Knowledge Representation

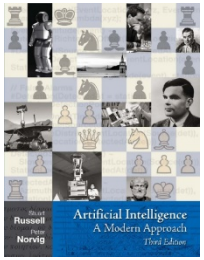
Prof. Dr. habil. Jana Koehler

Artificial Intelligence - Summer 2019

Deep thanks go to
Prof. Bernhard Nebel and
Prof. Franz Baader for
sharing their course material

Agenda

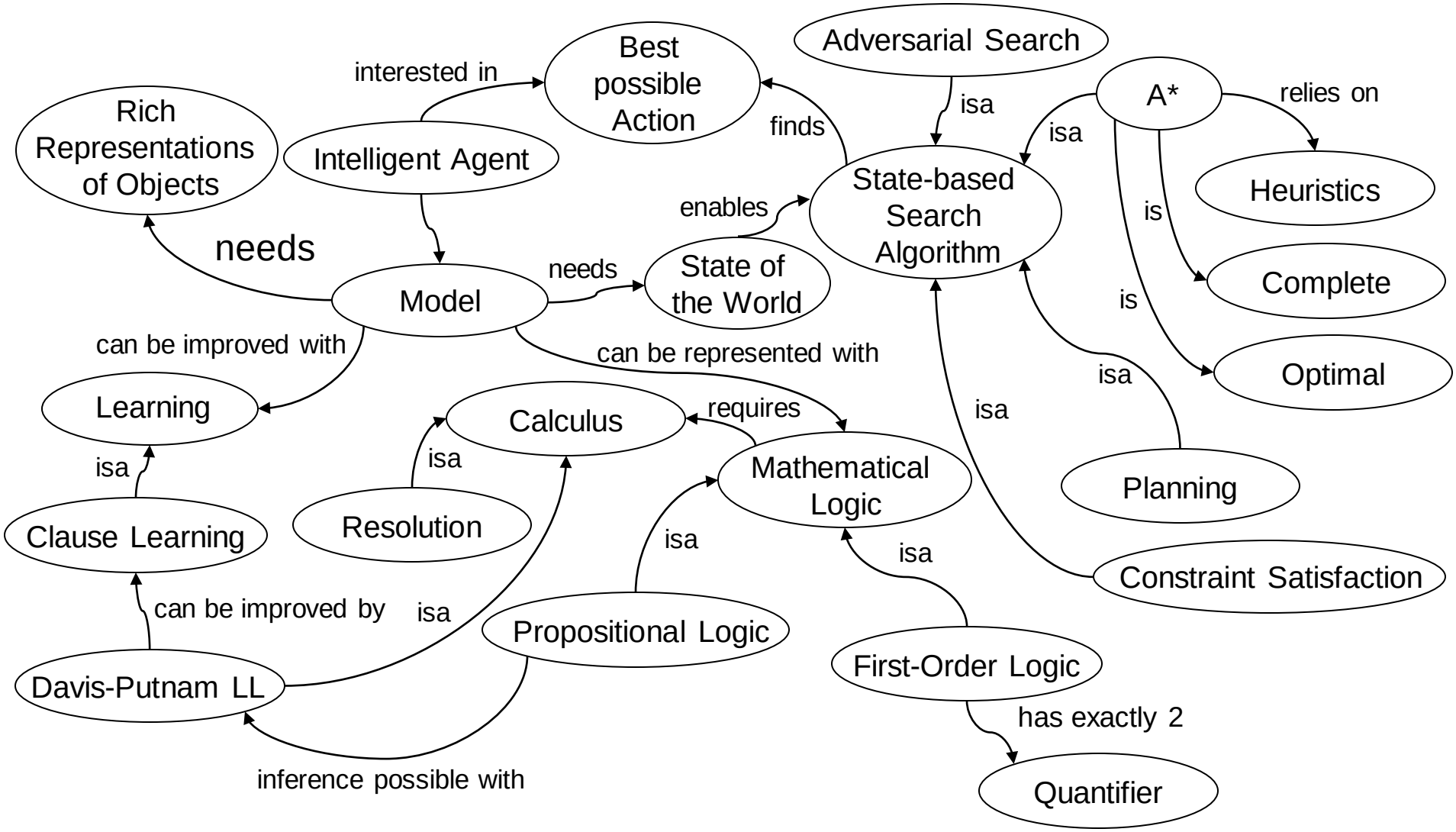
- Representation of conceptual knowledge
  - Frames, Semantic Nets, Description Logics
- The description logic $\mathcal{ALC}$
  - A-Box and T-Box representation
  - Reasoning procedures and complexity
- Nonmonotonic reasoning
  - Dealing with exceptions
  - Revising a knowledge base
- Web ontologies and the W3C OWL standard
  - Computing Subsumption in OWL
  - Querying the semantic web with Sparql



Chapter 12
(followed, but more formally, going deeper in selected areas)

Representation of Conceptual Knowledge

An Extract of My Conceptual Model of this Lecture



Remember: Symbolic Representations

A chair

- is a portable object
- has a horizontal surface at a suitable height for sitting
- has a vertical surface suitably positioned for leaning against

- Find a definition
  - using symbols, concepts, rules, some formalism
  - apply automated reasoning procedures

cisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We

Knowledge Representation

- Agents need *knowledge* before they can start to act intelligently
  - important objects in a domain, what properties these objects have, and how they relate to each other
    - Abstract concepts: “car”, “book”
    - Concrete instances of these concepts: Citroen C3 “ZH 2…”
    - Properties: “car has wheels = exactly 4”
    - Object-Object relations: “a car is a moving vehicle”
  - actions they can perform in a domain and how these affect the domain´s objects
    - We have seen PDDL and STRIPS as popular formalisms
  - about temporal relationships between events, spatio-temporal relations between objects, physical laws,…

Knowledge Representation and (!) Reasoning

- How can agents exploit the knowledge they have?

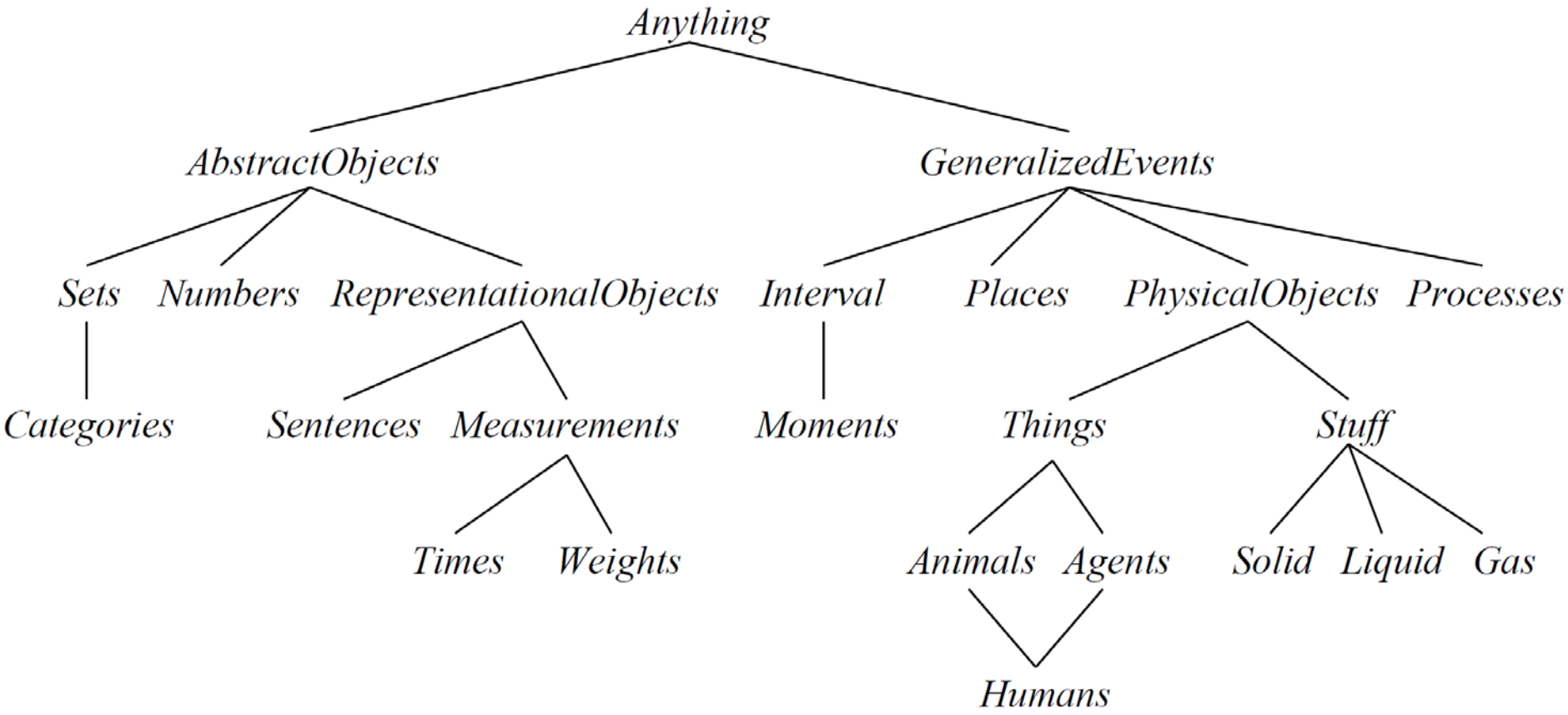
- They need some reasoning component to ask various questions about the objects and concepts in their knowledge base
  - Is “Citroen C3 Z…” a moving vehicle?
  - How many wheels does it have?
  - Which other cars does the agent know about?
  - Are there moving vehicles that are not cars?

➢ How can we formalize such a knowledge base and its calculus?

Categories and Objects

- We need to describe the objects in our world using categories
- Necessary to establish a common category system for different applications (in particular on the web)
- There are a number of quite general categories everybody and every application uses

The Upper Ontology: A General Category Hierarchy



Frames – Semantic Nets – Description Logics

- How to describe more specialized things?

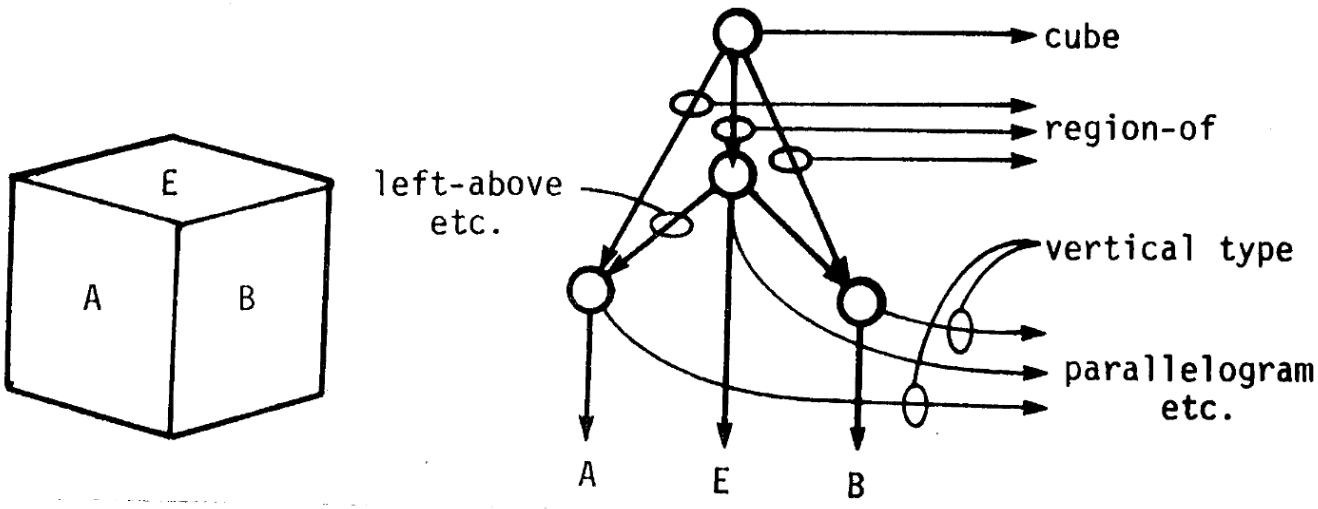
- Use definitions and/or necessary conditions referring to other already defined concepts:
  - A ***parent is a human with at least one child***.

- More complex description:
  - A proud-grandmother is a human, which is female with at least two children who are parents whose children are all computer science students.

Marvin Minsky: A Framework for Representing Knowledge

A frame is a data-structure for representing a stereotyped situation, like being in a certain kind of living room, or going to a child's birthday party. Attached to each frame are several kinds of information.

In the tradition of Guzman and Winston, we assume that the result of looking at a cube is a structure something like that in figure 1.1.



1974

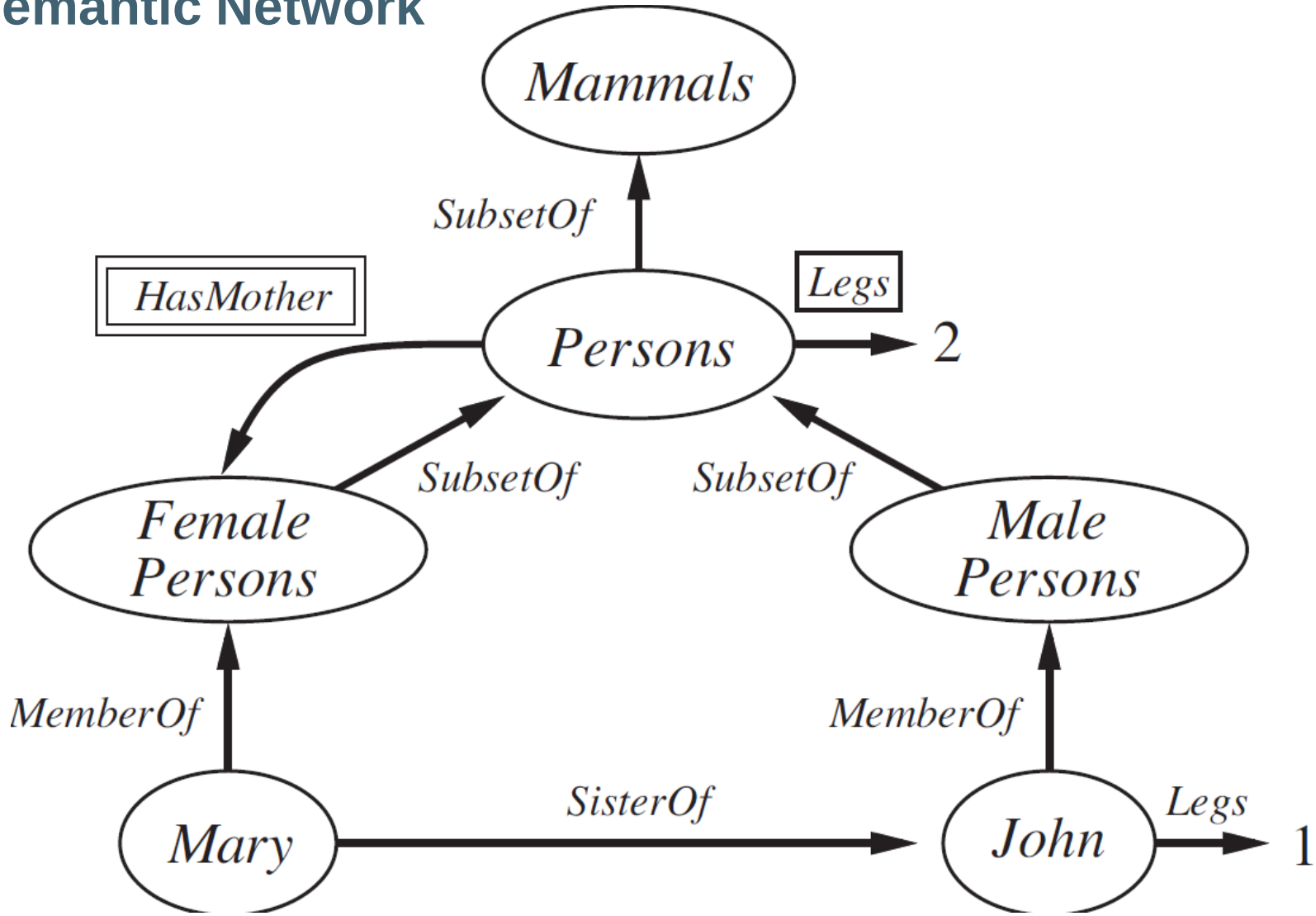
The substructures "A" and "B" represent details or decorations on two faces of the cube. When we move to the right, face "A" disappears from view, while the new face decorated with "C" is now seen. If we had to reanalyse the scene from the start, we would have to

(1) lose the knowledge about "A,"
(2) recompute "B," and
(3) compute the description of "C."

Semantic Networks

- In 1909, **Charles S. Peirce** proposed a graphical notation of nodes and edges named *existential graphs* that he called "the logic of the future“
- In 1956, **Richard H. Richens** proposes "Semantic Nets" as an "interlingua" for machine translation of natural languages
- In 1963, M. Ross Quillian presented a “notation for representing conceptual information”

A Semantic Network



Description Logics

- Many researchers contributed to the formalization of semantics networks as a fragment of first-order predicate logic (PL 1)
  - Semantics of DLs can be given using ordinary PL1
  - Alternatively, DLs can be considered as modal logics
    - Extensions of PL1 with operators expressing modalities
      - PL1: John is happy
      - ML: John is always happy, John is sometimes happy
- Reasoning for most DLs is much more efficient than for PL1
  - A family of DL languages was developed over the years of varying complexity (KL-ONE, CLASSIC, ALC, OWL)

The Notion of Description Logics

- Subfield of knowledge representation (KR), which is a subfield of AI
- Description Logic: name of a research field in AI/KR
- Description Logics: a family of knowledge representation languages
- Description Logic X: a member of this family

General Goals when Developing a Solution for KR

- formalism: well-defined syntax and formal, unambiguous semantics
- high-level description: only relevant aspects represented, others left out
- intelligent applications: must be able to reason about the knowledge, and infer implicit knowledge from the explicitly represented knowledge
- effectively used: need for practical reasoning tools and efficient implementations

Syntax

- Explicit symbolic representation of knowledge
  - Not implicit as for example in neural networks

$$\mathtt{Woman} \equiv \mathtt{Person} \sqcap \mathtt{Female}$$
$$\mathtt{Man} \equiv \mathtt{Person} \sqcap \neg\mathtt{Female}$$
$$\mathtt{Mother} \equiv \mathtt{Woman} \sqcap \exists\mathtt{hasChild}.\top$$
$$\mathtt{Person} \equiv \mathtt{Man} \sqcup \mathtt{Woman}$$
$$\bot \equiv \mathtt{Male} \sqcap \mathtt{Female}$$

$\mathtt{hasChild}(\mathtt{STEPHEN},\mathtt{MARC})$
$\mathtt{hasChild}(\mathtt{MARC},\mathtt{ANNA})$
$\mathtt{hasChild}(\mathtt{JOHN},\mathtt{MARIA})$
$\mathtt{hasChild}(\mathtt{ANNA},\mathtt{JASON})$

$\mathtt{Male}(\mathtt{JOHN})$
$\mathtt{Male}(\mathtt{MARC})$
$\mathtt{Male}(\mathtt{STEPHEN})$
$\mathtt{Male}(\mathtt{JASON})$
$\mathtt{Female}(\mathtt{MICHELLE})$
$\mathtt{Female}(\mathtt{ANNA})$
$\mathtt{Female}(\mathtt{MARIA})$

(Declarative) Semantics

- Mapping of symbolic expressions to an interpretation
- Notion of truth, which allows us to determine whether a symbolic expression is true in the world under consideration (has a model)
- Syntax & semantics determine the expressive power of a KR language
  - Not too low: can we represent all knowledge of interest?
  - Not too high: are the representational means adequate?

Reasoning

- Deduce implicit knowledge from the explicitly represented knowledge
  - Results should only depend on the semantics of the representation language, not on the syntactic representation
  - Semantically equivalent knowledge should lead to the same result

$$\forall x.\forall y.\,(male(y) \wedge \exists z.(has\_child(x,z) \wedge has\_child(z,y)) \rightarrow has\_grandson(x,y)$$

$$has\_child(John, Mary)$$

$$has\_child(Mary, Paul)$$

$$male(Paul)$$

$$grandson\_of(John, Paul)$$

implicit knowledge

Reasoning Procedure (Calculus)

- Ideally, we want a decision procedure for the problem:
  - **Soundness**: positive answers are correct
  - **Completeness**: negative answers are correct
  - **Termination**: always gives an answer in finite time

- As **efficient** as possible, preferable optimal wrt. the complexity of the problem and **practical** (easy to implement)

Challenge: Balancing Expressivity of Formalism and Efficiency of Reasoning Procedure

- Satisfiability in first-order logic does not have a decision procedure.
  - full first-order logic is thus not an appropriate knowledge representation formalism

- Satisfiability in propositional logic has a decision procedure, but the problem is NP-complete.
  - there are, however, highly optimized SAT solvers that behave well in practice
  - expressive power is, however, often not sufficient to express the relevant knowledge

The Description Logic $\mathcal{ALC}$

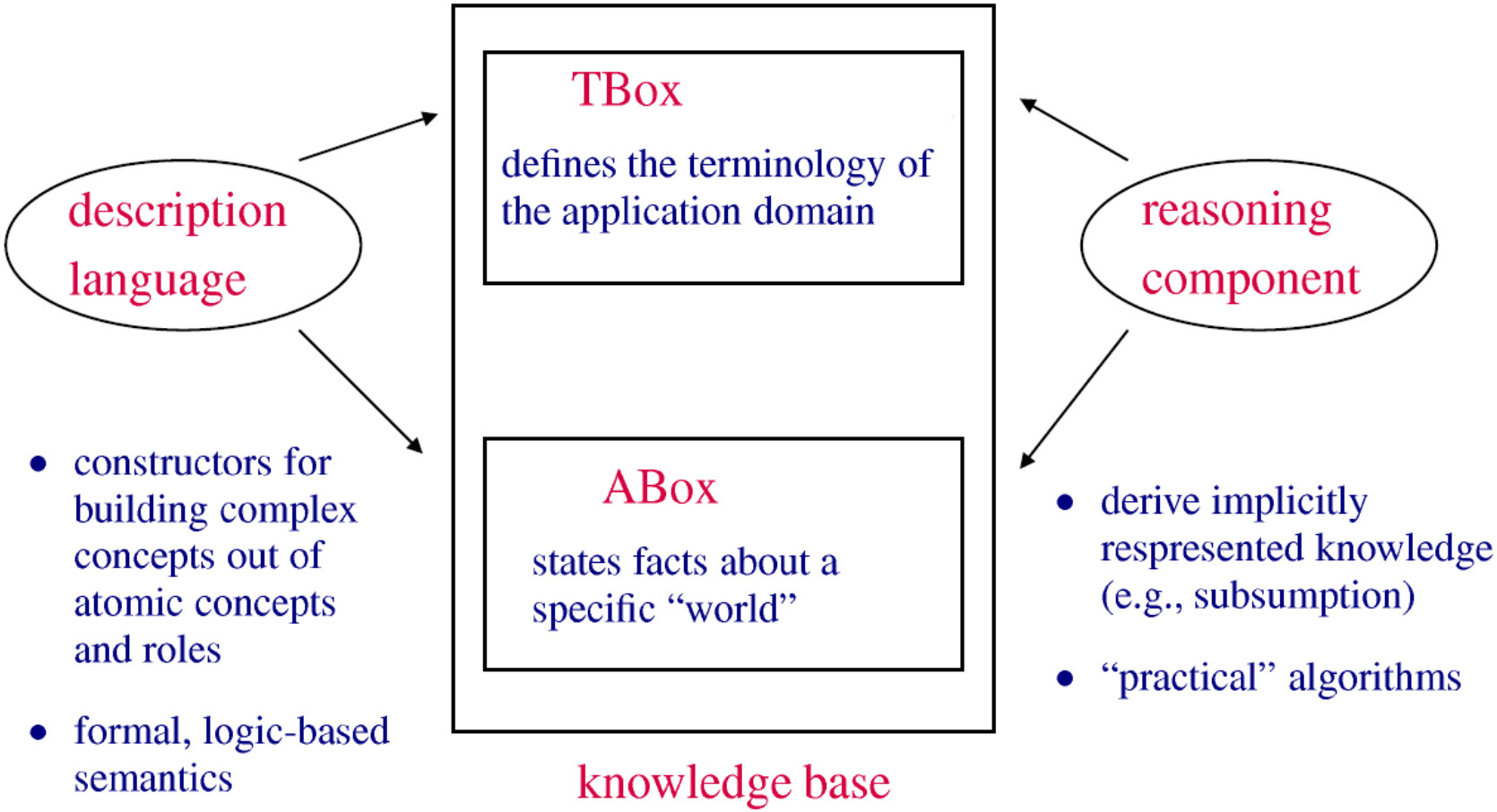
The Description Logic $\mathcal{ALC}$

$\mathcal{ALC}$ attributive language with complement
[Schmidt-Schauß&Smolka, 1991]

Naming scheme:

- basic language $\mathcal{AL}$
- extended with contructors whose "letter" is added after the $\mathcal{AL}$
- $\mathcal{C}$ stands for complement, i.e., $\mathcal{ALC}$ is obatined from $\mathcal{AL}$ by adding the complement ($\neg$) operator

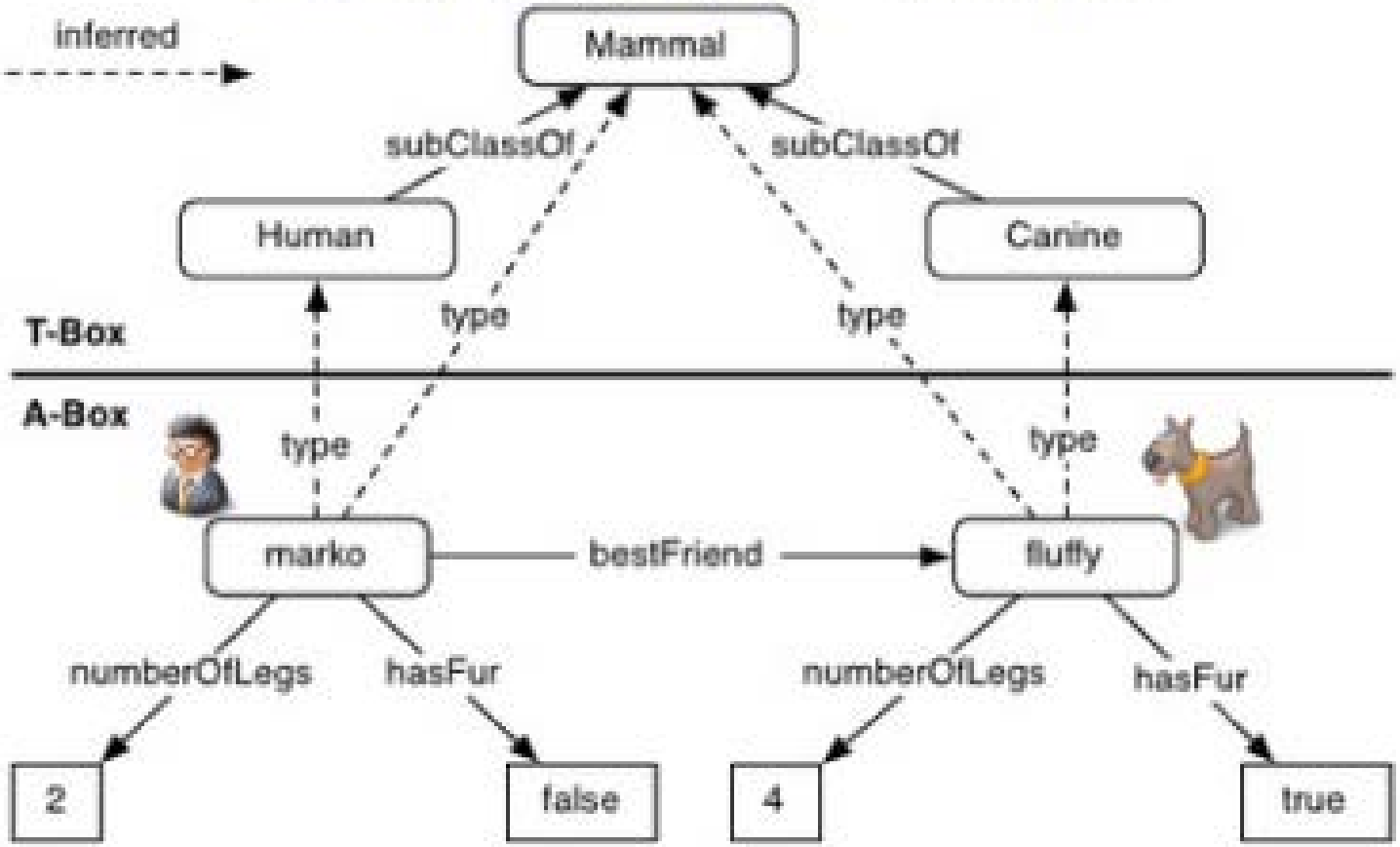
A Description Logic System



- constructors for building complex concepts out of atomic concepts and roles
- formal, logic-based semantics

- derive implicitly respresented knowledge (e.g., subsumption)
- "practical" algorithms

Description Logics - Example



* The T-Box includes other description information, but for diagram clarity, this was left out.

Syntax of $\mathcal{ALC}$

Let **C** and **R** be disjoint sets of concept names and role names, respectively.

$\mathcal{ALC}$-concept descriptions are defined by induction:

- If $A \in \mathbf{C}$, then A is an $\mathcal{ALC}$-concept description.
- If C, D are $\mathcal{ALC}$-concept descriptions, and $r \in \mathbf{R}$, then the following are $\mathcal{ALC}$-concept descriptions:
  - $C \sqcap D$ (conjunction)
  - $C \sqcup D$ (disjunction)
  - $\neg C$ (negation)
  - $\forall r.C$ (value restriction)
  - $\exists r.C$ (existential restriction)

Abbreviations:

- $\top := A \sqcup \neg A$ (top)
- $\bot := A \sqcap \neg A$ (bottom)
- $C \Rightarrow D := \neg C \sqcup D$ (implication)

$\mathcal{ALC}$ Examples

$$\text{Person} \sqcap \text{Female}$$

$$\text{Participant} \sqcap \exists\text{attends}.\text{Talk}$$

$$\text{Participant} \sqcap \forall\text{attends}.(\text{Talk} \sqcap \neg\text{Boring})$$

$$\text{Speaker} \sqcap \exists\text{gives}.(\text{Talk} \sqcap \forall\text{topic}.\text{DL})$$

$$\text{Speaker} \sqcap \forall\text{gives}.(\text{Talk} \sqcap \exists\text{topic}.(\text{DL} \sqcup \text{FuzzyLogic}))$$

Notation

- concept names are called atomic
- all other descriptions are called complex
- instead of ALC-concept description we often say ALC-concept or concept description or concept
- *A,B* often used for concept names
- *C,D* for complex concept descriptions
- *r,s* for role names

Semantics of $\mathcal{ALC}$

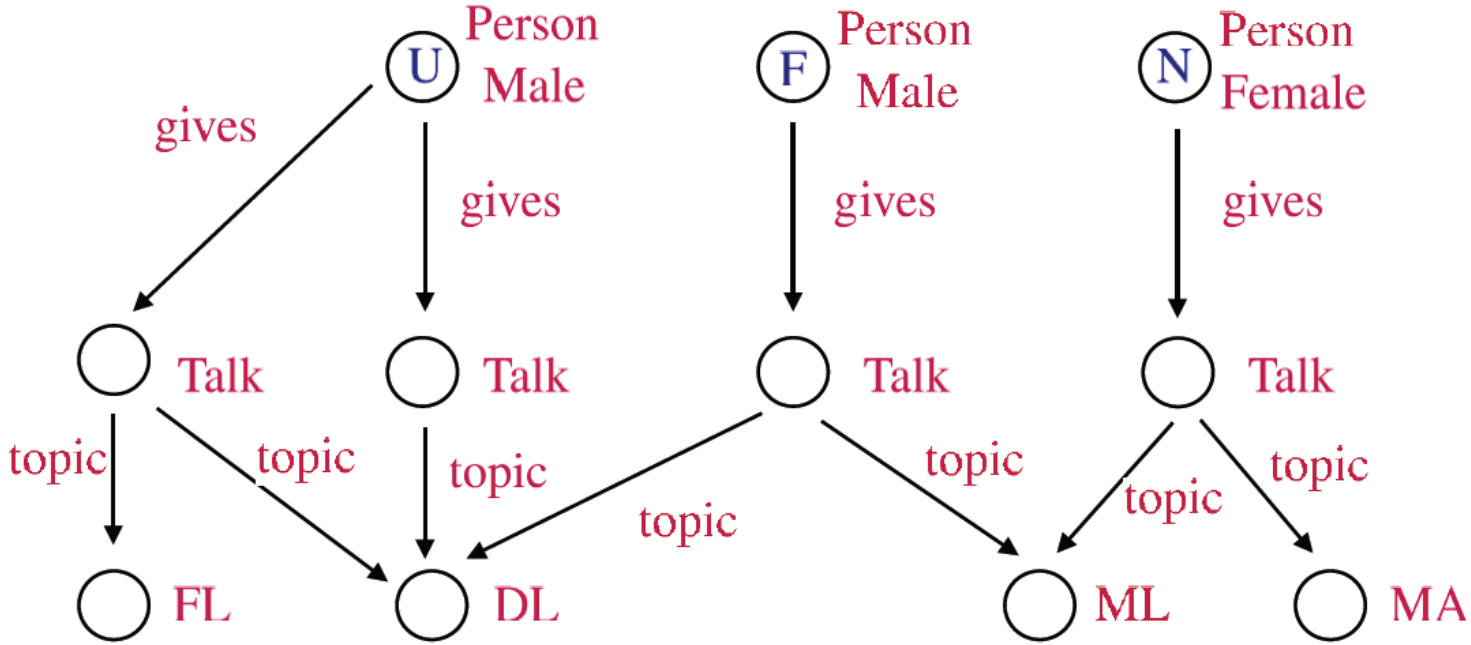
An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an extension mapping $\cdot^{\mathcal{I}}$:

- $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for all $A \in \mathbf{C}$, concepts interpreted as sets
- $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for all $r \in \mathbf{R}$. roles interpreted as binary relations

The extension mapping is extended to complex $\mathcal{ALC}$-concept descriptions as follows:

- $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $(C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $(\forall r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \text{for all } e \in \Delta^{\mathcal{I}} : (d, e) \in r^{\mathcal{I}} \text{ implies } e \in C^{\mathcal{I}}\}$
- $(\exists r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \text{there is } e \in \Delta^{\mathcal{I}} : (d, e) \in r^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}}\}$

Example of an Interpretation



Relationship with First-Order Predicate Logic

- Concept names are unary predicates, and role names are binary predicates.
- Interpretations for $\mathcal{ALC}$ can then obviously be viewed as first-order interpretations for this signature.
- Concept descriptions correspond to first-order formulae with one free variable.
- Given such a formula $\phi(x)$ with the free variable x and an interpretation $\mathcal{I}$, the extension of ϕ w.r.t. $\mathcal{I}$ is given by

$$\phi^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \mathcal{I} \models \phi(d)\}$$

- Goal: translate $\mathcal{ALC}$-concepts C into first-order formulae $\tau_x(C)$ such that their extensions coincide.

The T-Box

- A general concept inclusion is of the form $C \sqsubseteq D$ where C, D are concept descriptions.
- A TBox is a finite set of GCIs.
- The interpretation $\mathcal{I}$ satisfies the GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- The interpretation $\mathcal{I}$ is a model of the TBox $\mathcal{T}$ iff it satisfies all the GCIs in $\mathcal{T}$.

➢ 2 T-Boxes are equivalent if they have the same models

Acyclic T-Box

An acyclic TBox is a finite set of concept definitions that

- does not contain multiple definitions;
- does not contain cyclic definitions.

multiple definition

$$\cancel{\begin{array}{lcl} A & \equiv & C \\ A & \equiv & D \end{array}} \quad \text{for } C \neq D$$

$$\cancel{\begin{array}{lcl} A & \equiv & B \sqcap \forall r.P \\ B & \equiv & P \sqcap \forall r.C \\ C & \equiv & \exists r.A \end{array}}$$

cyclic definition

No cyclic definitions:

there is no sequence $A_1 \equiv C_1, \ldots A_n \equiv C_n \in \mathcal{T}$ $(n \geq 1)$ such that

- A_{i+1} occurs in C_i $(1 \leq i < n)$
- A_1 occurs in C_n

Example of an Acyclic T-Box

$$\text{Woman} \equiv \text{Person} \sqcap \text{Female}$$

$$\text{Man} \equiv \text{Person} \sqcap \neg\text{Female}$$

$$\text{Talk} \equiv \exists\text{topic}.\top$$

$$\text{Speaker} \equiv \text{Person} \sqcap \exists\text{gives}.\text{Talk}$$

$$\text{Participant} \equiv \text{Person} \sqcap \exists\text{attends}.\text{Talk}$$

$$\text{BusySpeaker} \equiv \text{Speaker} \sqcap (\geq 3\ \text{gives}.\text{Talk})$$

$$\text{BadSpeaker} \equiv \text{Speaker} \sqcap \forall\text{gives}.(\forall\text{attends}^{-1}.(\text{Bored} \sqcup \text{Sleeping}))$$

Inverse roles: if r is a role, then r^- denotes its inverse

$$(r^-)^{\mathcal{I}} := \{(e,d) \mid (d,e) \in r^{\mathcal{I}}\}$$

The A-Box

An assertion is of the form

$$a : C \text{ (concept assertion)} \quad \text{or} \quad (a, b) : r \text{ (role assertion)}$$

where C is a concept description, r is a role, and a, b are individual names from a set $\mathbf{I}$ of such names (disjoint with $\mathbf{C}$ and $\mathbf{R}$).

An ABox is a finite set of assertions.

An interpretation $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if it satisfies all its assertions:

$$a^{\mathcal{I}} \in C^{\mathcal{I}} \quad \text{for all } a : C \in \mathcal{A}$$
$$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}} \quad \text{for all } (a, b) : r \in \mathcal{A}$$

$\mathcal{I}$ assigns elements $a^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ to individual names $a \in \mathbf{I}$

Example of an A-Box

$$\text{FRANZ} : \text{Lecturer}, \quad (\text{FRANZ}, \text{TU03}) : \text{teaches},$$
$$\text{TU03} : \text{Tutorial}, \quad (\text{TU03}, \text{RinDL}) : \text{topic},$$
$$\text{RinDL} : \text{DL}$$

Knowledge Bases

A knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

The interpretation $\mathcal{I}$ is a model of the knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ iff it is a model of $\mathcal{T}$ and a model of $\mathcal{A}$.

Reasoning Services in Description Logics

- **Subsumption**
  - Determine whether one description is more general than (subsumes) the other
- **Classification**
  - Create a subsumption hierarchy
- **Satisfiability**
  - Is a description satisfiable?
- **Instance relationship**
  - Is a given object an instance of a concept description?
- **Instance retrieval**
  - Retrieve all objects for a given concept description

Formalization of Reasoning Services

Let $\mathcal{T}$ be a TBox.

Satisfiability:

C is satisfiable w.r.t. $\mathcal{T}$ iff $C^{\mathcal{I}} \neq \emptyset$ for some model $\mathcal{I}$ of $\mathcal{T}$.

Subsumption:

C is subsumed by D w.r.t. $\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$) iff

$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of the TBox $\mathcal{T}$.

Equivalence:

C is equivalent to D w.r.t. $\mathcal{T}$ ($C \equiv_{\mathcal{T}} D$) iff

$C^{\mathcal{I}} = D^{\mathcal{I}}$ for all models $\mathcal{I}$ of the TBox $\mathcal{T}$.

Examples

- $A \sqcap \neg A$ and $\forall r.A \sqcap \exists r.\neg A$ are not satisfiable (unsatisfiable)
- $A \sqcap \neg A$ and $\forall r.A \sqcap \exists r.\neg A$ are equivalent
- $A \sqcap B$ is subsumed by A and by B.
- $\exists r.(A \sqcap B)$ is subsumed by $\exists r.A$ and by $\exists r.B$
- $\forall r.(A \sqcap B)$ is equivalent to $\forall r.A \sqcap \forall r.B$
- $\exists r.A \sqcap \forall r.B$ is subsumed by $\exists r.(A \sqcap B)$

Formalization of Assertional Reasoning

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base.

Consistency:

$\mathcal{K}$ is consistent iff there exists a model of $\mathcal{K}$.

Instance:

a is an instance of C w.r.t. $\mathcal{K}$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{K}$.

Realization

- Computing the most specific concept names in the TBox to which an ABox individual belongs.

Woman	$\equiv$ Person $\sqcap$ Female	
Man	$\equiv$ Person $\sqcap \neg$Female	**Person(Anna),…**
Mother	$\equiv$ Woman $\sqcap \exists$hasChild.$\top$	
Person	$\equiv$ Man $\sqcup$ Woman	Male(JOHN)
$\bot$	$\equiv$ Male $\sqcap$ Female	Male(MARC)
		Male(STEPHEN)
hasChild(STEPHEN,MARC)		Male(JASON)
hasChild(MARC,ANNA)		Female(MICHELLE)
hasChild(JOHN,MARIA)		Female(ANNA)
hasChild(ANNA,JASON)		Female(MARIA)

- Anna is a Person, a Woman, and a Mother
  - Mother is the most specific concept

Further Equivalences

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base, C, D concept descriptions, and $a \in \mathbf{I}$.

1. $C \equiv_{\mathcal{T}} D$ iff $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$
2. $C \sqsubseteq_{\mathcal{T}} D$ iff $C \equiv_{\mathcal{T}} C \sqcap D$
3. $C \sqsubseteq_{\mathcal{T}} D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{T}$
4. C is satisfiable w.r.t. $\mathcal{T}$ iff $C \not\sqsubseteq_{\mathcal{T}} \bot$
5. C is satisfiable w.r.t. $\mathcal{T}$ iff $(\mathcal{T}, \{a : C\})$ is consistent
6. a is an instance of C w.r.t. $\mathcal{K}$ iff $(\mathcal{T}, \mathcal{A} \cup \{a : \neg C\})$ is inconsistent
7. $\mathcal{K}$ is consistent iff a is not an instance of $\bot$ w.r.t. $\mathcal{K}$

Complexity of Reasoning in $\mathcal{ALC}$

Complexity classes:

$$\text{PTime} \subseteq \text{NP} \subseteq \text{PSpace} \subseteq \text{ExpTime} \subseteq \text{NExpTime}$$

In $\mathcal{ALC}$, satisfiability of a concept description w.r.t. a TBox is decidable.

In $\mathcal{ALC}$, concept satisfiability and subsumption w.r.t. acyclic TBoxes are in PSpace.

In $\mathcal{ALC}$, concept satisfiability and subsumption w.r.t. general TBoxes are in ExpTime.

Expressivity and Undecidability

Consider the following part of a TBox about universities:

$$\begin{aligned} \mathsf{Course} &\sqsubseteq \exists \mathsf{held\text{-}at}.\mathsf{University} \\ \mathsf{Lecturer} &\sqsubseteq \exists \mathsf{teaches}.\mathsf{Course} \sqcap \exists \mathsf{employed\text{-}by}.\mathsf{University} \end{aligned}$$

To express that someone who teaches a course held at a university must be employed by that specific university, we need role value maps:

$$\top \sqsubseteq (\mathsf{teaches} \circ \mathsf{held\text{-}at} \sqsubseteq \mathsf{employed\text{-}by}).$$

Though very useful, role value maps are not available in modern DL systems since they cause undecidability:

In the extension of $\mathcal{ALC}$ with role value maps, concept satisfiability and subsumption (without TBoxes) are undecidable.

Nonmonotonic Reasoning

Limitations of Standard Logic

- Standard logic is **monotonic**:
  - **once** you prove something is true, it is **true forever**
- Monotonic Logic is **not a good fit to reality**
  - If the wallet is in the purse, and the purse is in the car, we can conclude that the wallet is in the car
  - But what if we take the purse out of the car?
  - Where is the wallet?

- Revising knowledge bases in the light of new information
- Dealing with exceptions

The Frame Problem in AI

- Specification of the properties that do *not change* as a result of an action
  - Impossible to enumerate explicitly

- A more elegant way to solve the frame problem is to fully describe the successor situation: (inertia=things do not change unless otherwise specified)

true after action
$\Leftrightarrow$ [action made it true or, already true and the action did not *falsify* it]

- **Closed world Assumption**: only the agent changes the situation (anything that is not mentioned as being changed, remains unchanged)

A Brief History of Nonmonotonic Logic

- John McCarthy developed circumscription in 1977/80 to deal with the frame problem in AI
- Yoav Shoham generalized circumscription to preferential entailment in 1987
- Drew McDermott and Jon Doyle developed nonmonotonic logics based on "consistency with current beliefs“ in 1980
- Ray Reiter developed default logic in 1978/80
- Robert Moore developed autoepistemic logic 1985
- Ilkka Niemelä and others developed *Answer Set Programming* (ASP) in 1999

Belief Revision

- Process of changing beliefs to take into account a new piece of information

- To understand what **nonmonotonic reasoning** means, consider a standard example:

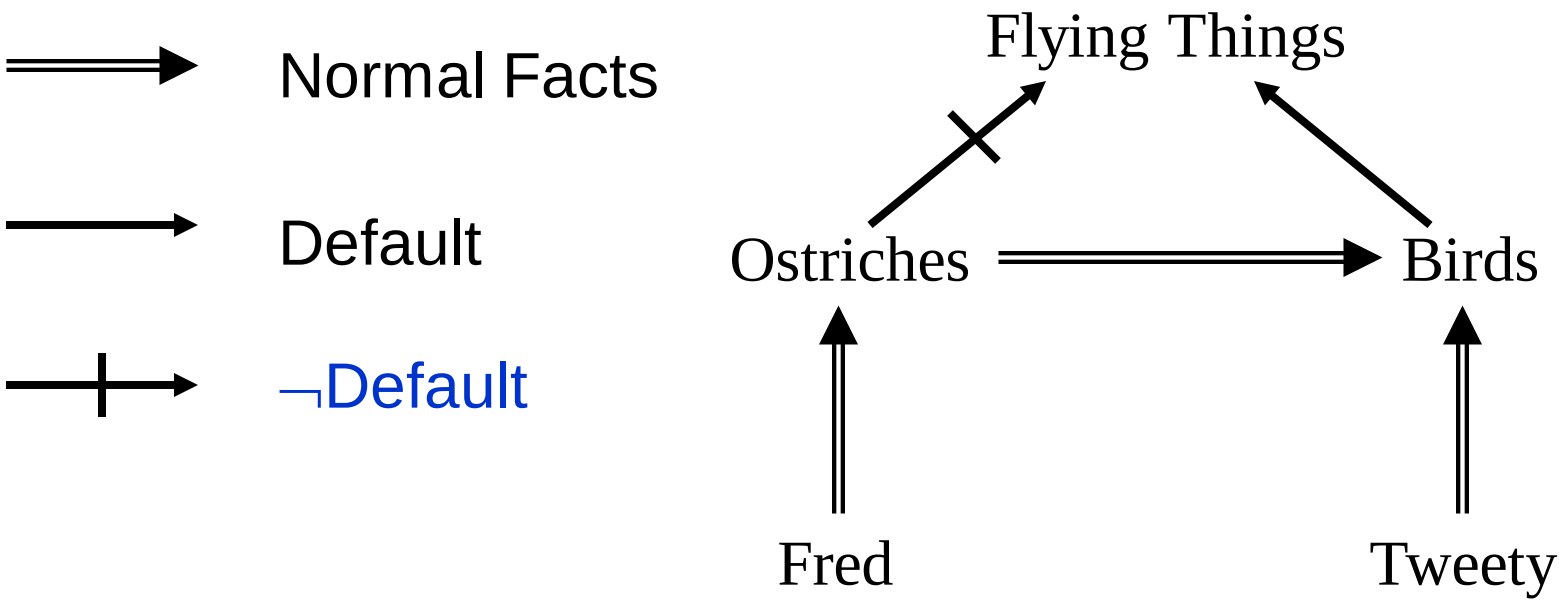
  “All birds fly."

  "Tweety is a bird."

  "Does Tweet fly?“

- The obvious answer is yes,
  - however what if later you learned that Tweety had a broken wing, then the answer becomes no,
  - what if you learned that tweet was an airplane pilot ...

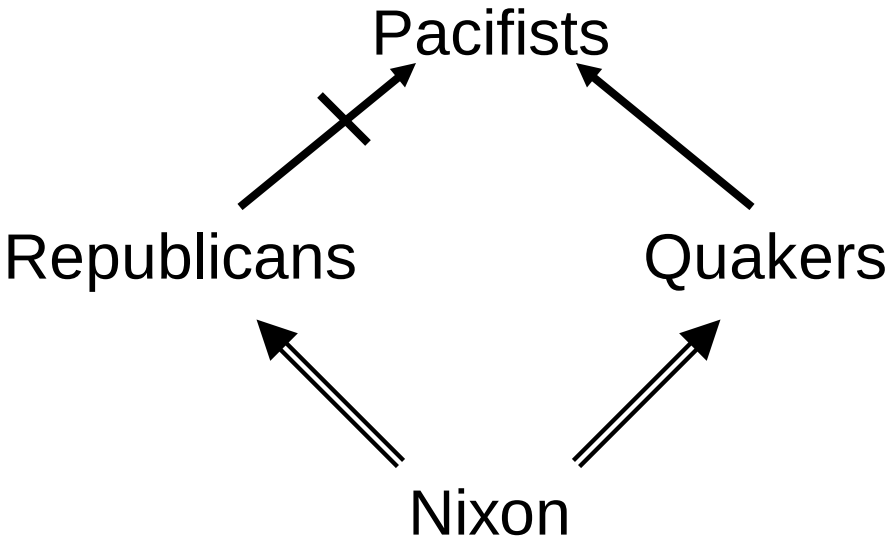
Inheritance Diagrams



The Nixon Diamond

- Usually, Quakers are pacifists
- Usually, Republicans are not pacifists
- Richard Nixon is both a Quaker and a Republican
- Is Nixon a Pacifist?

➢ default assumptions lead to mutually inconsistent conclusions



How many legs does Pat have?

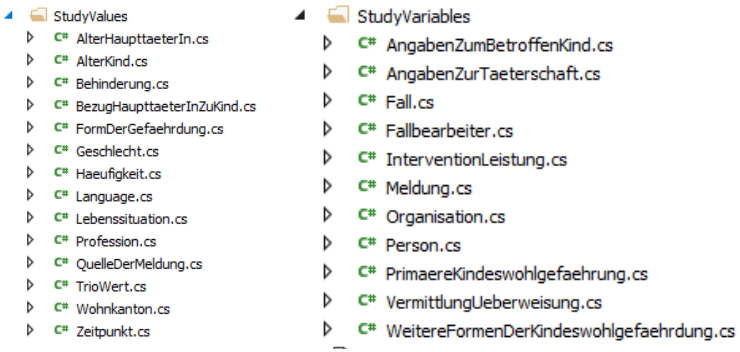
- Pat is a **Bat.**
- Bats are **Mammals**.
- Bats can fly.
- Bats have 2 legs.
- Mammals cannot fly.
- Mammals have 4 legs.

Web ontologies and the W3C OWL standard

Description Logic and Ontologies

- The W3C standards for OWL "Ontology Web Language" is based on Description Logic
- DBpedia is a famous ontology based on OWL
- Cyc https://www.cyc.com/ is another famous ontology based on description logics [Lenat & Guha, 1990]

The World of Data

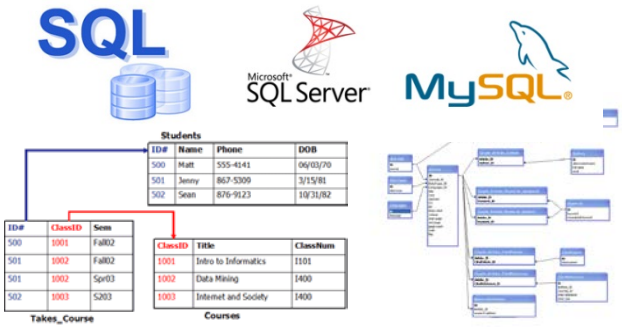


Object-Oriented Programming

Objects, Properties, Methods

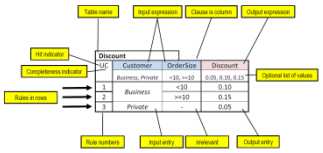
Semantic Web

concepts, relationships, ontologies



Relational Databases

Entities, Relations, Tables



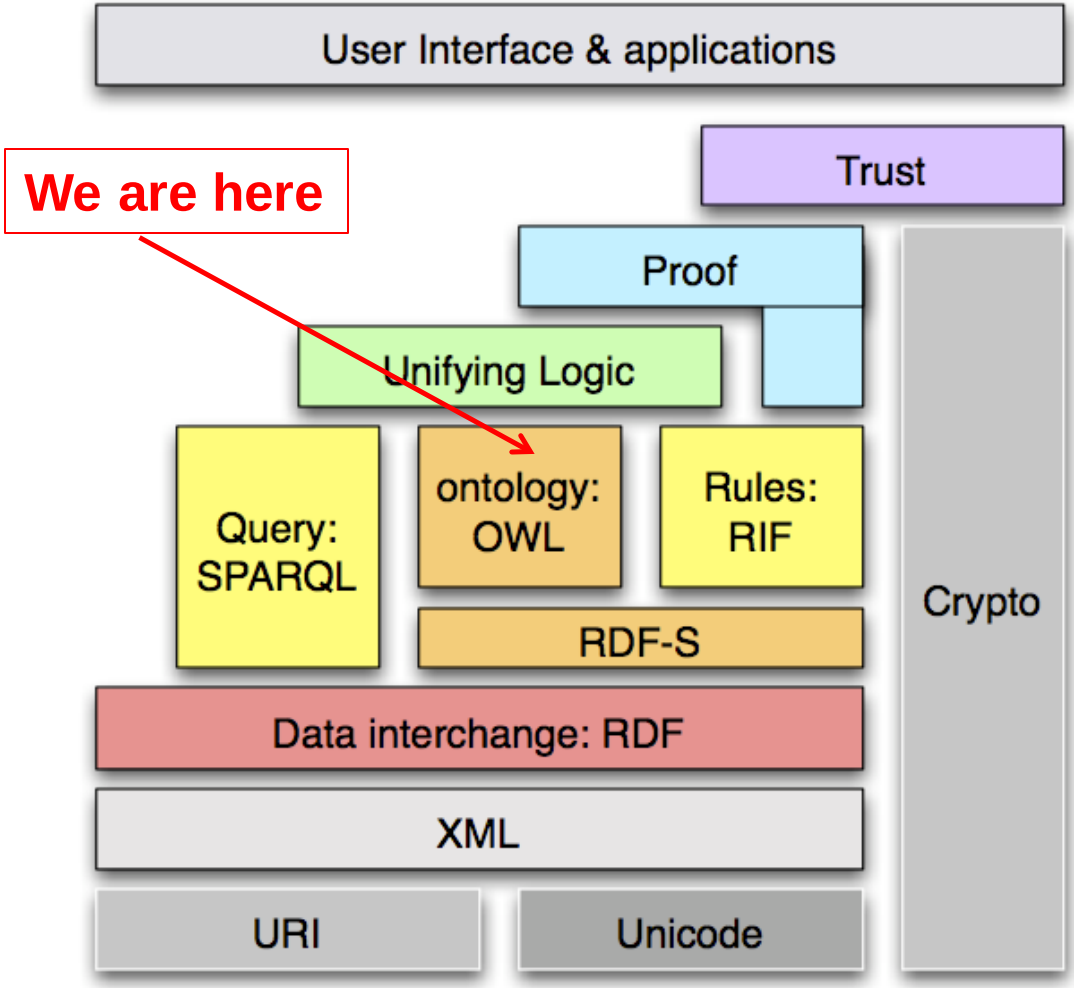
Deductive Databases

Business Rules …

(Distributed) NoSQL Databases

Attribute-Value-Pairs, Columns/Graphs

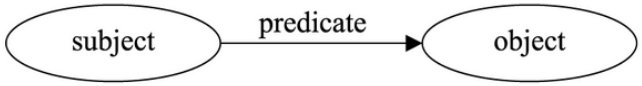
Semantic Web Architecture



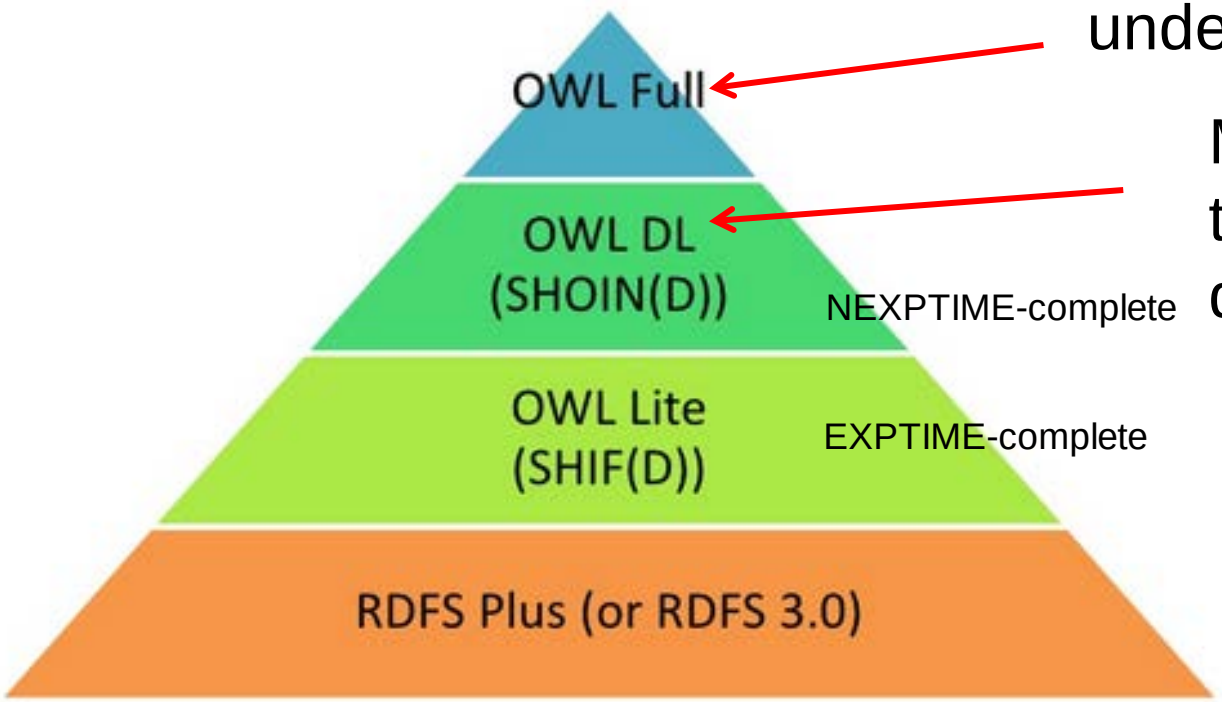
W3C, 2006
http://www.ansta.co.uk/blog/semantic-web-technologies-part-3-94/

- Ontology editor
  https://protege.stanford.edu

RDF Triples



The OWL Family



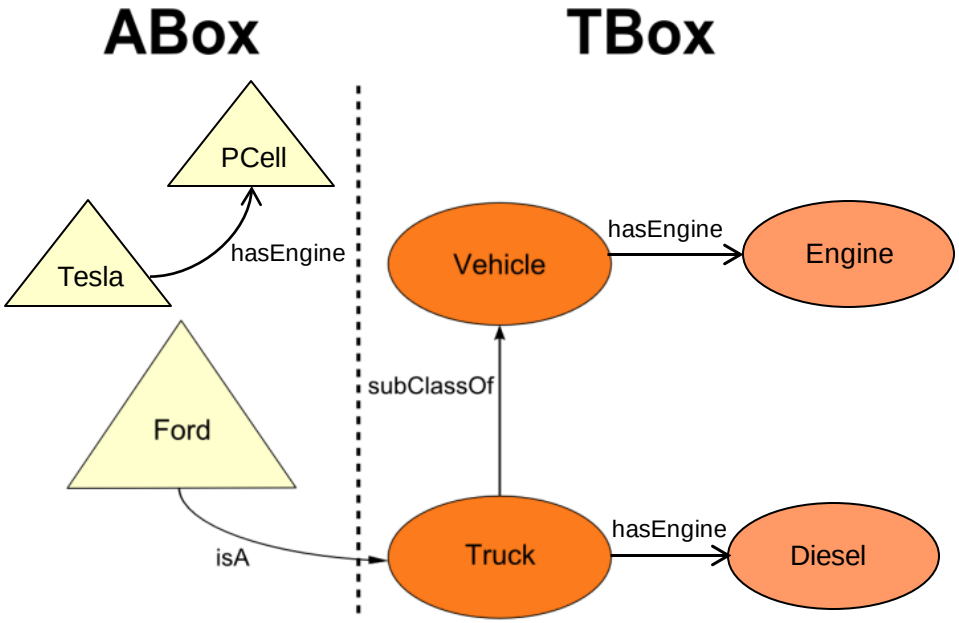
undecidable

More expressive than *ALC*, but still decidable

OWL DL is decidable

- the reasoner underlying any application will eventually answer our question!

OWL DL



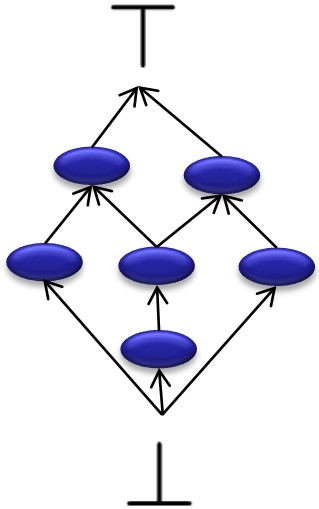
Abstract Syntax	DL Syntax
Descriptions (C)	
A	A
owl:Thing	$\top$
owl:Nothing	$\bot$
intersectionOf($C_1 \ldots C_n$)	$C_1 \sqcap \ldots \sqcap C_n$
unionOf($C_1 \ldots C_n$)	$C_1 \sqcup \ldots \sqcup C_n$
complementOf(C)	$\neg C$
oneOf($o_1 \ldots o_n$)	$\{o_1\} \sqcup \ldots \sqcup \{o_n\}$
restriction(R someValuesFrom(C))	$\exists R.C$
restriction(R allValuesFrom(C))	$\forall R.C$
restriction(R hasValue(o))	$R : o$
restriction(R minCardinality(n))	$\geqslant n\, R$
restriction(R minCardinality(n))	$\leqslant n\, R$
restriction(U someValuesFrom(D))	$\exists U.D$
restriction(U allValuesFrom(D))	$\forall U.D$
restriction(U hasValue(v))	$U : v$
restriction(U minCardinality(n))	$\geqslant n\, U$
restriction(U maxCardinality(n))	$\leqslant n\, U$
Data Ranges (D)	
D	D
oneOf($v_1 \ldots v_n$)	$\{v_1\} \sqcup \ldots \sqcup \{v_n\}$
Object Properties (R)	
R	R
inv(R)	R^-
Datatype Properties (U)	
U	U
Individuals (o)	
o	o
Data Values (v)	
v	v

Reasoning in OWL-DL: Class-Class Relationships

- **Class subsumption**

  Given classes C and D, determine if C is a subclass of D in the given ontology

  - build the class/subsumption hierarchy



- **Class satisfiability**

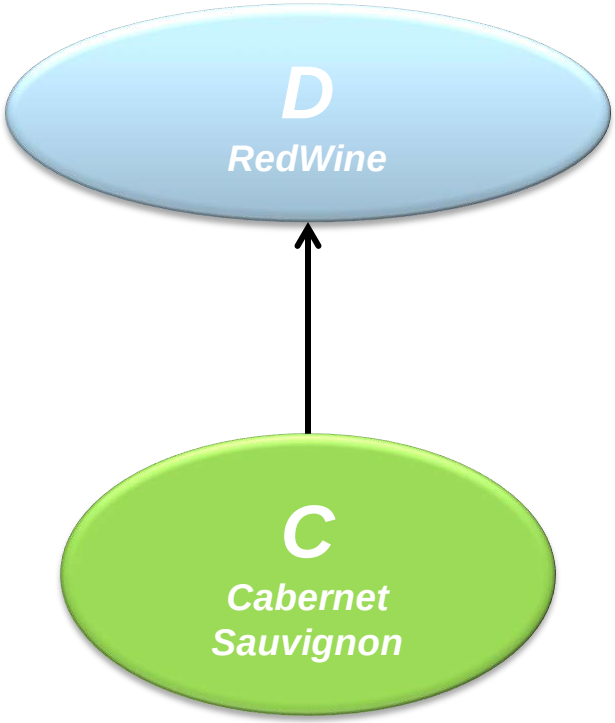
  Given a class C, determine if C is satisfiable (consistent) in the given ontology

  – C is satisfiable iff $C \not\sqsubseteq \bot$

Subsumption as Essential Class-Class Relationship

A class ***C*** is subsumed by a class ***D*** if and only if every model (satisfiable interpretation) of ***C*** is also a model of ***D***

$$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$$



- ***D subsumes C***
- ***C is a (subclass of) D***
- ***D is more general than C***
- ***C logically implies D***

- CabernetSauvignon isa RedWine
- CabernetSauvignon is subsumed by RedWine
- RedWine subsumes CabernetSauvigon

Computing Subsumption By Structural Comparison

1. Put class descriptions into a **<u>normal form representation</u>** exploiting equivalences
   - similar to the compution of normal forms (CNF/DNF) in First-Order Logics

2. Recursively descend into the structural parts of the descriptions and compare them to each other
   - if each (conjunctive) part of C is subsumed by some part of D, then C is subsumed by D
   - often done with graph traversal algorithms

A Simple Example

$$C \doteq \neg(\neg A \vee \neg B)$$
$$D \doteq A$$

$$C \sqsubseteq D \equiv C \rightarrow D$$

$$\neg(A \vee B) \equiv \neg A \wedge \neg B$$
$$\neg(\neg A) \equiv A$$

$$A \wedge B \rightarrow A$$

Example of Structural Comparison Rules for OWL-DL

Concept A	Concept B	Condition of $A \sqsubseteq B$
$\exists R.C$	$\exists S.D$	Iff $R \preceq S$ and $C \sqsubseteq D$
$\forall R.C$	$\forall S.D$	Iff $S \preceq R$ and $C \sqsubseteq D$
$\geqslant nR.C$	$\geqslant mS.D$	Iff $R \preceq S$ and $C \sqsubseteq D$ and $n \geqslant m$
$\leqslant nR.C$	$\leqslant mS.D$	Iff $S \preceq R$ and $D \sqsubseteq C$ and $n \leqslant m$

An Example

$E \doteq$ [**AND** Company
[**ALL** :Manager B-SchoolGrad]
[**EXISTS** 1 :Exchange]]

$D \doteq$ [**AND** Company
[**ALL** :Manager [**AND** B-SchoolGrad [**EXISTS** 2 :TechnicalDegree]]]
[**FILLS** :Exchange nasdaq]]

$D \sqsubseteq E$ **?**

minCardinality vs. hasValue ?

if e_j is of the form [**EXISTS** n r], then the corresponding d_i must be of the form [**EXISTS** n' r], for some $n' \geq n$; in the case where $n = 1$, the matching d_i can be of the form [**FILLS** r c], for any constant c;

A More Detailed Example

WellRoundedCo $\doteq$
[**AND** Company [**ALL** :Manager [**AND** B-SchoolGrad
[**EXISTS** 1 :TechnicalDegree]]]]

HighTechCo $\doteq$
[**AND** Company [**FILLS** :Exchange nasdaq] [**ALL** :Manager Techie]]

Techie $\doteq$ [**EXISTS** 2 :TechnicalDegree]

These definitions amount to a WellRoundedCo being a company whose managers are business school graduates who each have at least one technical degree, a HighTechCo being a company listed on the NASDAQ whose managers are all Techies, and a Techie being someone with at least two technical degrees.

Does CoolTecCo subsume HighTechCo?

CoolTecCo ≐

[**AND** Company
[**ALL** :Manager [**AND** B-SchoolGrad [**EXISTS** 2 :TechnicalDegree]]]
[**FILLS** :Exchange nasdaq]]

↑ ?

WellRoundedCo ≐
[**AND** Company [**ALL** :Manager [**AND** B-SchoolGrad
[**EXISTS** 1 :TechnicalDegree]]]]

HighTechCo ≐
[**AND** Company [**FILLS** :Exchange nasdaq] [**ALL** :Manager Techie]]

Techie ≐ [**EXISTS** 2 :TechnicalDegree]

- **how about the <u>intersection</u> of WellRoundedCo and HighTechCo?**

(1) Expand Definitions

WellRoundedCo $\doteq$
[**AND** Company [**ALL** :Manager [**AND** B-SchoolGrad
[**EXISTS** 1 :TechnicalDegree]]]]

HighTechCo $\doteq$
[**AND** Company [**FILLS** :Exchange nasdaq] [**ALL** :Manager Techie]]

Techie $\doteq$ [**EXISTS** 2 :TechnicalDegree]

First, we would expand the definitions of WellRoundedCo and HighTechCo, and then Techie, yielding this:

[**AND** [**AND** Company
[**ALL** :Manager [**AND** B-SchoolGrad
[**EXISTS** 1 :TechnicalDegree]]]]
[**AND** Company
[**FILLS** :Exchange nasdaq]
[**ALL** :Manager [**EXISTS** 2 :TechnicalDegree]]]]

Flatten and Combine AND Operators

```
[AND [AND Company
          [ALL :Manager [AND B-SchoolGrad
                             [EXISTS 1 :TechnicalDegree]]]]
     [AND Company
          [FILLS :Exchange nasdaq]
          [ALL :Manager [EXISTS 2 :TechnicalDegree]]]]
```

Next, we flatten the **AND** operators at the top level and then combine the **ALL** operators over :Manager:

```
[AND Company
     [ALL :Manager [AND B-SchoolGrad
                        [EXISTS 1 :TechnicalDegree]
                        [EXISTS 2 :TechnicalDegree]]]
     Company
     [FILLS :Exchange nasdaq]]
```

Remove Redundancy and Combine Operators

```
[AND Company
     [ALL :Manager [AND B-SchoolGrad
                        [EXISTS 1 :TechnicalDegree]
                        [EXISTS 2 :TechnicalDegree]]]
     Company
     [FILLS :Exchange nasdaq]]
```

Finally, we remove the redundant Company concept and combine the **EXISTS** operators over :TechnicalDegree, yielding the following:

```
[AND Company
     [ALL :Manager [AND B-SchoolGrad [EXISTS 2 :TechnicalDegree]]]
     [FILLS :Exchange nasdaq]]
```

CoolTecCo ≐

```
[AND Company
     [ALL :Manager [AND B-SchoolGrad [EXISTS 2 :TechnicalDegree]]]
     [FILLS :Exchange nasdaq]]
```

Applying DL-Specific Structural Rules

[**AND** Company
  [**ALL** :Manager B-SchoolGrad]
  [**EXISTS** 1 :Exchange]]

?↑

[**AND** Company
  [**ALL** :Manager [**AND** B-SchoolGrad [**EXISTS** 2 :TechnicalDegree]]]
  [**FILLS** :Exchange nasdaq]]

e_j subsumes d_i

if e_j is of the form [**EXISTS** n r], then the corresponding d_i must be of the form [**EXISTS** n' r], for some $n' \geq n$; in the case where $n = 1$, the matching d_i can be of the form [**FILLS** r c], for any constant c;

Computing Subsumption by Logical Proof

- ***D subsumes C if and only if C logically implies D***

- For $C \sqsubseteq D$ we need to show that

$$K \models C \rightarrow D$$

$$K \models \neg C \vee D$$

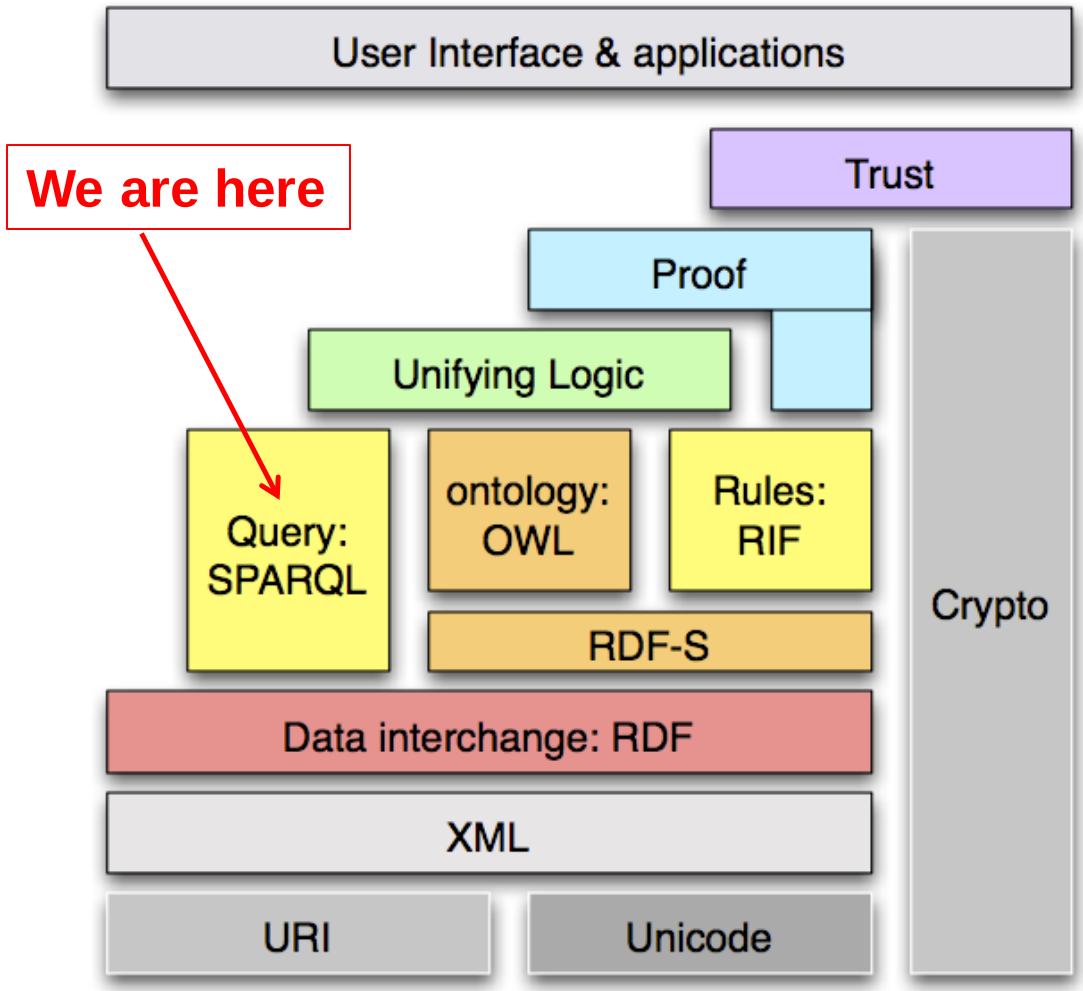
- using a Tableau theorem prover to construct a satisfying instance

$$K \not\models \neg(\neg C \vee D)$$

$$K \not\models C \wedge \neg D$$

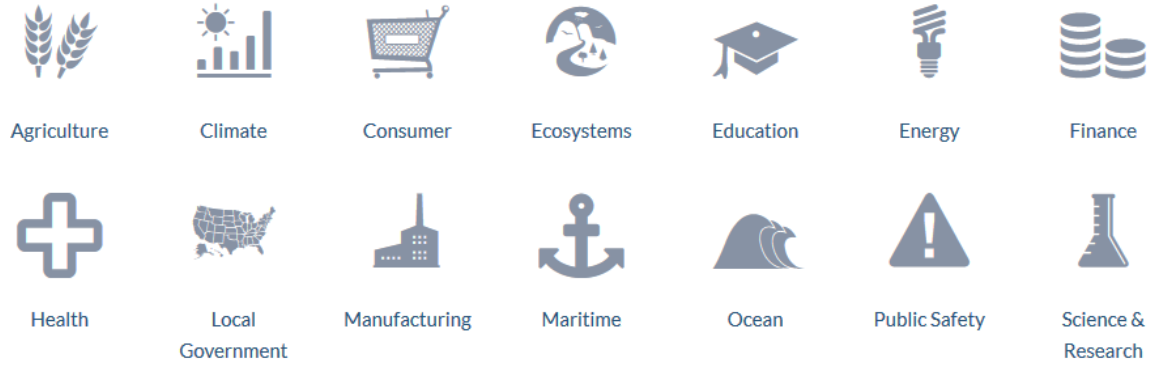
- using a SAT Checker to prove unsatisfiability

Querying the Semantic Web with SPARQL

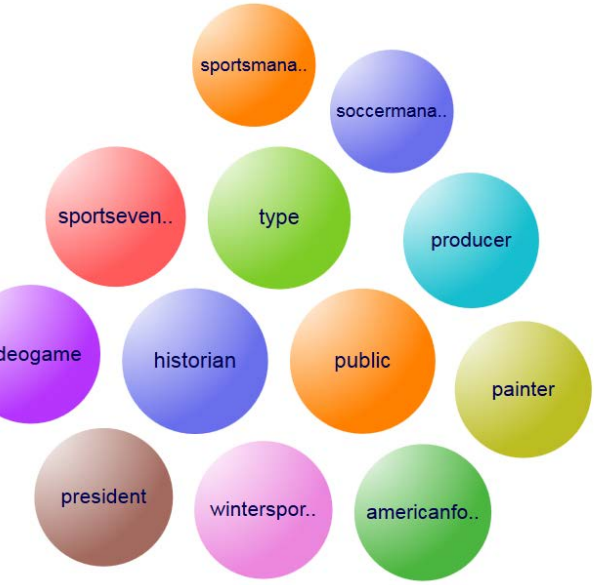


RDF Stores on the Web

- https://www.w3.org/wiki/SparqlEndpoints
  - z.B. BBC, DBPedia, DBLP, data.gov, …

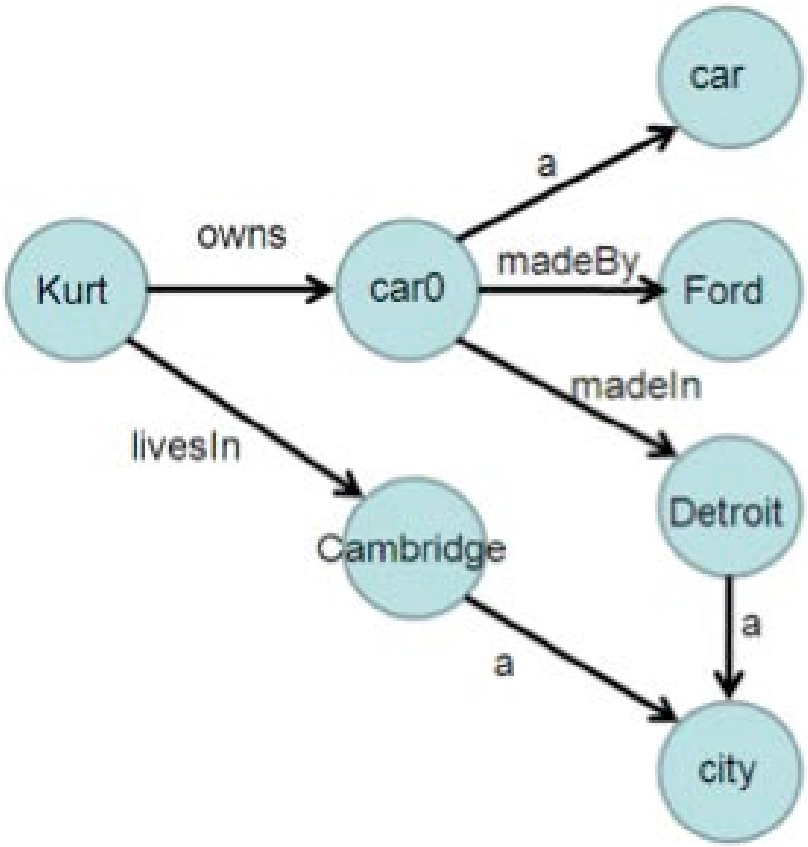


data.gov



DBpedia Bubble Navigator

RDF

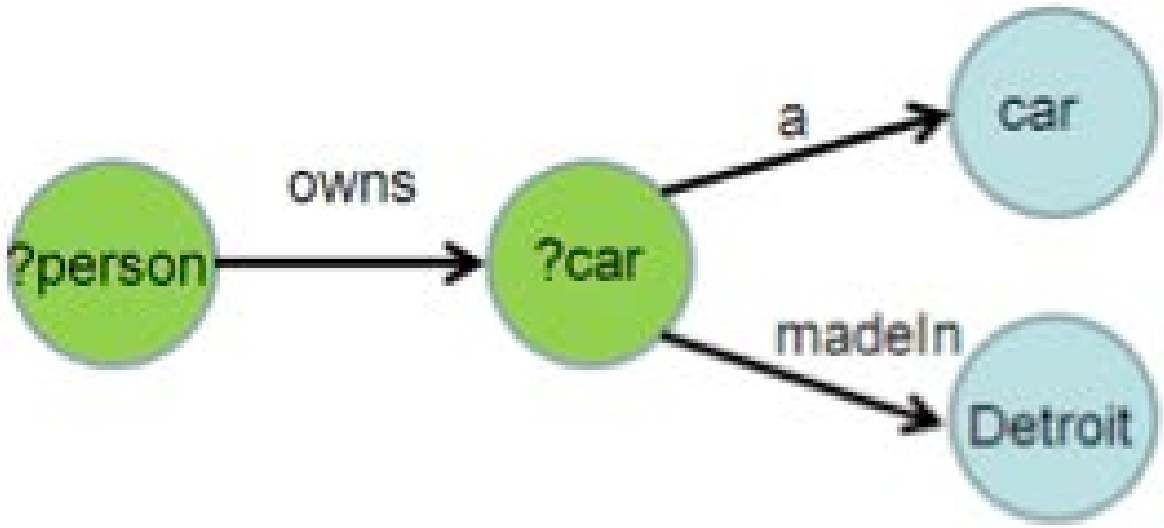


7 triples (in the ABox)

- Kurt lives in Cambridge
- Kurt owns an object car0
- car0 is a car
- car0 was made by Ford
- car0 was made in Detroit
- Detroit is a city
- Cambridge is a city

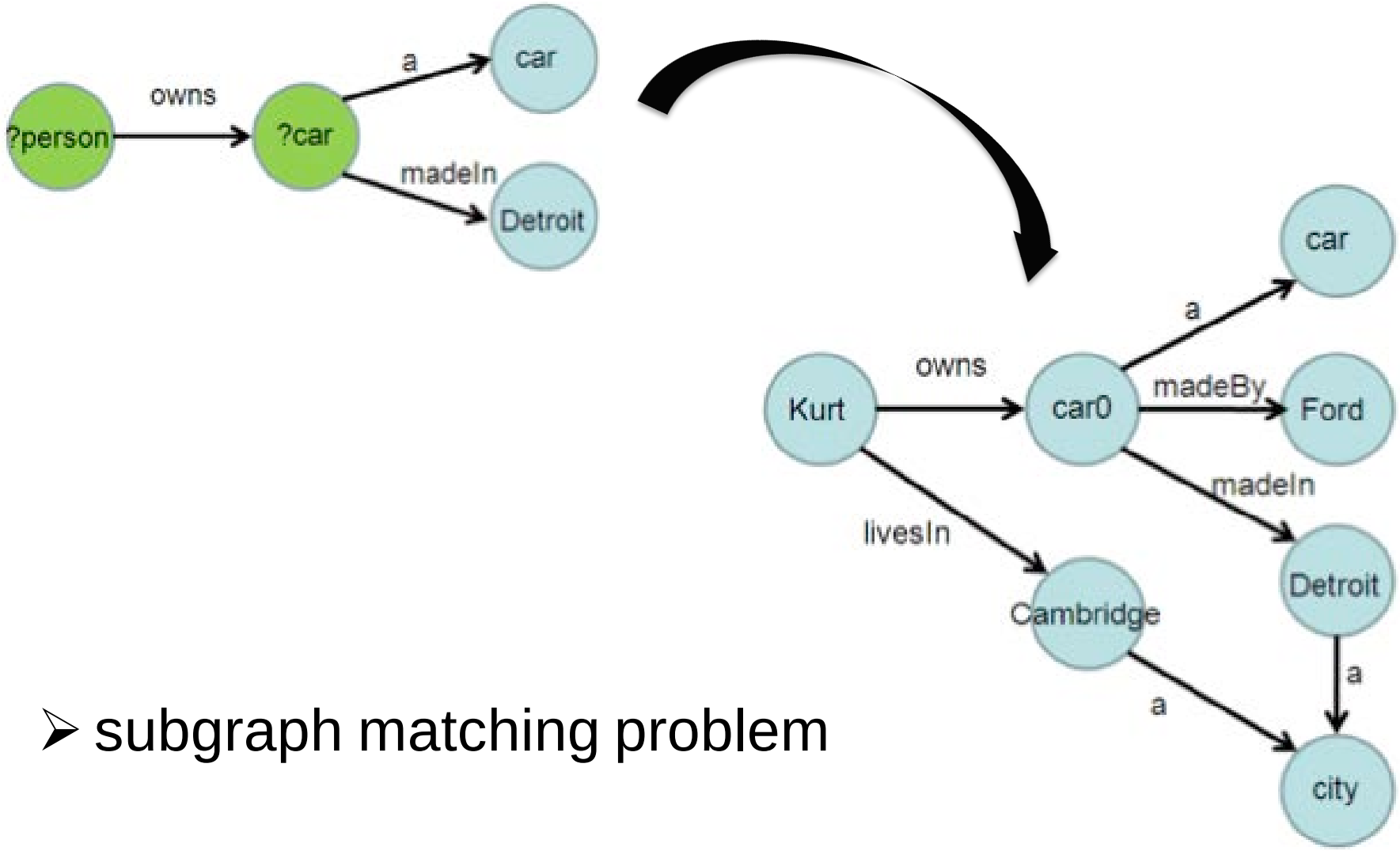
Query an RDF Store - SPARQL

Find all persons who own a car that was made in Detroit



all matches to the variable ?person such that
?person owns an entity represented by the variable ?car
where ?car is a car and was made in Detroit

Matching a Query Against an RDF Triplestore



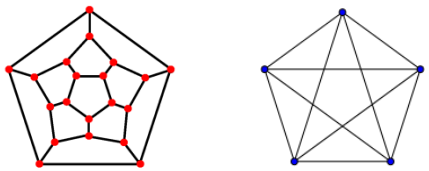
- subgraph matching problem

Subgraph Isomorphism

- NP-complete [Cook, 1971]

Let $G = (V, E)$, $H = (V', E')$ be graphs. Is there a subgraph $G_0 = (V_0, E_0) : V_0 \subseteq V, E_0 \subseteq E \cap (V_0 \times V_0)$ such that $G_0 \cong H$? I.e., does there exist an $f: V_0 \to V'$ such that $(v_1, v_2) \in E_0 \Leftrightarrow (f(v_1), f(v_2)) \in E'$?

- For any fixed pattern H with ℓ vertices
  - polynomial $O(n^{\ell})$ time
- Planar subgraph isomorphism
  - linear time O(n) [Eppstein, 1999]



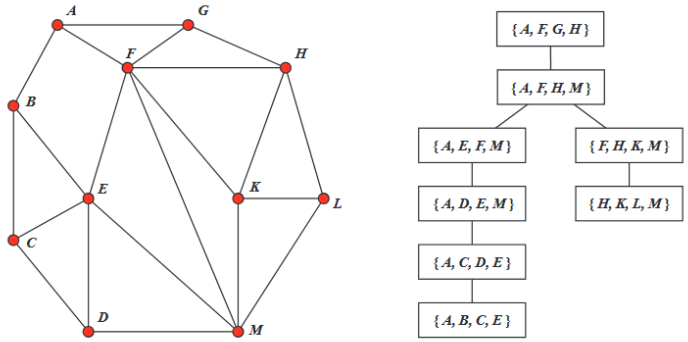
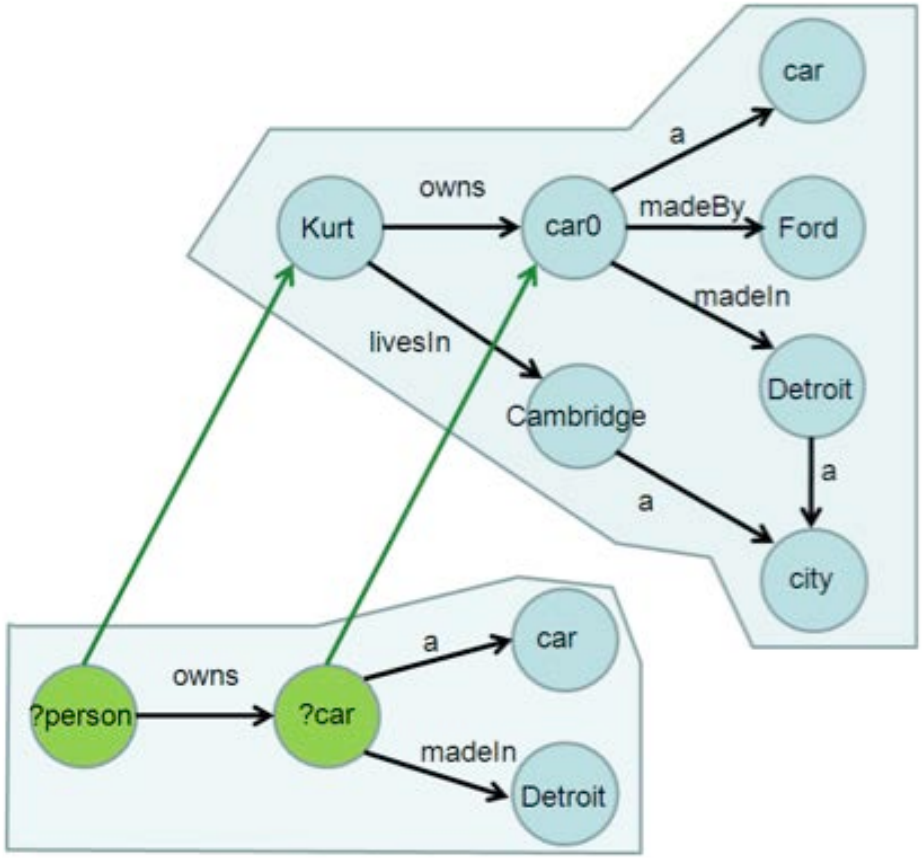


Figure 1: Tree decomposition of a planar graph.

Answering the Query

Bind variables in the query to nodes in the data graph

- such that the query clauses align with the data triples




```
SELECT ?person
WHERE {
?person :owns ?car .
?car :a :car .
?car :madeIn :Detroit .
}
```

SPARQL - SPARQL Protocol and RDF Query Language

Declare prefix shortcuts (*optional*)

```
PREFIX foo: <…>
PREFIX bar: <…>
…
SELECT …
FROM <…>
FROM NAMED <…>
WHERE {
    …
}
GROUP BY …
HAVING …
ORDER BY …
LIMIT …
OFFSET …
VALUES …
```

Define the dataset (*optional*)

Query result clause

Query pattern

Query modifiers (*optional*)

+ Access Protocol (HTTP, SOAP)

Complex Query Patterns

SELECT queries

Project out specific variables and expressions:

```
SELECT ?c ?cap (1000 * ?people AS ?pop)
```

Project out all variables:

```
SELECT *
```

Project out distinct combinations only:

```
SELECT DISTINCT ?country
```

Results in a table of values (in XML or JSON):

?c	?cap	?pop
ex:France	ex:Paris	63,500,000
ex:Canada	ex:Ottawa	32,900,000
ex:Italy	ex:Rome	58,900,000

A . B ⇨ Conjunction

Join results by matching the values of any variables in common.

A OPTIONAL { B } ⇨ Left Join

Join results by matching if possible

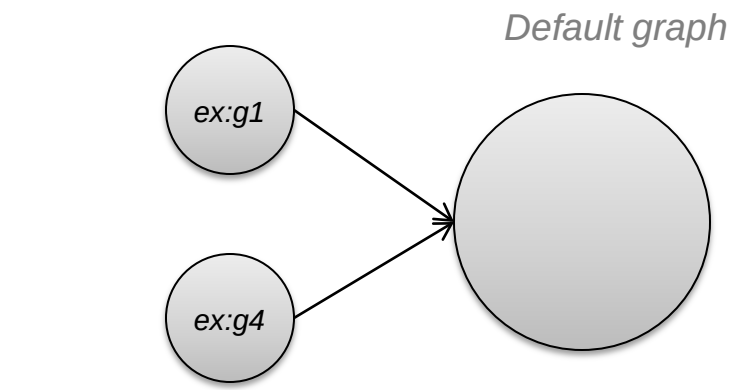
{ A } UNION { B } ⇨ Disjunction

Results of solving A and the results of solving B

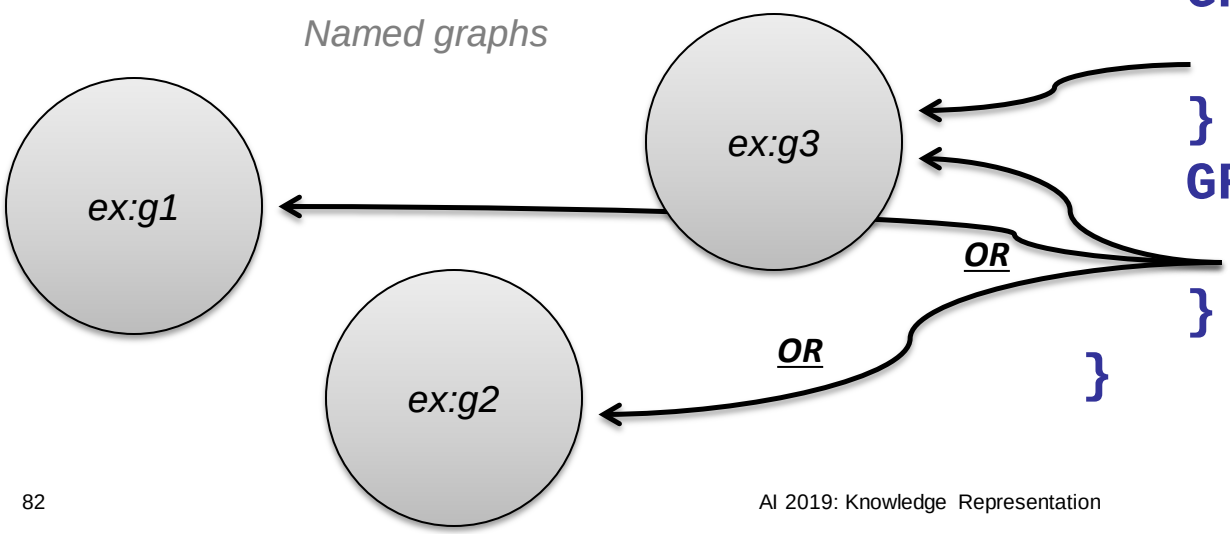
A MINUS { B } ⇨ Negation

Include only results from solving A that are *not compatible* with any of the results from B

Querying Multiple RDF Triplestores



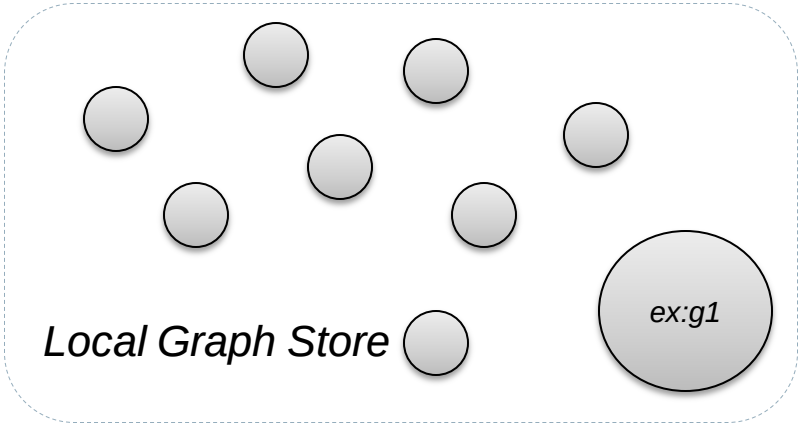

```
PREFIX ex: <…>
SELECT …
FROM ex:g1
FROM ex:g4
FROM NAMED ex:g1
FROM NAMED ex:g2
FROM NAMED ex:g3
WHERE {
  … A …
  GRAPH ex:g3 {
    … B …
  }
  GRAPH ?graph {
    … C …
  }
}
```



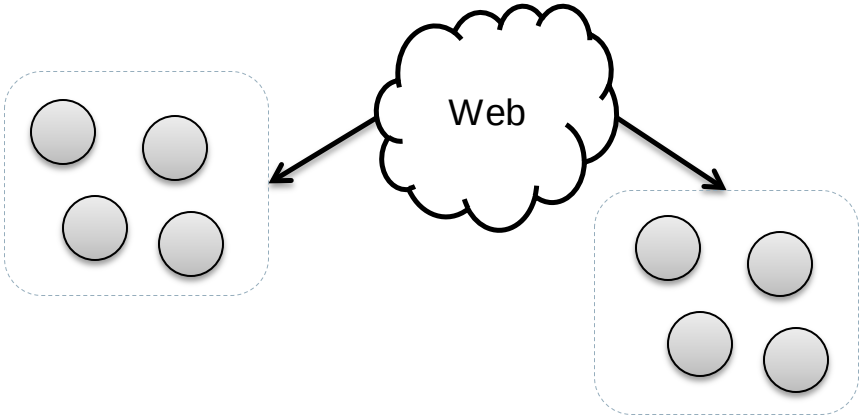
Distributing Queries over Multiple Triplestores

(1) Query a local collection of stores

- Build local store with copies of relevant external stores



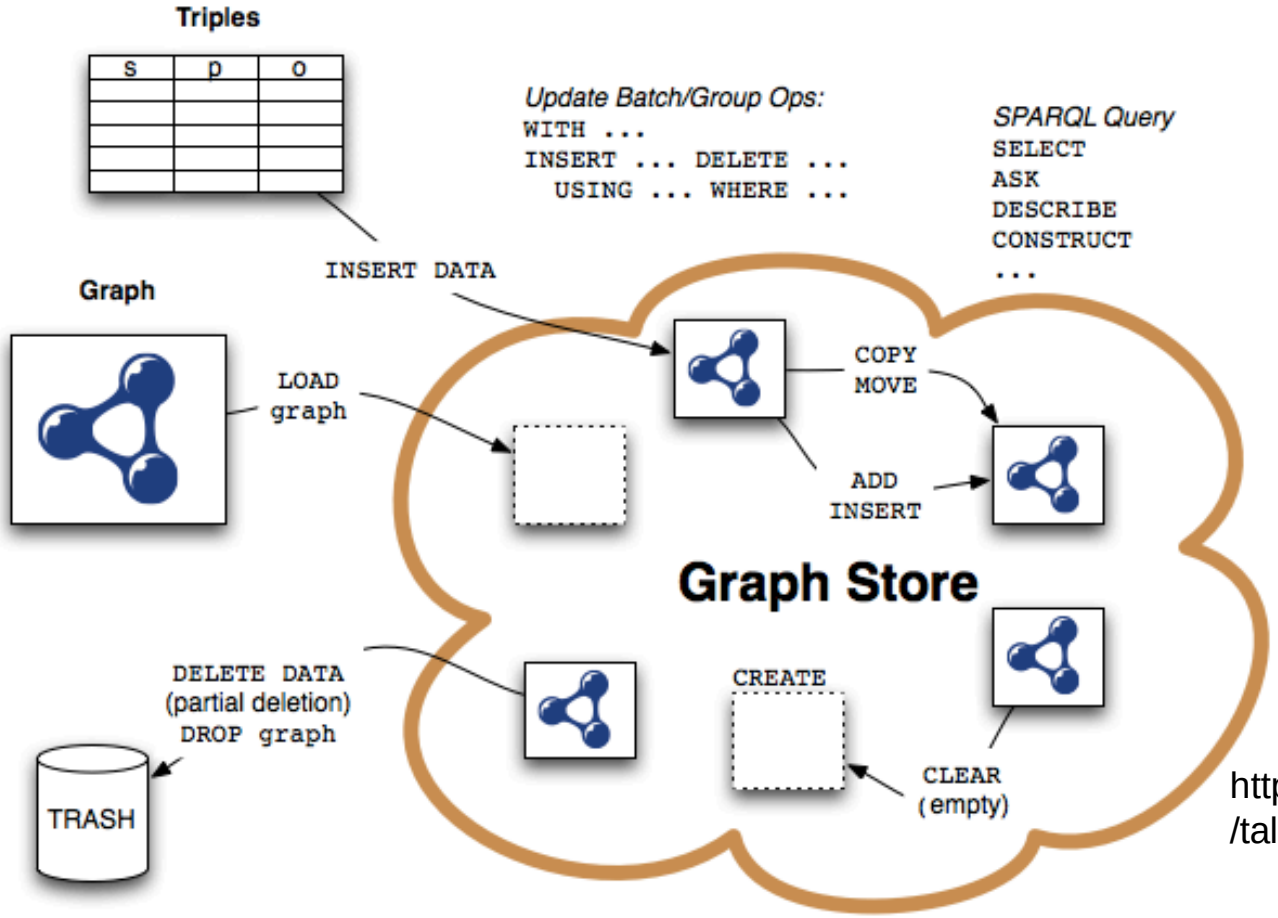
(2) Issue follow-up queries to external stores



(3) Use Query Federation

(4) Automatic Link Traversal

(1) Build Local Store



https://www.dajobe.org/talks/201105-sparql-11/

- Reduce to the problem of querying a single store
- All relevant sources must be integrated and up to date

(2) Follow-Up Queries

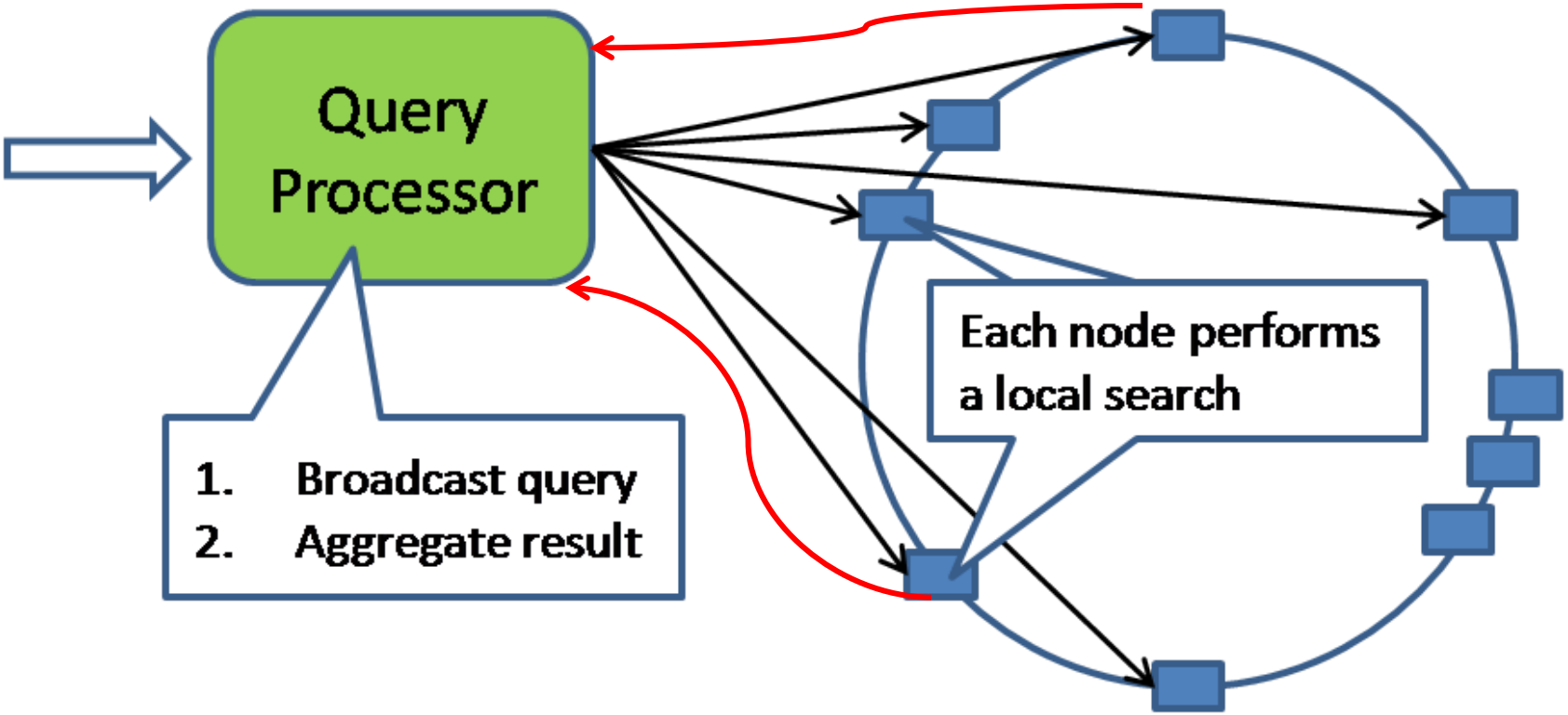
- Take results from a first query to substitute placeholders in subsequent query templates

➢ Need to write explicit query program logic

```
String source = "http://cb.semsol.org/sparql";
String source2 = "http://dbpedia.org/sparql";
String query = "SELECT ?s WHERE { ...";
ResultSet set =
QueryExecutionFactory.sparqlService(source,query).execSelect();
while (set.hasNext()) {
        ....
        ResultSet  set2=
QueryExecutionFactory.sparqlService(…).execSelect();
        while ( set2.hasNext() ) {
                ...
        }
}
....
```

(3) Query Federation

- Query a mediator which distributes subqueries to relevant sources and integrates the retrieved results



➢ Mediator solves the problem

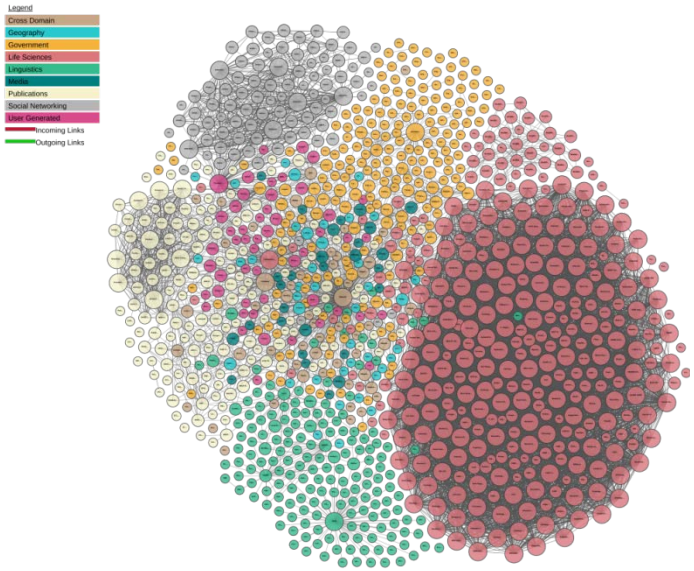
(4) Automatic Link Traversal

- Traverse RDF links during query evaluation

Subject: http://dig.csail.mit.edu/data#DIG
Predicate: http://xmlns.com/foaf/0.1/ member
Object: http://www.w3.org/People/Berners-Lee/card#i

"Tim Berners-Lee is a member of the MIT Decentralized Information Group"

- No need to know all data sources in advance
- No need to write program logic
- Queried data is up to date
- But: Can take very long
  - unsuitable for some queries
  - results might be incomplete



Important Research Directions in KR&R

- Argumentation, explanation finding, causal reasoning, abduction
- Belief revision and update, belief merging
- Computational aspects of knowledge representation
- Similarity-based and contextual reasoning
- Inconsistency- and exception tolerant reasoning, paraconsistent logics
- Reasoning about preferences
- Preference-based reasoning
- Qualitative reasoning, reasoning about physical systems
- Reasoning about actions and change
- Spatial reasoning and temporal reasoning
- Uncertainty, representations of vagueness

Working Questions

1. What type of knowledge do we encode with description logics? What other types of knowledge do you know?
2. Explain the difference between the knowledge represented in the TBox and the one in the ABox?
3. Why are DLs of different expressivity defined?
4. Given a statement in natural language, can you encode it in the description logic ALC?
5. What operators does ALC contain?
6. Which DL reasoning services do you know? Explain them.
7. What can you say about the complexity of DL reasoning?
8. How can we compute concept subsumption?

Working Questions Continued

9. Explain informally what nonmonotic reasoning is.
10. What is the frame problem in AI?
11. What is OWL? How does it relate to DLs such as ALC?
12. How does an OWL ontology relate to a DL ABox/TBox?
13. Do you know examples of OWL ontologies on the web?
14. How can we query OWL ontologies?
15. Which computational problem is at the core of Sparql queries?

Recommended Reading

- S. Russell, P. Norvig: Artificial Intelligence – A Modern Approach, 3rd Edition, Chapter 12, http://aima.cs.berkeley.edu/
- Knowledge Representation & Reasoning by R. Brachman, H. Levesque: Morgan Kaufmann 2004. (available online)
- F. Baader, C. Lutz, I. Horrocks, U. Sattler: An Introduction to Description Logic. Cambridge University Press, 2017.
- F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2nd edition, 2007.
- M. Gelfond, Y. Kahl: Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach, Cambridge University Press, 2014.