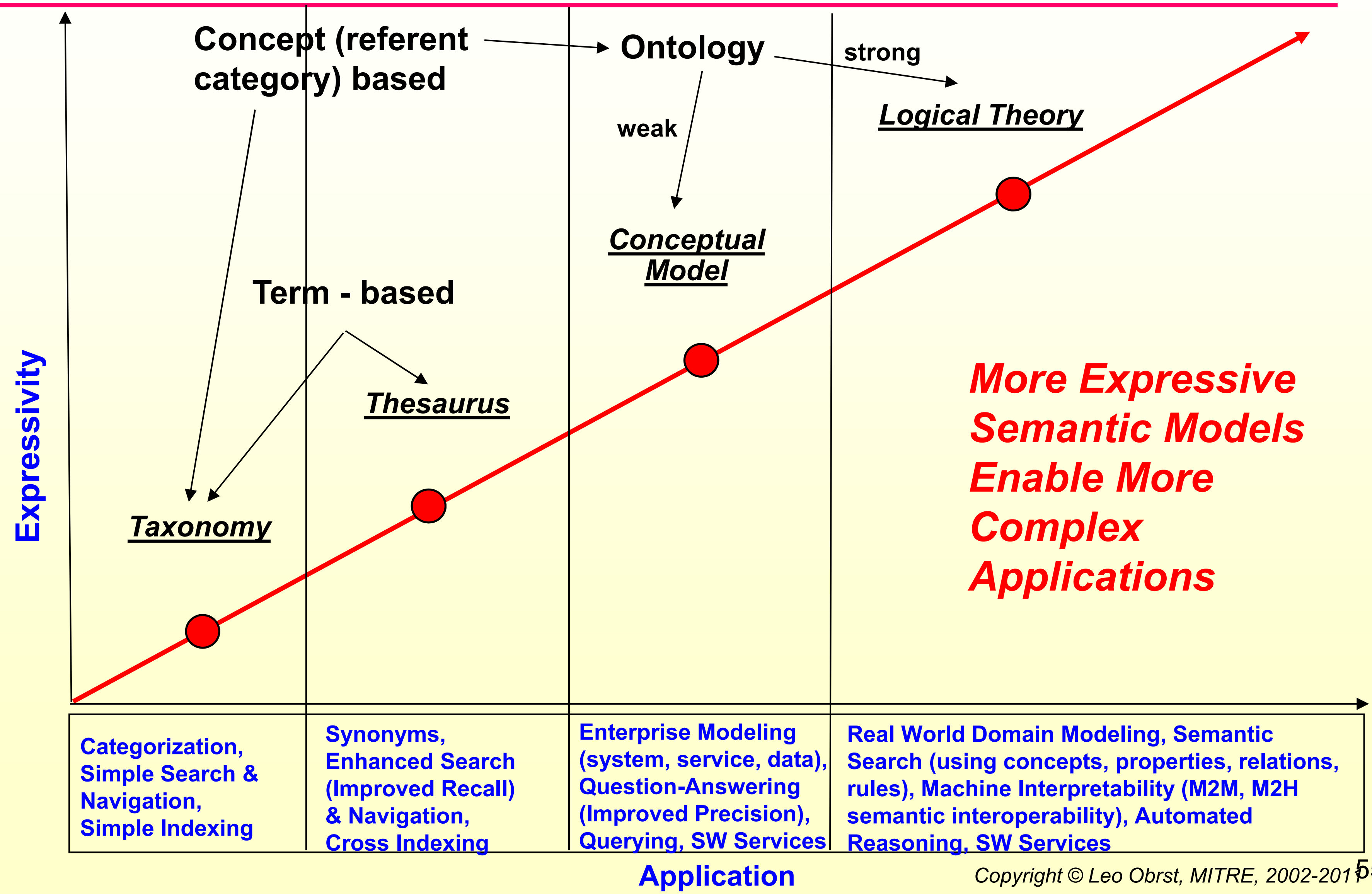
Principled, consistent local semantics captured, some real domain semantics represented as persistent & maintained models (local ontologies); term & concept (referent) distinguished; databases and information extraction routines use local ontologies

OMM Level 1

Mainstream syntactic/structural DB technology (+ data warehouses + data marts), unstructured data addressed by procedural information extraction, no persistent linkage of semantics to syntax/structure, ad hoc local semantics sometimes captured in data dictionary & commented in extraneous code; no clear distinction made between term & concept (referent)

Least Mature

Ontology Spectrum: Complexity of Applications



Recall and Precision

- **Recall**

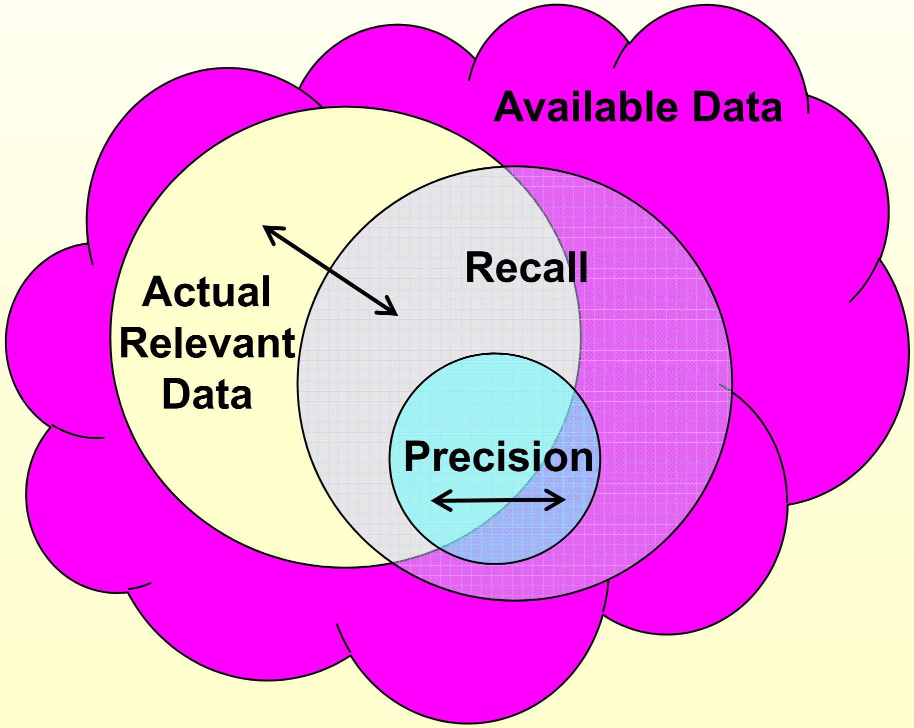
  The percentage of relevant documents retrieved
  Calculation:
  Number of relevant docs retrieved
  Number of relevant docs (i.e., which should have been retrieved)

  For classification:
  The number of true positives
  The number of positives (true positives + false negatives)

- **Precision**

  The percentage of retrieved documents judged relevant
  Calculation:
  Number of relevant docs retrieved
  Number of docs retrieved

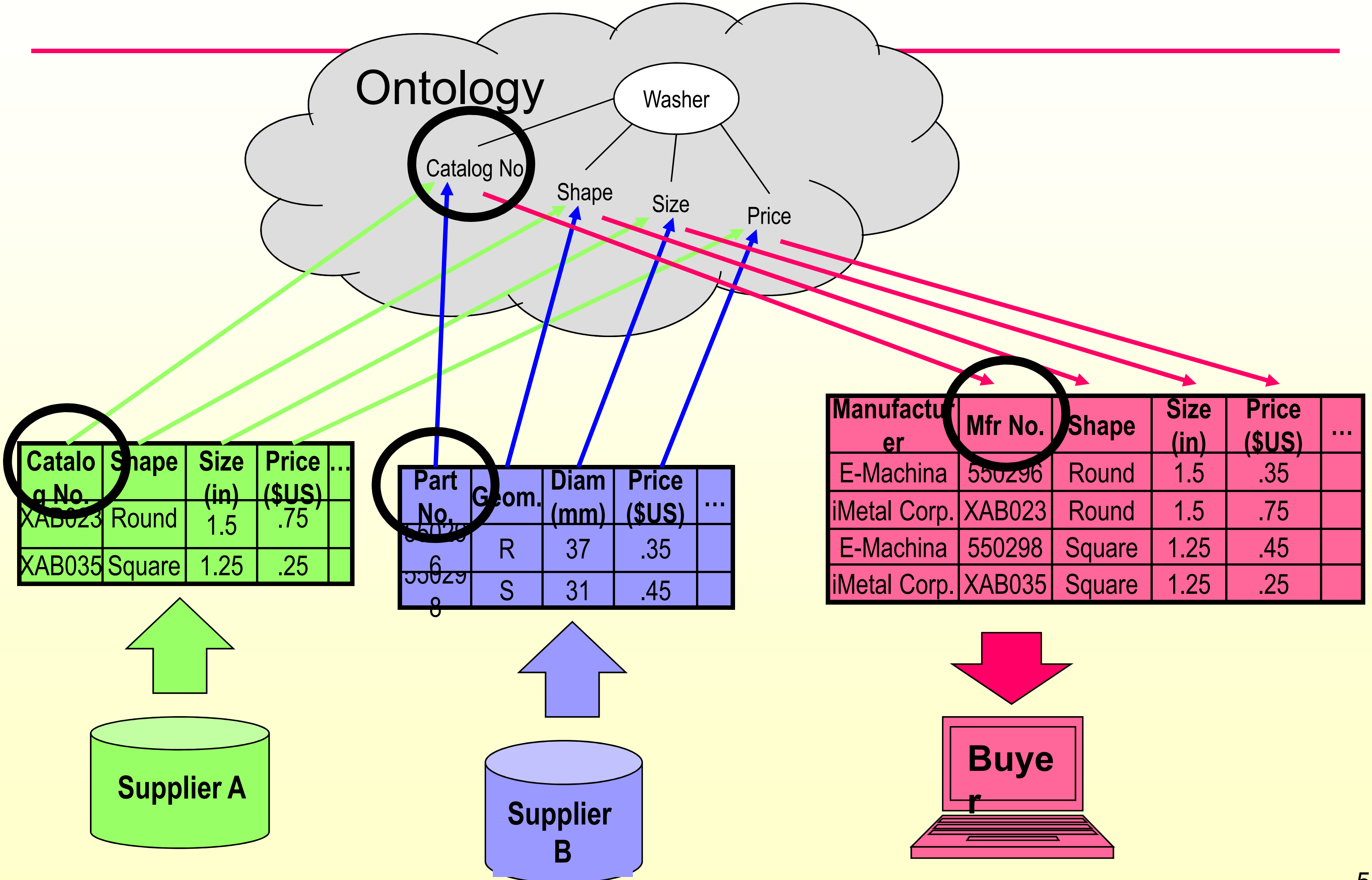
  For classification
  The number of true positives
  The number of positives (true positives + false positives)



What Problems Do Ontologies Help Solve?

- Heterogeneous database problem
  - Different organizational units, Service Needers/Providers have radically different databases
  - Different **syntactically:** what's the format?
  - Different **structurally:** how are they structured?
  - Different **semantically:** what do they mean?
  - They all speak different languages
- Enterprise-wide system interoperability problem
  - Currently: system-of-systems, vertical stovepipes
  - Ontologies act as conceptual model representing enterprise consensus semantics
  - Well-defined, sound, consistent, extensible, reusable, modular models
- Relevant document retrieval/question-answering problem
  - What is the meaning of your query?
  - What is the meaning of documents that would satisfy your query?
  - Can you obtain only meaningful, relevant documents?

A Business Example of Ontology

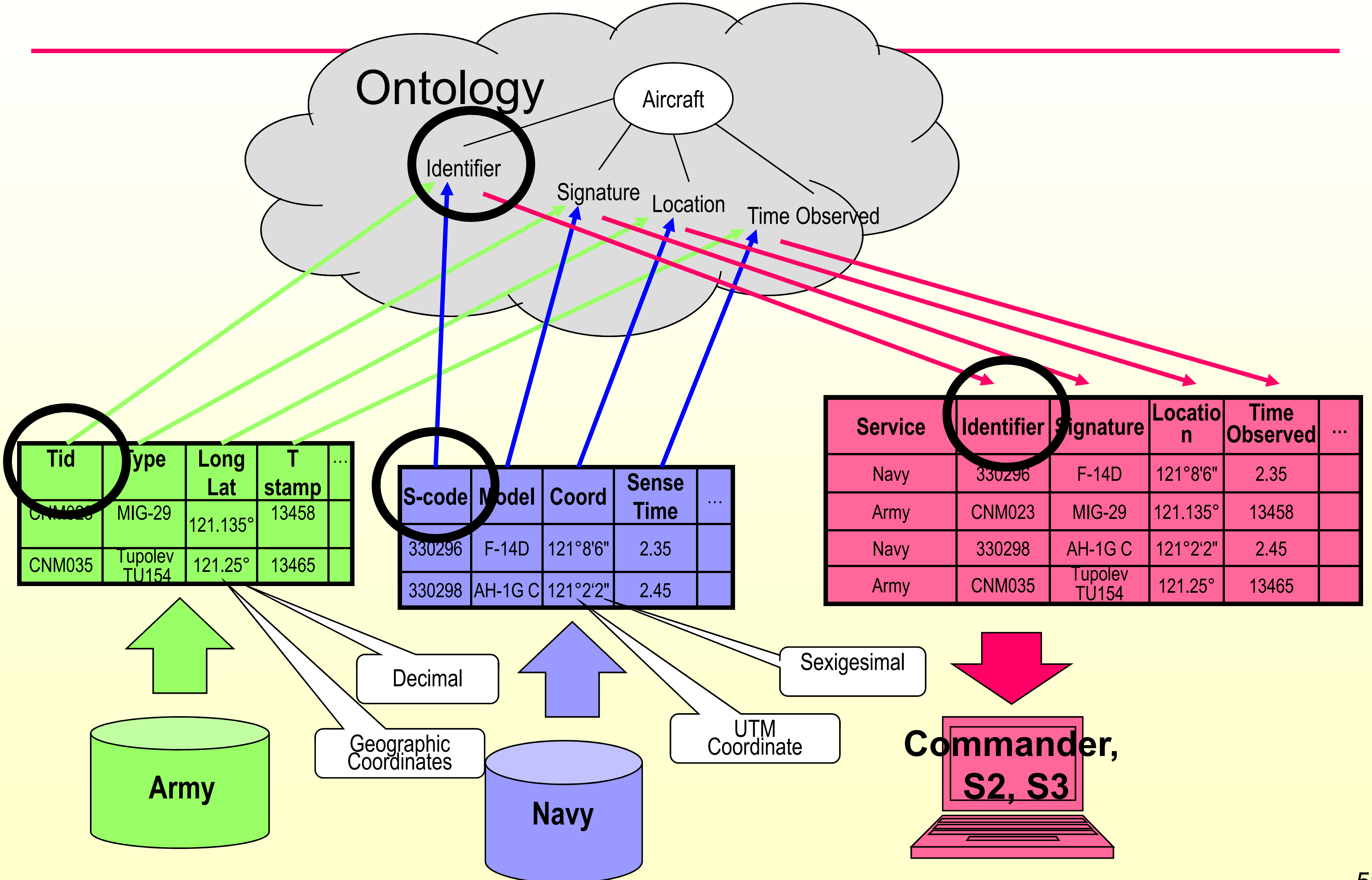


Catalog No.	Shape	Size (in)	Price ($US)	…
XAB023	Round	1.5	.75	
XAB035	Square	1.25	.25	

Part No.	Geom.	Diam (mm)	Price ($US)	…
550296	R	37	.35	
550298	S	31	.45	

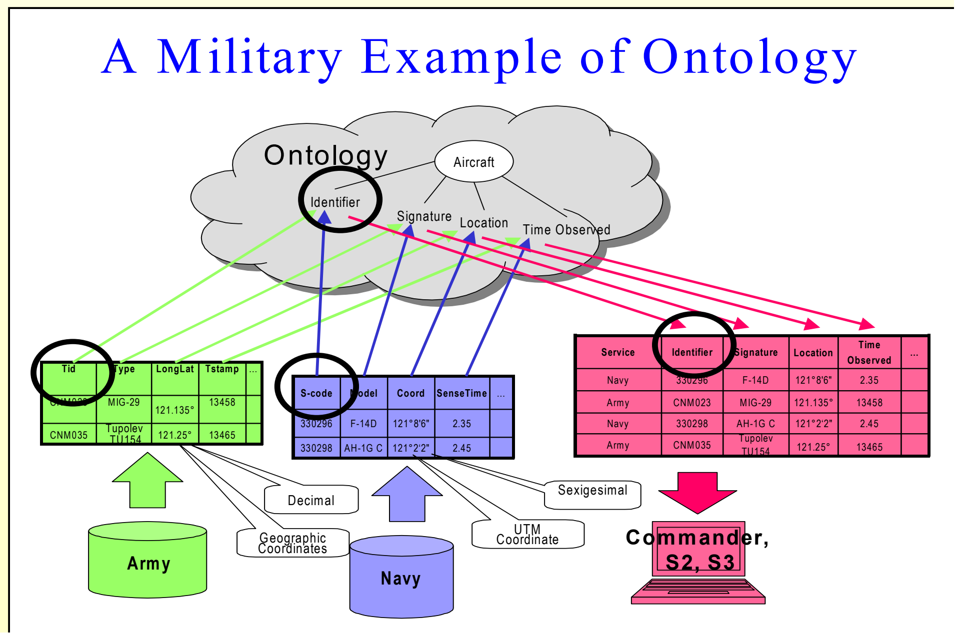
Manufacturer	Mfr No.	Shape	Size (in)	Price ($US)	…
E-Machina	550296	Round	1.5	.35	
iMetal Corp.	XAB023	Round	1.5	.75	
E-Machina	550298	Square	1.25	.45	
iMetal Corp.	XAB035	Square	1.25	.25	

A Military Example of Ontology



Ontologies & the Data Integration Problem

- DBs provide generality of storage and efficient access
- Formal data model of databases insufficiently semantically expressive
- The process of developing a database discards meaning
  - Conceptual model → Logical Model → Physical Model
  - Keys signify some relation, but no solid semantics
  - DB Semantics = Schema + Business Rules + Application Code
- Ontologies can represent the rich common semantics that spans DBs
  - Link the different structures
  - Establish semantic properties of data
  - Provide mappings across data based on meaning
  - Also capture the rest of the meaning of data:
    - Enterprise rules
    - Application code (the inextricable semantics)



Complexity of Semantic Integration with/without Ontologies

- An ontology allows for near linear semantic integration (actually 2n-1) rather than near n^2 (actually n^2 - n) integration
  - Each application/database maps to the "lingua franca" of the ontology, rather than to each other

Ordinary Integration: N^2

A ⇄ B

A ⇄ C

B ⇄ C

2 Nodes	2 Edges
3 Nodes	6 Edges
4 Nodes	12 Edges
5 Nodes	20 Edges

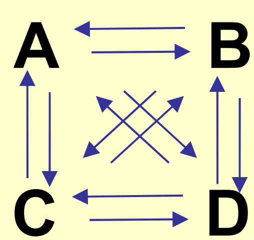
Add D:

A ⇄ D

B ⇄ D

C ⇄ D

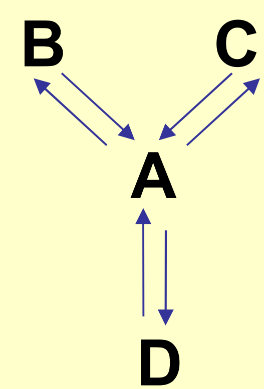
⇒



Ontology Integration: N

B ⇄ A ⇄ C

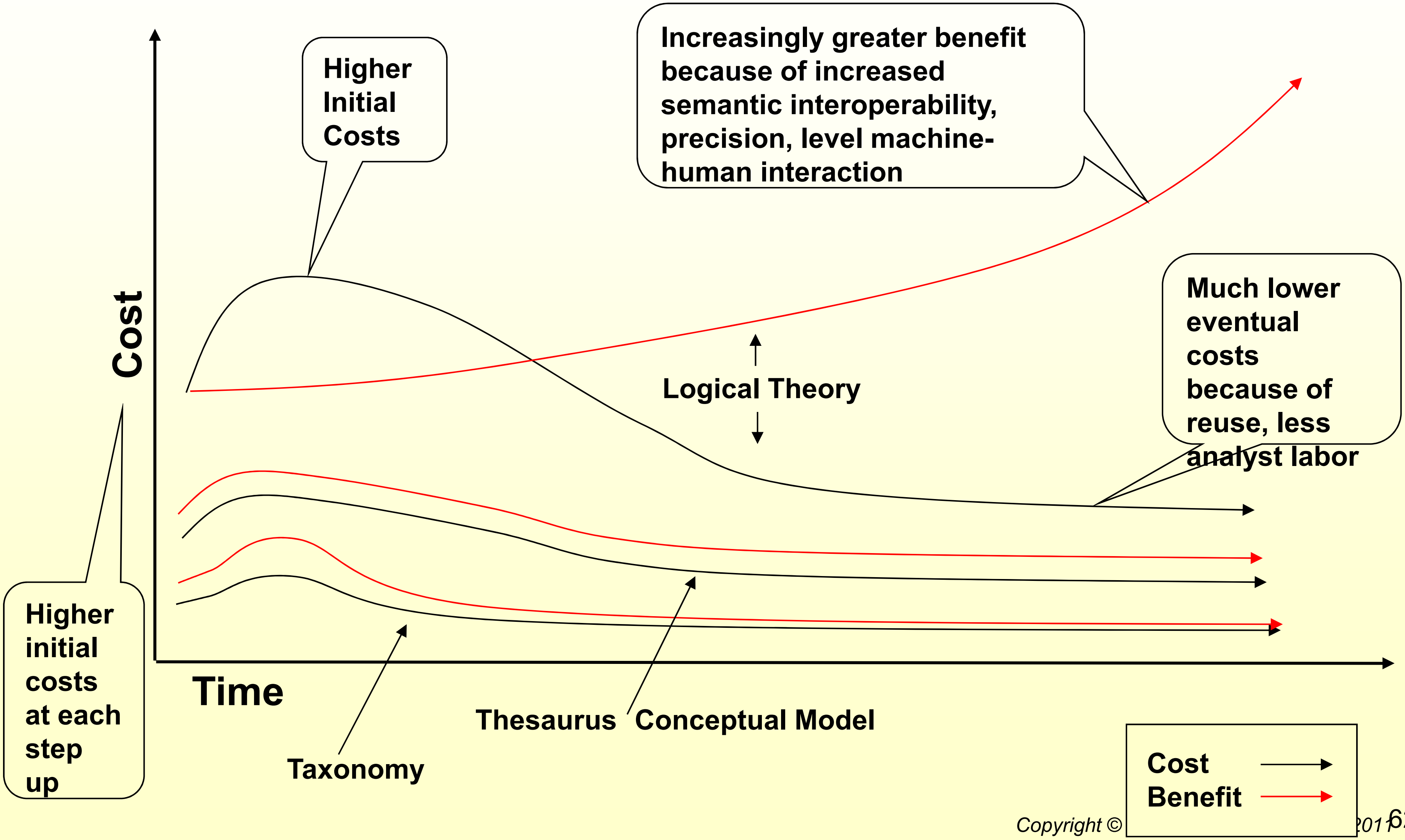
2 Nodes	2 Edges
3 Nodes	4 Edges
4 Nodes	6 Edges
5 Nodes	8 Edges



Add D:

A ⇄ D ⇒

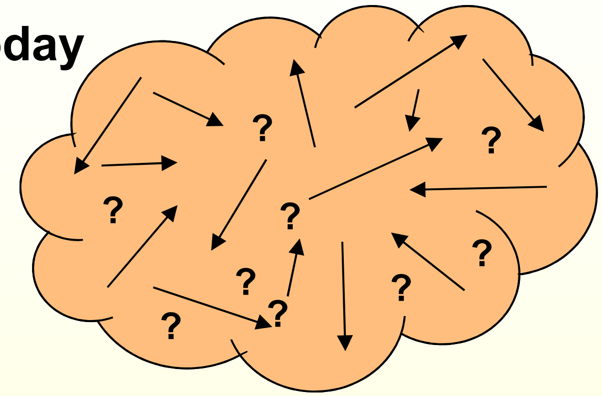
Approximate Cost/Benefit of Moving up the Ontology Spectrum



The Semantic Web

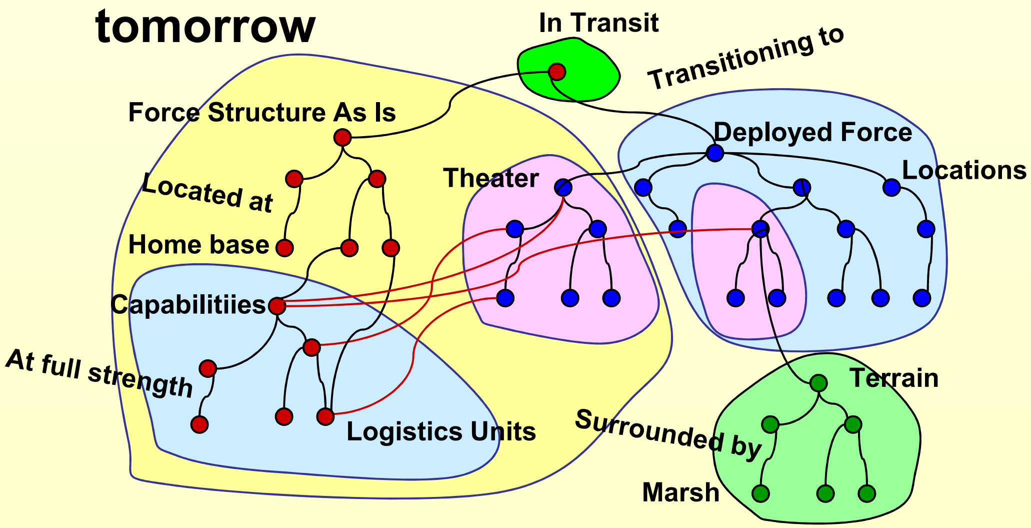
- Current Web is a collection of links and resources
  - Is syntactic & structural only
  - Excludes semantic interoperability at high levels.
  - Google of today is string based (keyword) & has no notion of the semantics (meaning) of your query
- Semantic Web extends the Current Web so information is given well-defined meaning
  - Enables semantic interoperability at high levels
  - Google of tomorrow will be concept based
  - Able to evaluate knowledge in context

Web today



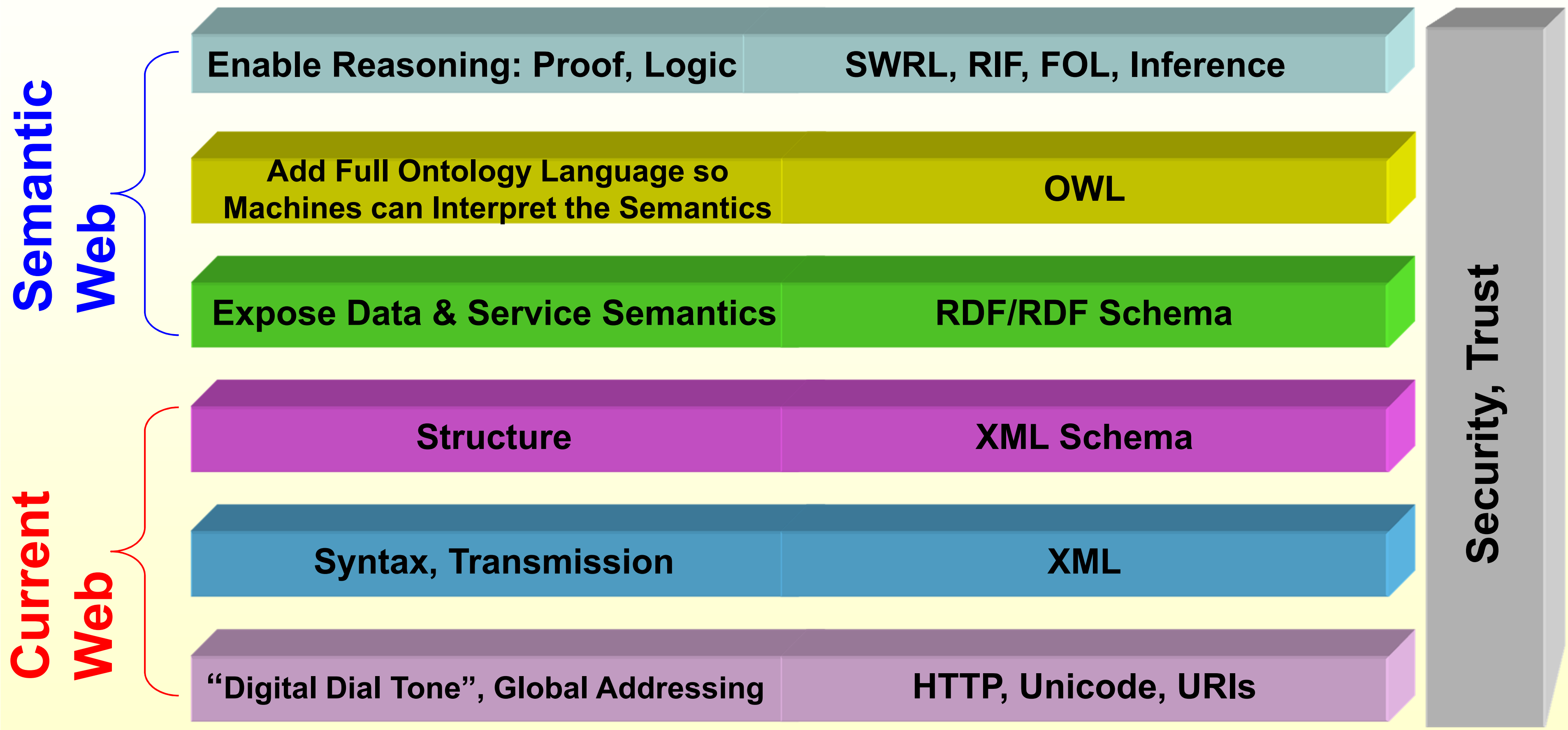
Humans have to do the understanding

Semantic Web tomorrow



Machines partially understand what humans mean

Semantic Web: Another View



- **Anyone, anywhere can add to an evolving, decentralized "global database"**
- **Explicit semantics enable looser coupling, flexible composition of services and data**

Semantic Web Languages

- Numerous efforts have led to recent convergence on W3C recommendations
- 10 Feb '04 W3C released recommendations on
  - Resource Description Framework (RDF)
    - Used to represent information and to exchange knowledge in the Web
  - OWL Web Ontology Language (OWL) as W3C
    - Used to publish and share sets of terms called ontologies, supporting advanced Web search, software agents and knowledge management
  - See http://www.w3.org/ for more information
- RDF and OWL are now international standards
- Both RDF and OWL observe the *Open World Assumption*: new knowledge can always be added to what already exists
- RIF: W3C Recommendation,

What the Languages Provide: RDF/S

- RDFS enables you to make simple, generic statements about your Web object classes, properties
- RDF enables you to make specific statements about your Web object instances (of those classes, properties)
- RDF/S enables you also to make statements about statements (reification), but tells you nothing about those embedded statements
- A set of RDF statements can be viewed in 3 ways:
  - **A set of triples: consider them as rows/tuples in a database**
  - **A directed graph: consider them as a complex, navigatable data structure**
  - **An inference closure over the relations of the graph: consider them as as a machine-interpretable representation of knowledge from which an inference engine can infer new knowledge not expressly encoded**

RDF/S, a spectrum of views: *database row, graph structured object, inference closure*

Resource Description Framework/Schema (RDF/S)

- There is one Language, two levels: RDF is the Language
  - **RDFS** expresses **Class** level relations describing acceptable instance level relations
  - **RDF** expresses **Instance** level semantic relations phrased in terms of a triple:
  - **Statement: <resource, property, value>, <subject, verb, object>, <object1, relation1, object2>**
- *Resources*
  - All things being described by RDF expressions are called resources
    - **An entire Web page such as the HTML document**
    - **Part of a Web page**
    - **A collection of pages**
    - **An object that is not directly accessible via the Web**
  - Always named by URIs plus optional anchor ids
- *Properties*
  - A specific aspect, characteristic, attribute, or relation used to describe a resource
  - Specific meaning
  - Permitted values
  - Relationship with other properties
- *Statements*
  - A specific resource together with a named property plus the value of that property for that resource is an RDF statement

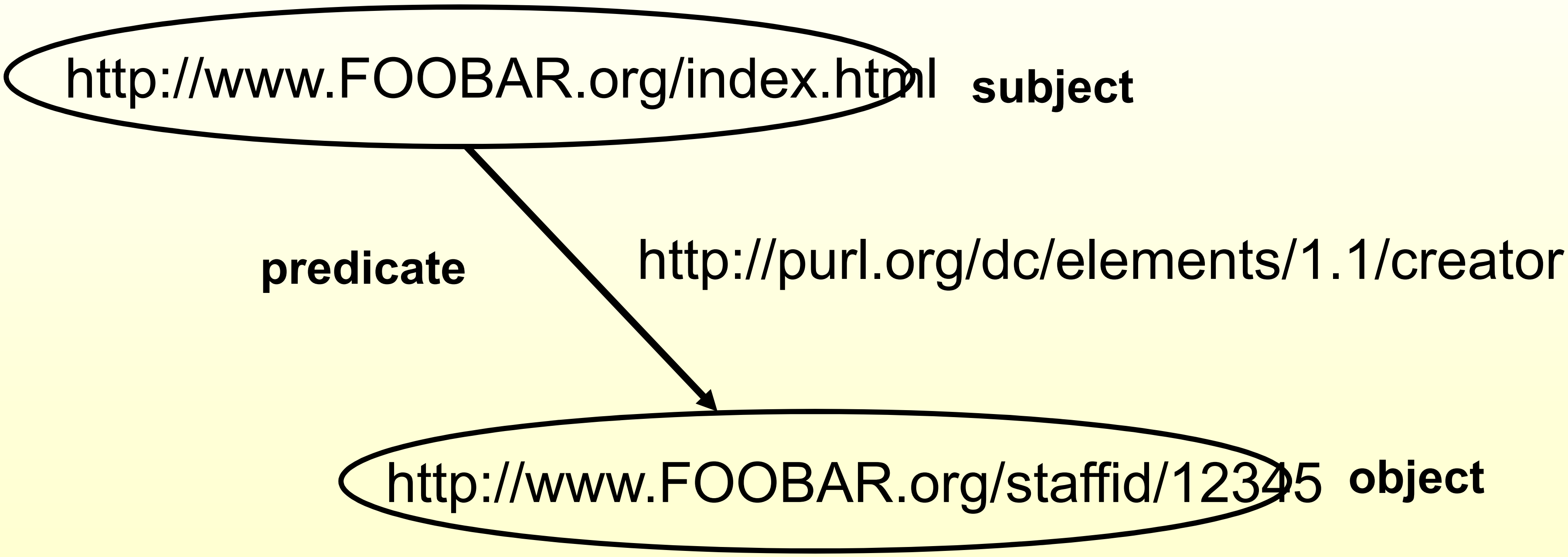
Positive, Existential subset of First Order Logic: no NOT, no ALL: Can’t represent “John is NOT a terrorist”, “All IBMers are overpaid”

RDF/S Model: Statements

- *Statements*
  - A specific resource together with a named property plus the value of that property for that resource is an RDF statement
  - I.e., Triples:
    - <Subject Predicate Object>
    - <Resource Property PropertyValue>
    - <Leo,hasColleague,Jim>
  - PropertyValue can be:
    - another resource (referenced via URI)
    - A literal (primitive datatype defined by XML), i.e., a resource (specified by a URI) or a simple string or other primitive datatype defined by XML

RDF/S Model: A Directed Graph

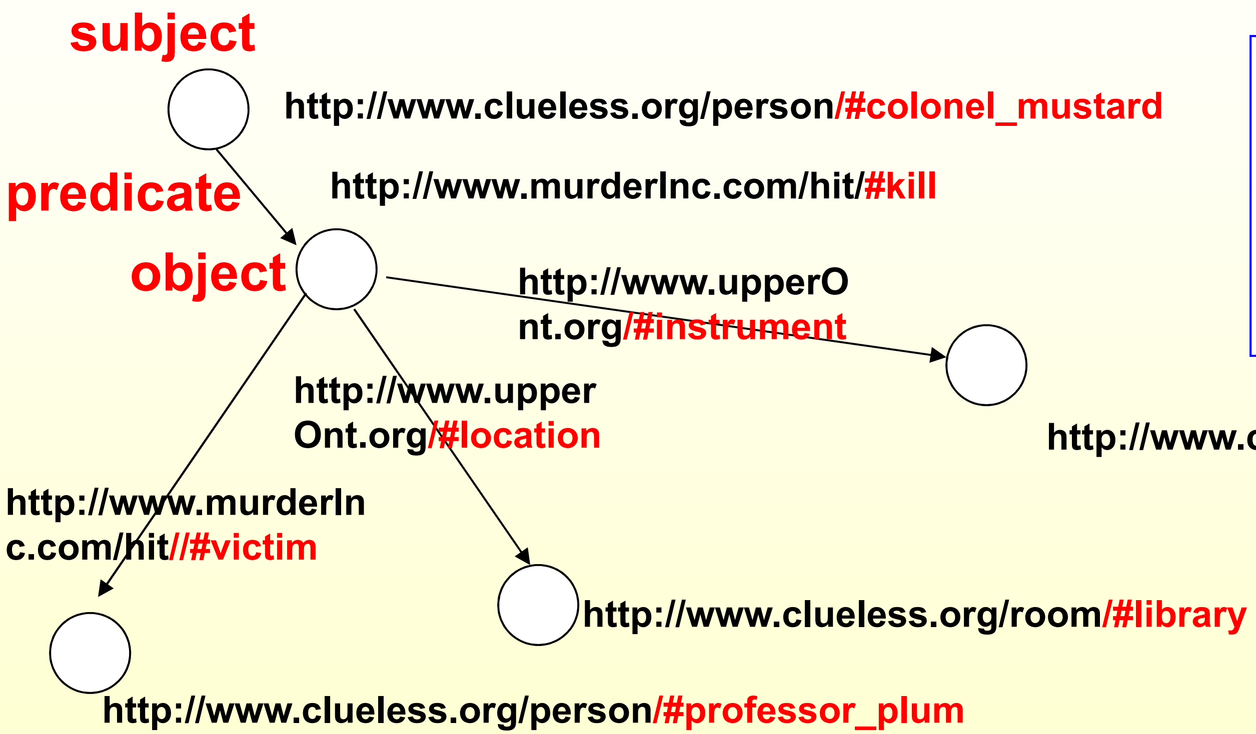
*"The creator of page http://www.FOOBAR.org/index.html is http://www.FOOBAR.org/staffid/12345"



This is also a conceptual graph (with URIs as names)

RDF/S Model: A Directed Graph

Colonel Mustard killed Professor Plum in the Library with the Lead Pipe



- **Predicate: relation or attribute**
- **If the predicate is a relation, then the Object is another "object"**
- **If the predicate is an attribute, then the Object is a "value"**

NOTE: This is also a conceptual graph (with URIs as "names")

<u>Reification:</u> A statement about a statement (but uninterpreted, no truth asserted):
John thinks X, where X = "Colonel Mustard killed Professor Plum in the Library with the Lead Pipe"; don't know what X 'means'

What the Languages Provide: OWL

- OWL enables you to make complex, generic statements about your Web object classes, properties
- OWL's instances are expressed as RDF statements
- OWL has 3 dialects/layers, increasingly more complex: **OWL-Lite, OWL-DL, OWL-Full**
- OWL is only an ONTOLOGY language (like RDFS) & a Description Logic (classification via subsumption)
- OWL uses everything below it in the Semantic Web stack:
  - **Has a presentation/exchange XML syntax, XML datatypes**
  - **RDF instances**
  - **RDFS generic (ontology) statements: how depends on the OWL dialect**
  - **OWL is expressed in an XML exchange and presentation syntax**
- OWL enables you to map among ontologies:
  - **Import one ontology into another: all things that are true in the imported ontology will thereby be true in the importing ontology**
  - **Assert that a class, property, or instance in one ontology/knowledge base is equivalent to one in another ontology**

OWL Language Levels*

Language Level	Description
OWL Full	The complete OWL. For example, a class can be considered both as a collection of instances (individuals) and an instance (individual) itself.
OWL DL (description logic)	Slightly constrained OWL. Properties cannot be individuals, for example. More expressive cardinality constraints.
OWL Lite	A simpler language but one that is more expressive than RDF/S. Simple cardinality constraints only (0 or 1).

*Daconta, Obrst, Smith, 2003; cf. also OWL docs at http://www.w3.org/2001/sw/WebOnt/

OWL LITE

- OWL Lite enables you to define an ontology of classes and properties and the instances (individuals) of those classes and properties
- This and all OWL levels use the *rdfs:subClassOf* relation to defined classes that are subclasses of other classes and which thus inherit those parent classes properties, forming a subsumption hierarchy, with multiple parents allowed for child classes
- Properties can be defined using the *owl:objectProperty* (for asserting relations between elements of distinct classes) or *owl:datatypeProperty* (for asserting relations between class elements and XML datatypes)*, owl:subproperty, owl:domain,* and *owl:range* constructs

*Daconta, Obrst, Smith, 2003; cf. also OWL docs at http://www.w3.org/2001/sw/WebOnt/

OWL DL

- OWL DL extends OWL Lite by permitting cardinality restrictions that are not limited to 0 or 1

- Also, you can define classes based on specific property values using the *hasValue* construct

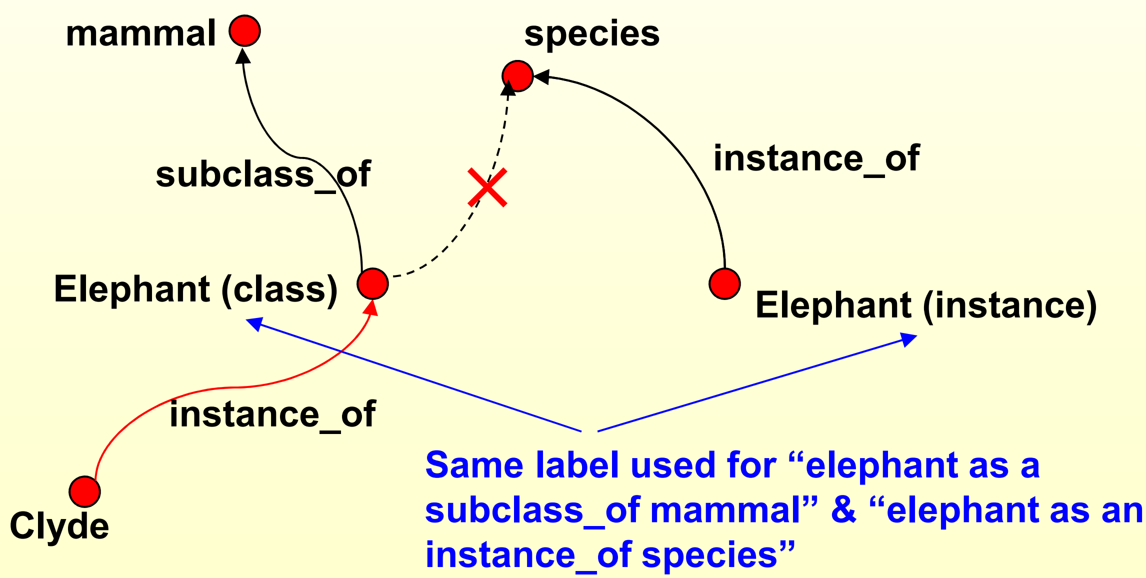
- At the OWL DL level, you can create *class expressions* using Boolean combinators (set operators) such as *unionOf, intersectionOf,* and *complementOf*

- Furthermore, classes can be enumerated (listed) using the *oneOf* construct or specified to be disjoint using *disjointWith* construct

*Daconta, Obrst, Smith, 2003; cf. also OWL docs at http://www.w3.org/2001/sw/WebOnt/

OWL FULL

- OWL Full extends OWL DL by permitting classes to be treated simultaneously as both collections and individuals (instances)
- Also, a given *datatypeProperty* can be specified as being *inverseFunctional*, thus enabling, for example, the specification of a string as a unique key



****Clyde is an elephant.**

Elephant is a species.

Therefore, Clyde is a species.

WRONG!

Clyde is an elephant.

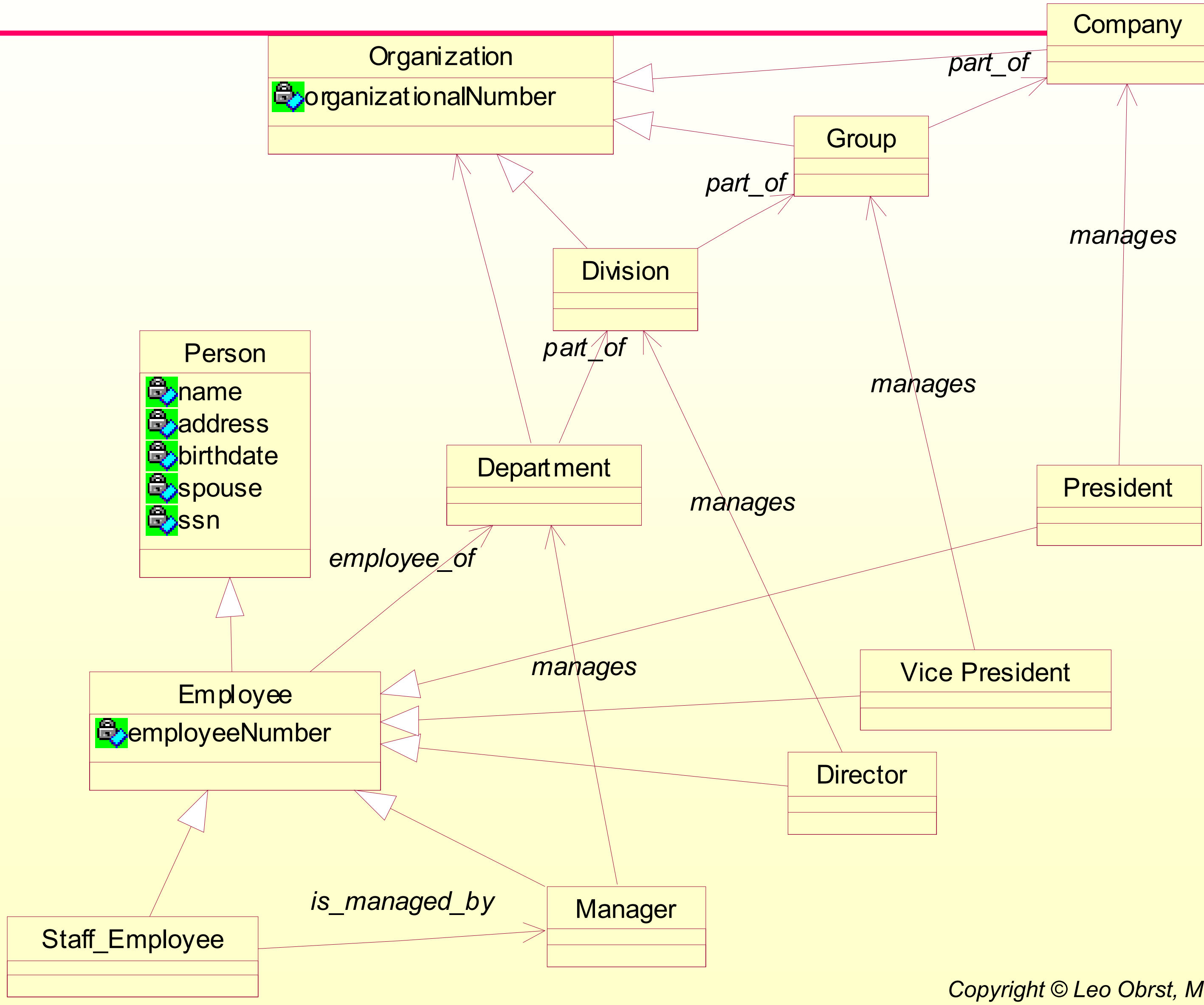
Elephant is a mammal.

Therefore, Clyde is a mammal.

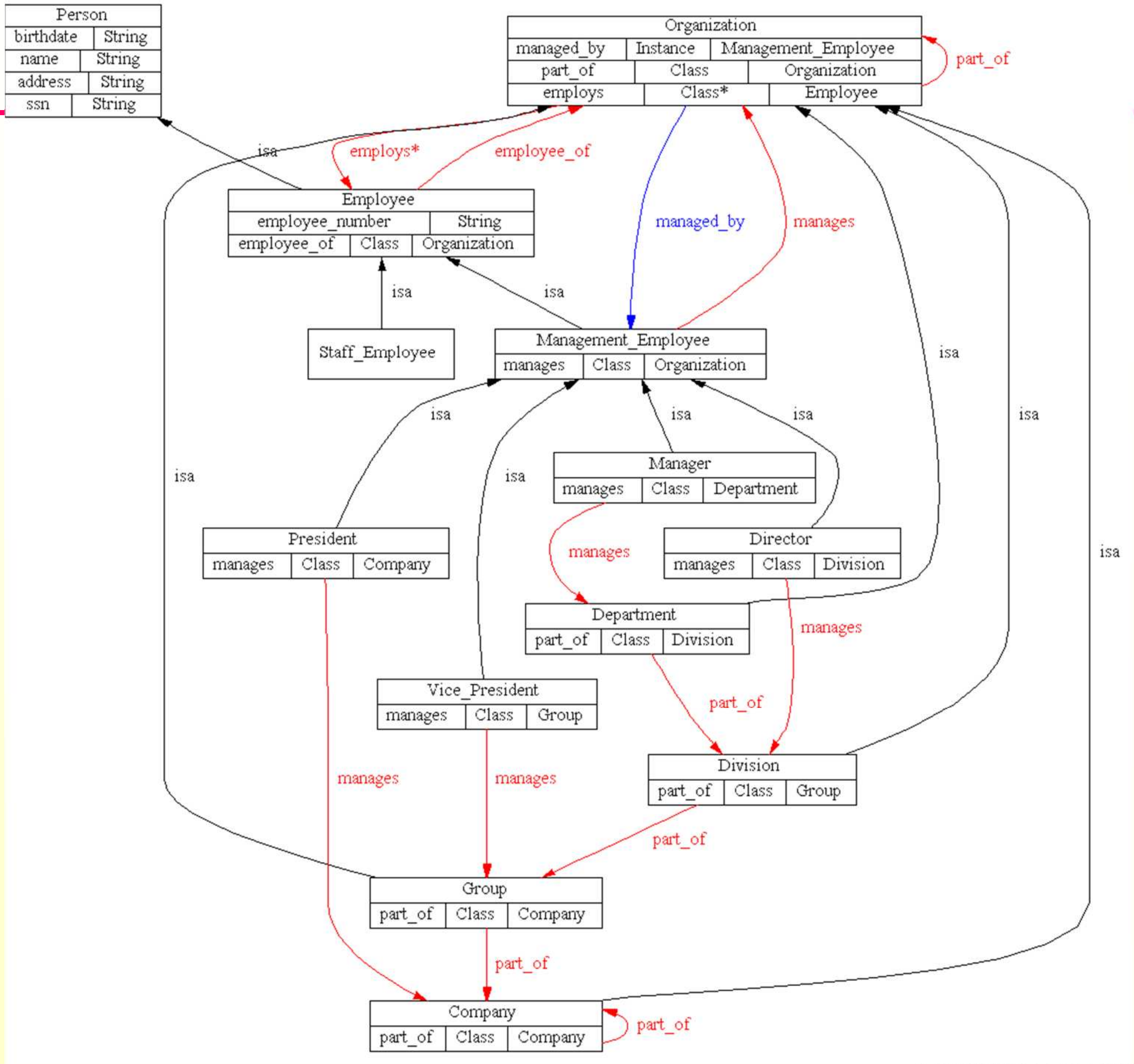
RIGHT!

*Daconta, Obrst, Smith, 2003; cf. also OWL docs at http://www.w3.org/2001/sw/WebOnt/
**Sowa, John. 2000. Knowledge Representation: Logical, Philosophical, and Computational Foundations. Pacific Grove, CA: Brooks/Cole Thomson Learning.

Human Resource Model in UML



Human Resource Ontology in Protégé



OWL Human Resource Ontology Fragment

- Define a class called *Management_Employee* (1), then a subclass of that class, called *Manager* (2), and finally, an instance of the Manager class – *JohnSmith* (3)
  - The *subclass* relation is *transitive,* meaning that *inheritance* of properties from the parent to the child (subclass of parent) is enabled
  - So a *Manager* inherits all the properties defined for its superclass *Management_Employee*

```
1. <owl:Class rdf:ID="Management_Employee">
2. <owl:Class rdf:ID="Manager">
        <rdfs:subClassOf
  rdf:resource="#Management_Employee"/>
   </owl:Class>
3. <Manager rdf:ID="JohnSmith" />
```

- Define the property *employs* with domain *Organization* and range, *Employee*

```
<owl:ObjectProperty rdf:ID="employs">
  <rdfs:domain rdf:resource="#Organization"/>
  <rdfs:range rdf:resource="#Employee"/>
 </owl:ObjectProperty>
```

OWL Human Resource Ontology Fragment

- Define property *employee_of* with domain *Employee*, range *Organization*

```
<owl:ObjectProperty rdf:ID="employee_of">
   <rdfs:domain rdf:resource="#Employee"/>
   <rdfs:range rdf:resource="#Organization"/>
 </owl:ObjectProperty>
```

- *employee* and *employee_of* are inverses of each other
- In OWL, this inverse relation can be stated in a different way, with the same semantics

```
<owl:ObjectProperty rdf:ID="employee_of">
   <owl:inverseOf rdf:resource="#employs" />
 </owl:ObjectProperty>
```

OWL Wine Ontology: Snippets*

- **Header, Namespace information**

```
<owl:Ontology rdf:about=""> <rdfs:comment>An example OWL
  ontology</rdfs:comment> <owl:priorVersion
  rdf:resource="http://www.w3.org/TR/2003/PR-owl-guide-20031215/wine"/>
  <owl:imports rdf:resource="http://www.w3.org/TR/2004/REC-owl-guide-
  20040210/food"/> <rdfs:label>Wine Ontology</rdfs:label>  …
```

- **Three Root Classes**

```
<owl:Class rdf:ID="Winery"/>
<owl:Class rdf:ID="Region"/>
<owl:Class rdf:ID="ConsumableThing"/>
```

- **Define a Subclass**

```
<owl:Class rdf:ID="PotableLiquid"> <rdfs:subClassOf
  rdf:resource="#ConsumableThing" /> ... </owl:Class>
```

- **Define an Individual (Instance)**

```
<owl:Thing rdf:ID="CentralCoastRegion" /> <owl:Thing
  rdf:about="#CentralCoastRegion"> <rdf:type rdf:resource="#Region"/>
  </owl:Thing>
```

- **Define a property**

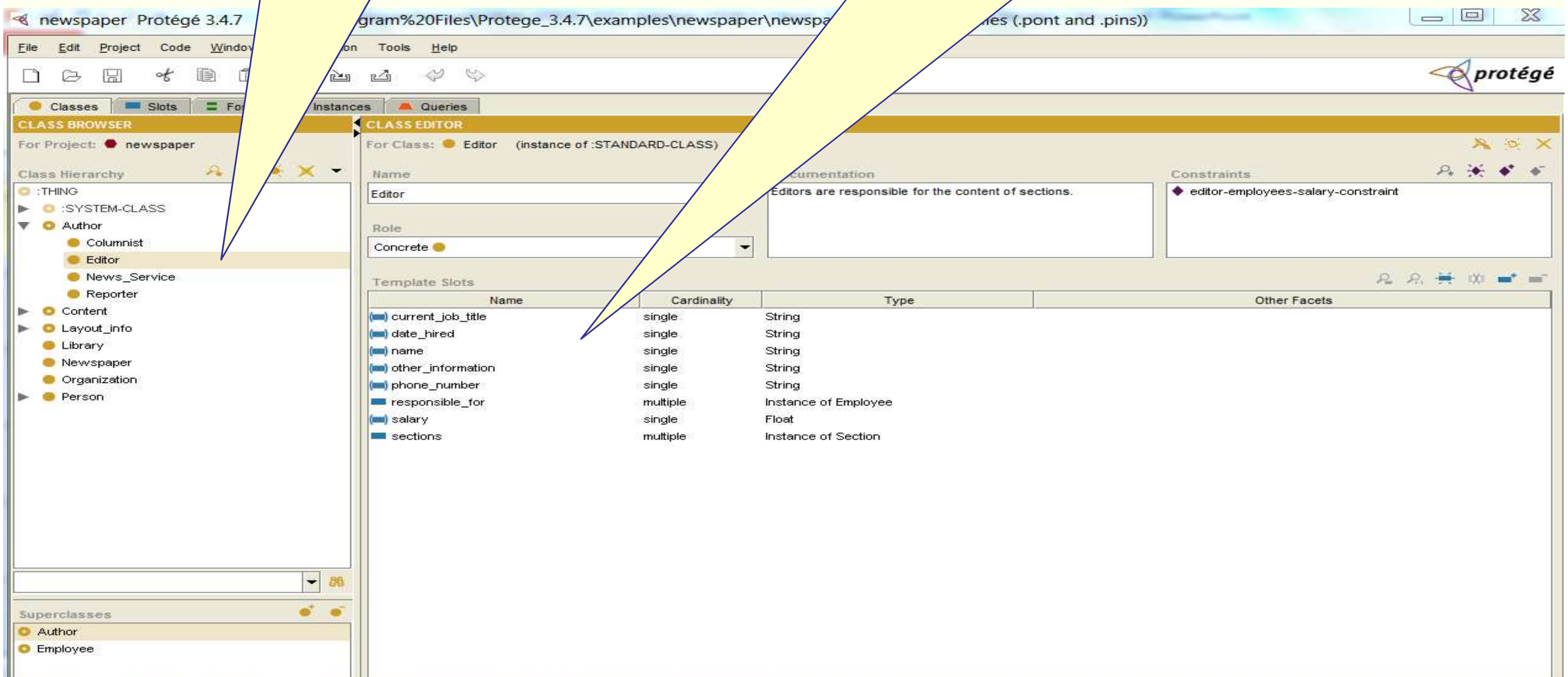
```
<owl:ObjectProperty rdf:ID="madeFromGrape"> <rdfs:domain
  rdf:resource="#Wine"/> <rdfs:range rdf:resource="#WineGrape"/>
  </owl:ObjectProperty>
```

*** From the OWL Guide, http://www.w3.org/TR/2004/REC-owl-guide-20040210/**

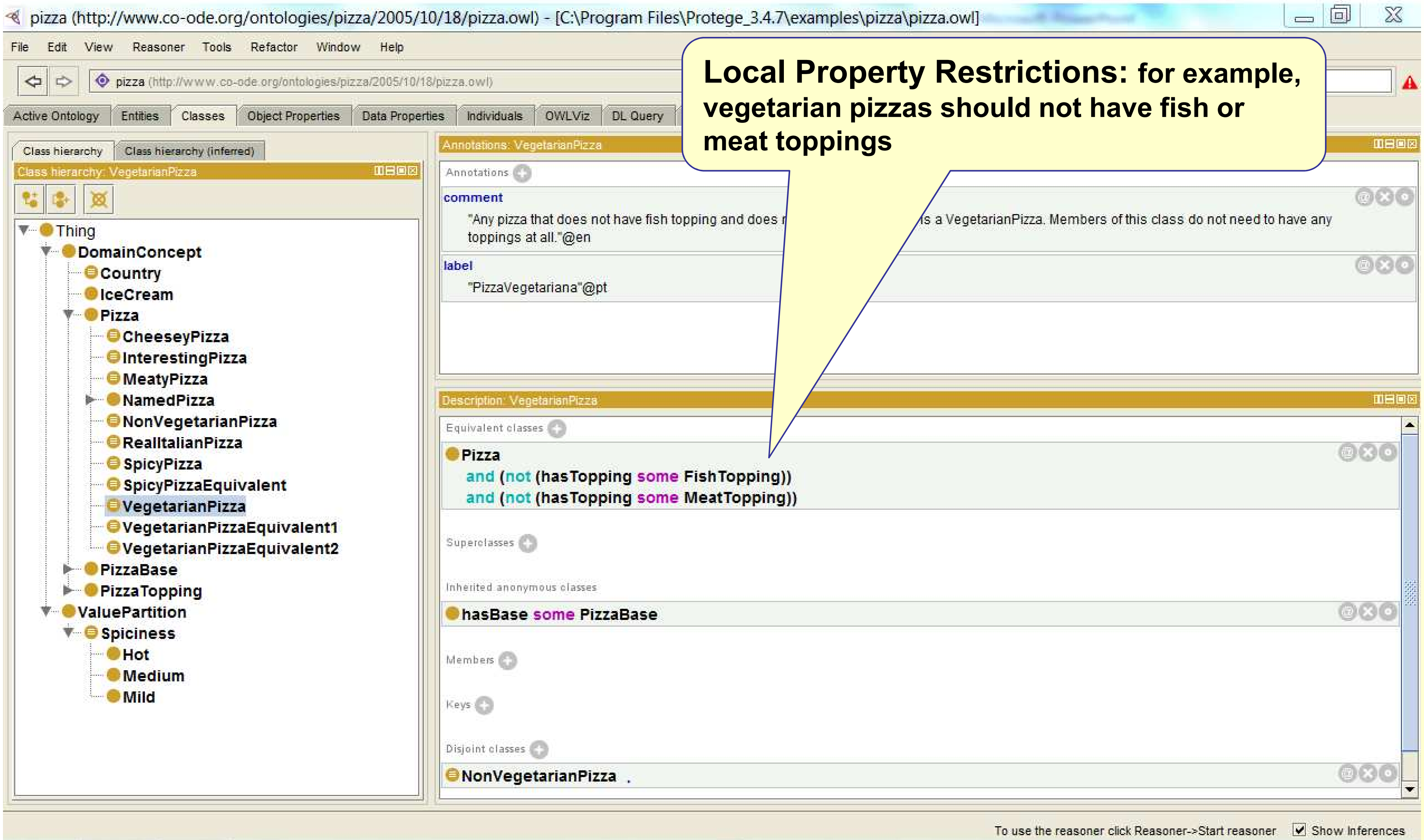
Protégé 3.4+ Newspaper Example: http://protege.stanford.edu/

Class hierarchy: for example, columnists, editors, reporters, and news services are authors

Slot descriptions: for example, editors have names, phone numbers, salaries; they are also responsible for other employees and contents of sections



Protégé 4.1: OWL Pizza Ontology



Local Property Restrictions: for example, vegetarian pizzas should not have fish or meat toppings

OWL 2 (1)

- OWL 2 is a W3C Recommendation (27 Oct 2009)*
- Compatible with OWL 1 (04 Feb 2004)
- New features
  - Increased datatype coverage: Designed to take advantage of the new datatypes and clearer explanations available in XSD 1.1 (not yet a recommendation)
  - Syntactic Sugar for more easily saying things in OWL:
  - New constructs that increase expressivity
  - Simple meta-modeling capabilities
  - Extended annotation capabilities
  - Profiles

*** http://www.w3.org/TR/owl2-new-features//**

OWL 2 (2)

- Syntactic Sugar for more easily saying things in OWL:
  - DisjointUnion:
    - DisjointUnion(:CarDoor :FrontDoor :RearDoor :TrunkDoor) : A :CarDoor is exclusively either a :FrontDoor, a :RearDoor or a:TrunkDoor and not more than one of them.
  - DisjointClasses
    - DisjointClasses(:LeftLung :RightLung) : Nothing can be both a :LeftLung and a :RightLung.
  - NegativeObject(Data)PropertyAssertion
    - NegativeObjectPropertyAssertion(:livesIn :ThisPatient :IleDeFrance) :ThisPatient does not live in the :IleDeFrance region.
  - Self-restriction on Properties: “local reflexivity”
    - SubClassOf(:AutoRegulatingProcess ObjectHasSelf(:regulate)): Auto-regulating processes regulate themselves.
  - Property Qualified Cardinality Restrictions: counted cardinality restrictions (Min, Max, Exact)
    - ObjectMaxCardinality(3 :boundTo :Hydrogen): Class of objects bound to at most three different :Hydrogen
  - Many others

OWL 2 (3)

- Simple meta-modeling capabilities:
  - Punning: allows different uses of the same term and an individual
  - OWL 2 DL still imposes certain restrictions: it requires that a name cannot be used for both a class and a datatype and that a name can only be used for one kind of property; semantically names are distinct for reasoners
- Annotations:
  - AnnotationAssertion: for annotation of ontology entities
  - Annotation: for annotations of axioms and ontologies
  - Etc.
- New constructs that increase expressivity
  - Declarations: a declaration signals that an entity is part of the vocabulary of an ontology. A declaration also associates an entity category (class, datatype, object property, data property, annotation property, or individual) with the declared entity
  - Declaration(NamedIndividual(*:Peter*)): *Peter* is declared to be an individual

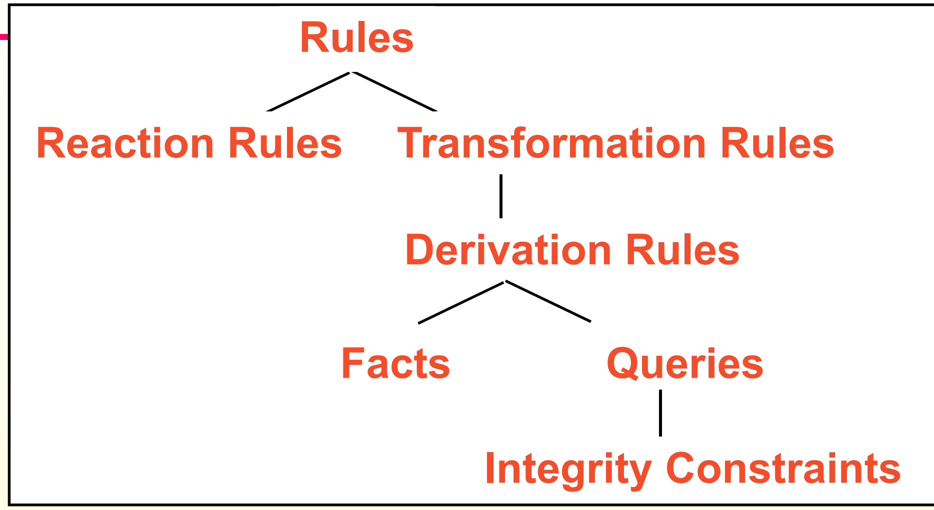
OWL 2 (4)

- Profiles:
  - OWL 1 defined two major dialects, OWL DL and OWL Full, and one syntactic subset (OWL Lite)
  - Needs:
    - Some large-scale applications (e.g., in the life sciences) are mainly concerned with language scalability and reasoning performance problems and are willing to trade off some expressiveness in return for computational guarantees, particularly w.r.t. classification
    - Other applications involve databases and so need to access such data directly via relational queries (e.g., SQL)
    - Other applications are concerned with interoperability of the ontology language with rules and existing rule engines
  - Therefore, 3 profiles (sublanguages, i.e., syntactic subsets of OWL 2) are defined: **OWL 2 EL, OWL 2 QL, and OWL 2 RL***
- And more!

***http://www.w3.org/TR/owl2-profiles/**

Semantic Web Rules: RuleML, SWRL (RuleML + OWL), RIF

RuleML Rule Taxonomy*



*Adapted from Harold Boley, Benjamin Grosof, Michael Sintek, Said Tabet, Gerd Wagner. 2003. RuleML Design, 2002-09-03: Version 0.8. http://www.ruleml.org/indesign.html

- ***Reaction rules*** **can be reduced to general rules that return no value. Sometimes these are called "condition-action" rules. Production rules in expert systems are of this type**
- ***Transformation rules*** **can be reduced to general rules whose 'event' trigger is always activated. A Web example of transformation rules are the rules expressed in XSLT to convert one XML representation to another. "Term rewrite rules" are transformation rules, as are ontology-to-ontology mapping rules**
- ***Derivation rules*** **can be reduced to transformation rules that like characteristic functions on success just return true. Syntactic A** $|-_P$ **B and Semantic Consequence A** $|=_P$ **B are derivation rules**
- ***Facts*** **can be reduced to Facts can be reduced to derivation rules that have an empty (hence, 'true') conjunction of premises. In logic programming, for example, facts are the ground or instantiated relations between "object instances"**
- ***Queries*** **can be reduced to derivation rules that have – similar to refutation proofs – an empty (hence, 'false') disjunction of conclusions or – as in 'answer extraction' – a conclusion that captures the derived variable bindings**
- ***Integrity constraints*** **can be reduced to queries that are 'closed' (i.e., produce no variable bindings)**

So Which Rules Are Useful, Good, Bad, Ugly?

☺ Good

- Logical rules are declarative, confirmable by human beings, machine semantically-interpretable, non-side-effecting
- Logical rules can express everything that production (expert system) rules, procedural rules can
- Logical rules can express business, policy rules, static/dynamic rules

☠ Bad

- Rules expressed in procedural code if-then-else case statements are non-declarative, inspectable by human beings, confirmable with documentation and observance of conformance to documentation, side-effecting (ultimate side-effect: negating a value and returning true for that value)

☹ Ugly

- Expert systems rules “simulate” inference, are pre-logical, have side-effects, tend toward non-determinism, force all knowledge levels to the same level (this is why ontologies and ontological engineering came about), are horrible to debug

Rule Interchange Format (RIF)*

- RIF is a rule language based on XML syntax
- RIF provides multiple versions, called *dialects:*
  - **Core:** the fundamental RIF language, and a common subset of most rule engines (It provides "safe" positive datalog with builtins)
  - **BLD (Basic Logic Dialect**): adds to Core: logic functions, equality in the *then*-part, and named arguments (This is positive Horn logic, with equality and builtins)
  - **PRD (Production Rules Dialect):** adds a notion of forward-chaining rules, where a rule *fires* and then performs some action, such as adding more information to the store or *retracting* some information (This is comparable to production rules in expert systems, sometimes called condition-action, event-condition-action, or reaction rules)
- RIF SPARQL, triple-store, reasoners:
  - http://www.w3.org/2005/rules/wiki/Implementations

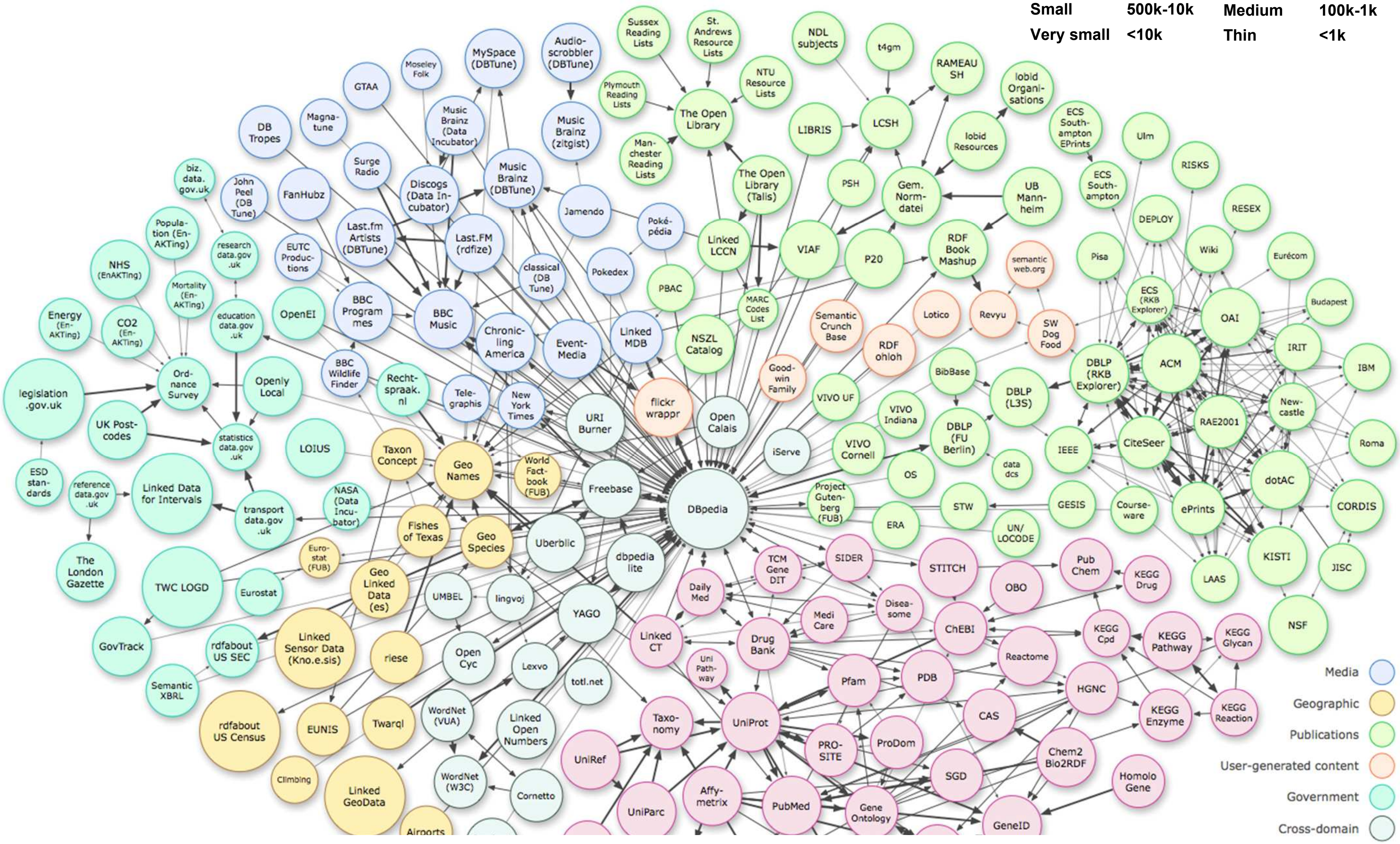
•**http://www.w3.org/2005/rules/wiki/RIF_Working_Group**
•**http://www.w3.org/2005/rules/wiki/RIF_FAQ**

Linked Data

http://linkeddata.org/, http://richard.cyganiak.de/2007/10/lod/

Circle size	Triple
Very large	>1B
Large	1B-10M
Medium	10M-500k
Small	500k-10k
Very small	<10k

Arrow thickness	Triple count
Thick	>100k
Medium	100k-1k
Thin	<1k



In May, 2009, 4.7 billion RDF triples, interlinked by around 142 million RDF links, reported by W3C's Linking Open Data Project In Sept, 2010, the new diagram contained 203 linked datasets which together serve 25 billion RDF triples to the Web and were interconnected by 395 million RDF links.

Where is the Technology Going?

- Not quite there: "The Semantic Web is very exciting, and now just starting off in the same grassroots mode as the Web did 10 years ago ... In 10 years it will in turn have revolutionized the way we do business, collaborate and learn."
  - Tim Berners-Lee, CNET.com interview, 2001-12-12
- We can look forward to:
  - Semantic Integration/Interoperability, not just data interoperability
  - Applications and services with trans-community semantics
  - Device interoperability in the ubiquitous computing future: achieved through semantics & contextual awareness
  - True realization of intelligent agent interoperability
  - Intelligent semantic information retrieval & search engines
  - Next generation semantic electronic commerce/business & web services
  - Semantics beginning to be used once again in NLP

Key to all of this is effective & efficient use of explicitly represented semantics (ontologies)

The Point (s)

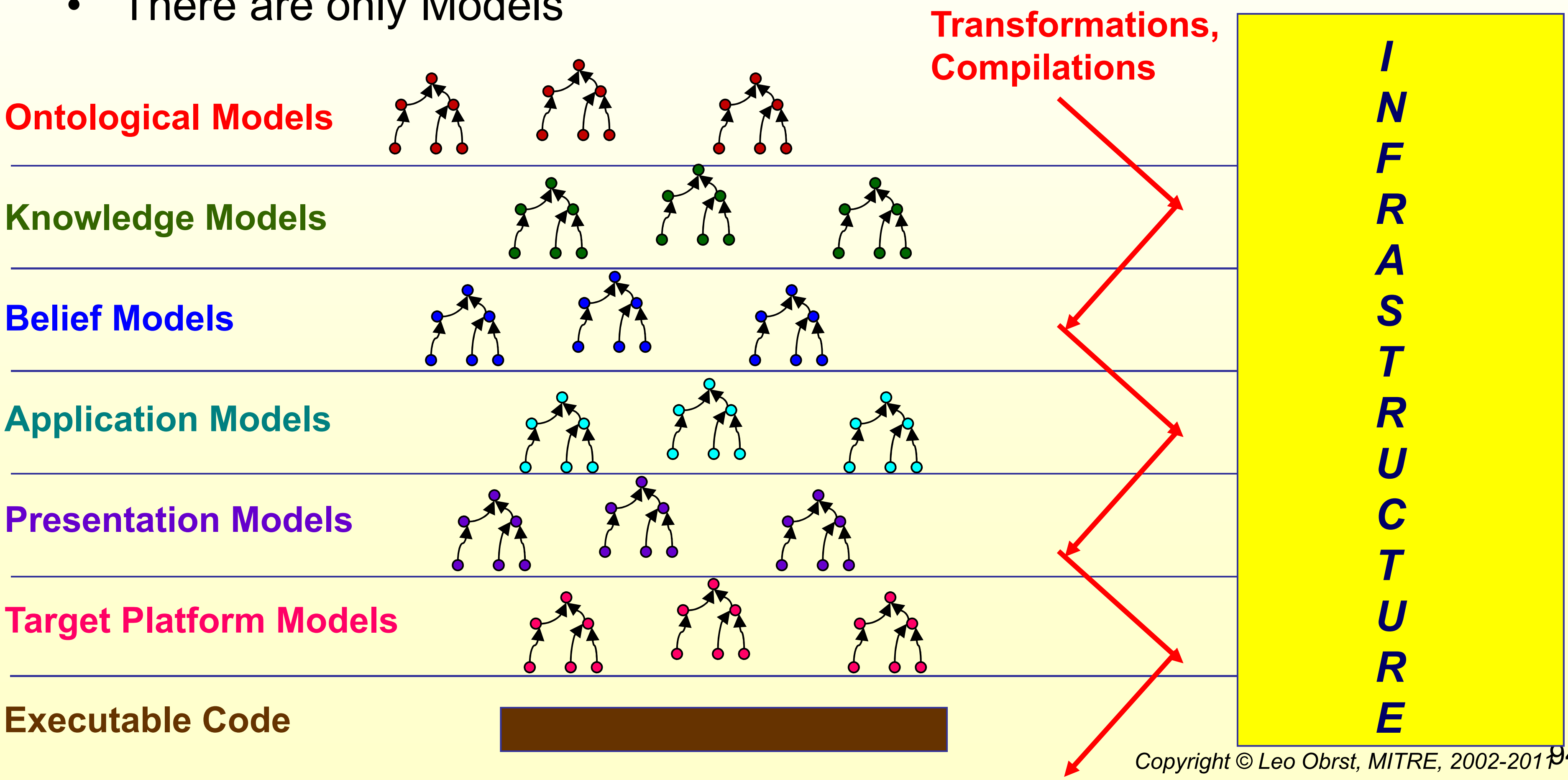
- The point is that we need to model our best human theories (naïve or scientific, depending on our system needs)
- In a declarative fashion (so that humans can easily verify them)
- And get our machines to work off them, as **models** of what humans do and **mean**
- We need to build our systems, our databases, our intelligent agents, and our documents on these **models of human meaning**
- These models must:
  - Represent once (if possible)
  - Be semantically reasonable (sound)
  - Be modular (theories or micro-theories or micro-micro-theories)
  - Be reused. Be composable. Be plug-and-playable
  - Be easily created and refined. Adaptable to new requirements, dynamically modifiable
  - Be consistent or boundably consistent so that our machines can reason and give use conclusions that are sound, trustable or provable, and secure
- We need to enable machines to come up to our human conceptual level (rather than forcing humans to go down to the machine level)

Conclusion

- We have discussed Syntax and Semantics, and what the distinctions are
- Ontology Spectrum and the Range of Semantic Models: from Taxonomy (both Weak and Strong) to Thesaurus to Conceptual Model (Weak Ontology) to Logical Theory (Strong Ontology)
- Logic: Propositional and Predicate Logic, Description Logics
- Ontologies: Levels, Architecture, Maturity Model, Complexity of Applications, Recall/Precision, Integration, Notional Cost/Benefit
- Semantic Web: RDF/S, OWL, SWRL, RIF

What do we want the future to be?

- 2100 A.D: models, models, models
- There are no human-programmed programming languages
- There are only Models



Thank You! Questions? lobrst@mitre.org

Lunch!