Artificial Intelligence

Knowledge Representation

Prof. Dr. habil. Jana Koehler

Artificial Intelligence - Summer 2020

Deep thanks goes 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 $\mathcal{ALC}$
  - ABox and TBox representations
  - 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

Recommended Reading

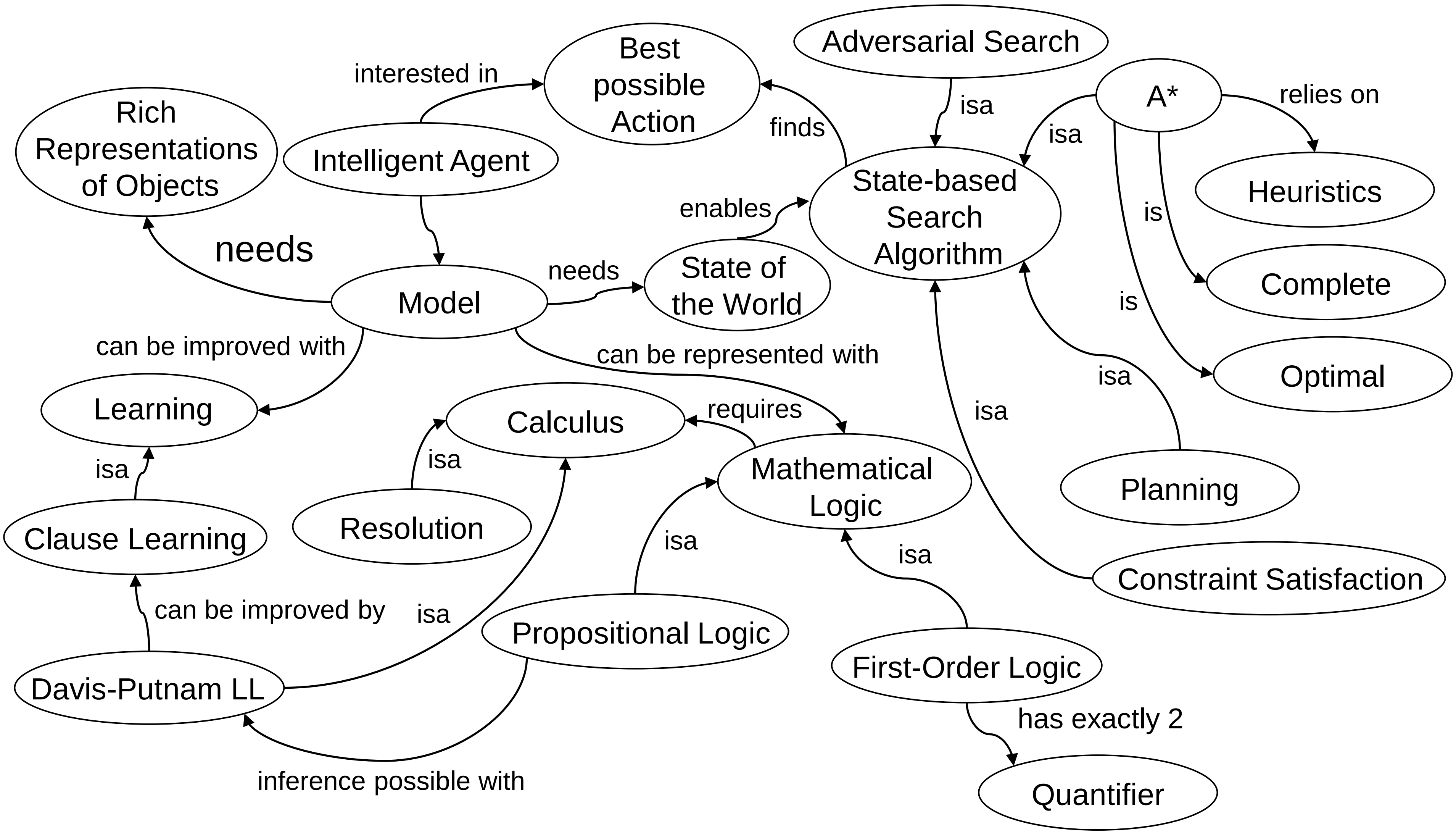
- AIMA Chapter 12: Knowledge Representation
  - 12.1 Ontological Engineering
  - 12.2 Categories and Objects
  - 12.5 Reasoning Systems for Categories
  - 12.7 The Internet Shopping World
  - 12.8 Summary

Additional Reading

- 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

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, they need to know
  - relevant 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 (objects): Citroen C3 “SB..”
    - properties: “car has wheels = exactly 4”
    - concept-concept relations: “a car is a moving vehicle”
  - actions they can perform and how these affect the domain´s objects
    - For example, PDDL and STRIPS are popular formalisms
  - temporal relationships between events, spatio-temporal relations between objects, physical laws,…

Knowledge Representation and (!) Reasoning

- How can agents exploit the knowledge they have?

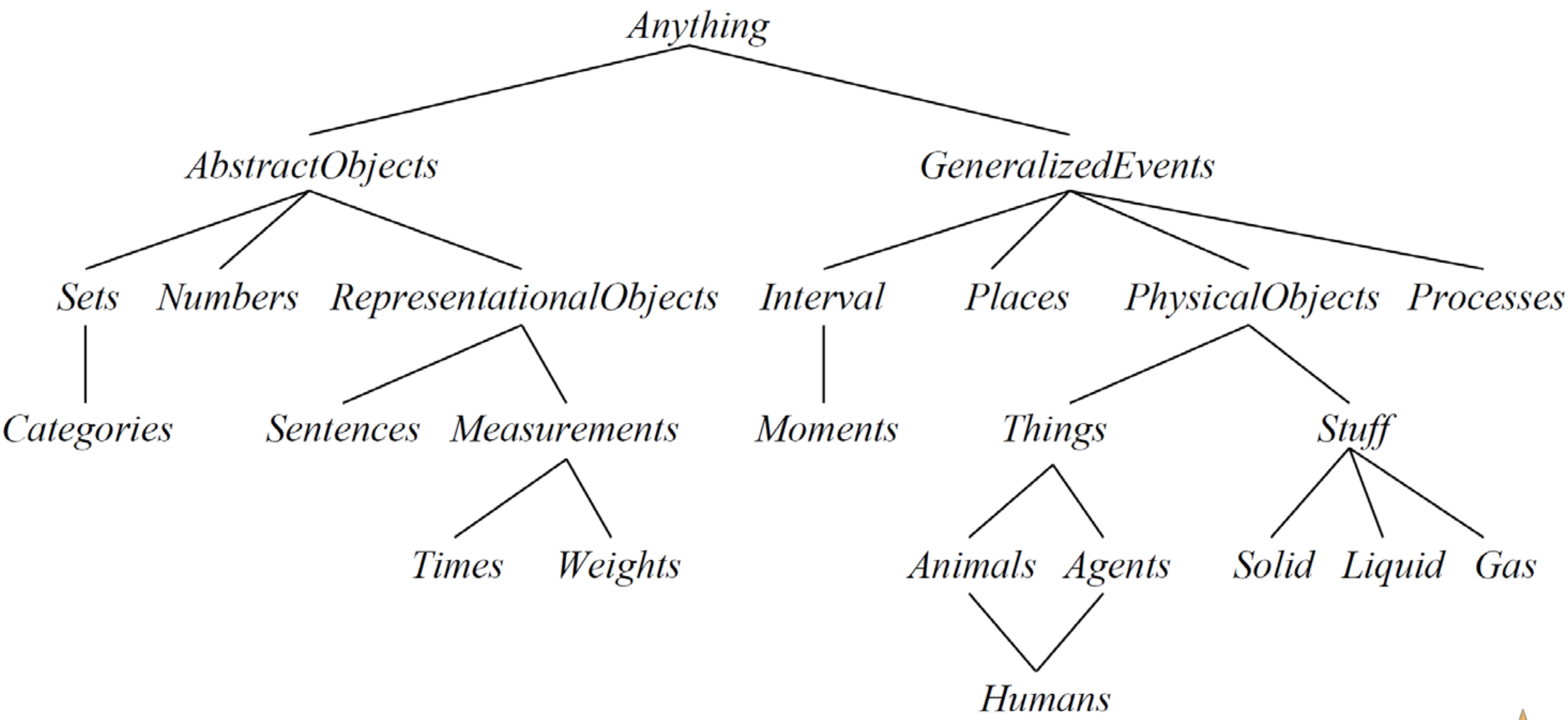
- They need some reasoning component to ask various questions about the objects and concepts in their knowledge base
  - Is “Citroen C3 SB-CH…” a moving vehicle?
  - How many wheels does it have?
  - Which other cars does the agent know about?
  - Are there moving vehicles which are not cars?

➢ 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?

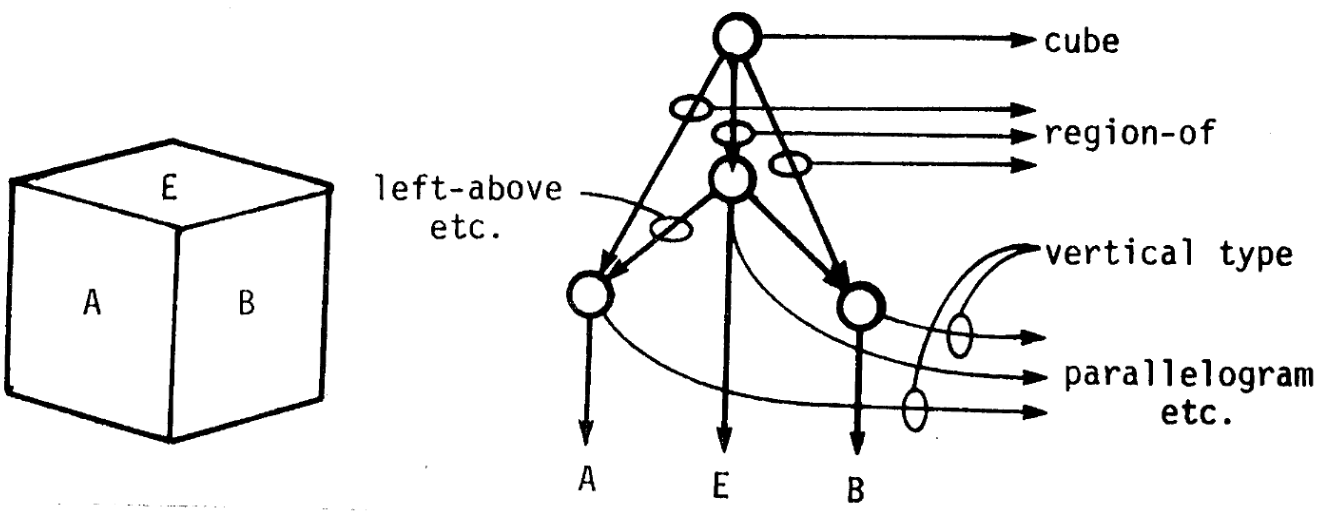
- Use definitions and/or necessary conditions referring to other already defined concepts:
  - A *parent is a human with at least one child*.

- More complex description:
  - A proud-grandmother is a human, who is female with at least two children who are parents and 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

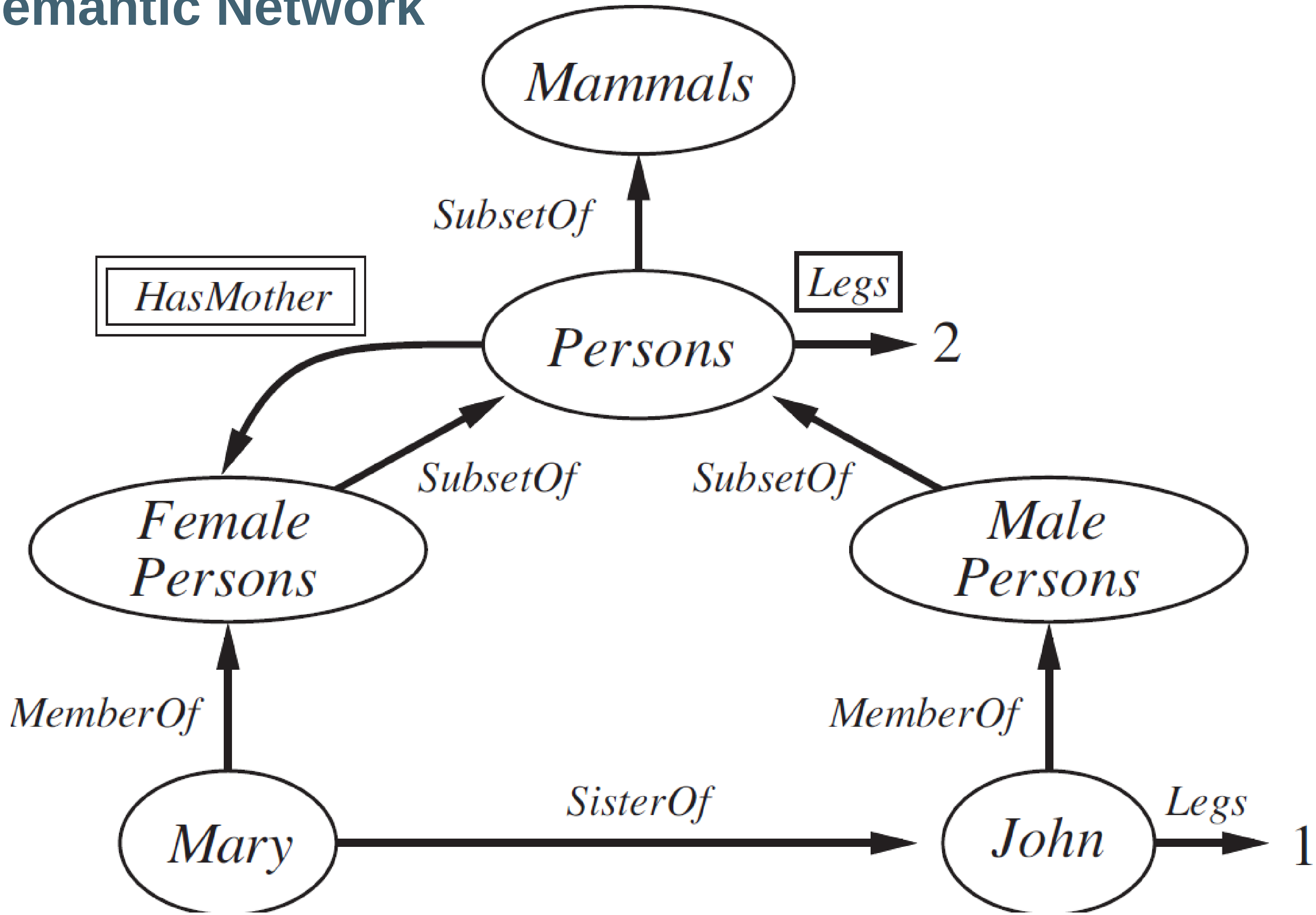
The substructures "A" and "B" represent details or decorations on two faces of the cube. When we move to the right, face "A" disappears from view, while the new face decorated with "C" is now seen. If we had to reanalyse the scene from the start, we would have to

(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 semantic networks as a fragment of first-order predicate logic (PL1)
  - 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 problems in most DLs are decidable
  - A family of DL languages of varying complexity (KL-ONE, CLASSIC, ALC, OWL) was developed over the years

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

$$
\begin{array}{lcl}
\text{Woman} & \equiv & \text{Person} \sqcap \text{Female} \\
\text{Man} & \equiv & \text{Person} \sqcap \neg\text{Female} \\
\text{Mother} & \equiv & \text{Woman} \sqcap \exists \text{has.Child}.\top \\
\text{Person} & \equiv & \text{Man} \sqcup \text{Woman} \\
\bot & \equiv & \text{Male} \sqcap \text{Female}
\end{array}
$$

Person(JOHN), Person(MARC),
Person(STEPHEN),
Person(JASON),
Person(MICHELLE),
Person(ANNA), Person(MARIA)

hasChild(STEPHEN, MARC)
hasChild(MARC, ANNA)
hasChild(JOHN, MARIA)
hasChild(ANNA, JASON)

Male(JOHN), Male(MARC),
Male(STEPHEN), Male(JASON),
Female(MICHELLE),
Female(ANNA), Female(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 representation and reasoning 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

$$\forall x, y \colon \big(male(y) \land \exists z \colon \big(has\_child(x, z) \land has\_child(z, y)\big) \longrightarrow has\_grandson(x, y)\big)$$

$has\_child(John, Mary)$

$has\_child(Mary, Paul)$

$male(Paul)$

Implicit knowledge:
$has\_grandson(John, Paul)$

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 w.r.t. 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, see Schmidt-Schauß & Smolka, 1991

Naming scheme:

- Basic language $\mathcal{AL}$
- Extended with constructors whose “letter” is added after $\mathcal{AL}$
- $\mathcal{C}$ stand for complement, i.e., $\mathcal{ALC}$ is obtained from $\mathcal{AL}$ by adding the complement operator ($\neg$)

A Description Logic System

description language

TBox
defines the terminology of the application domain

ABox
states facts about a specific "world"

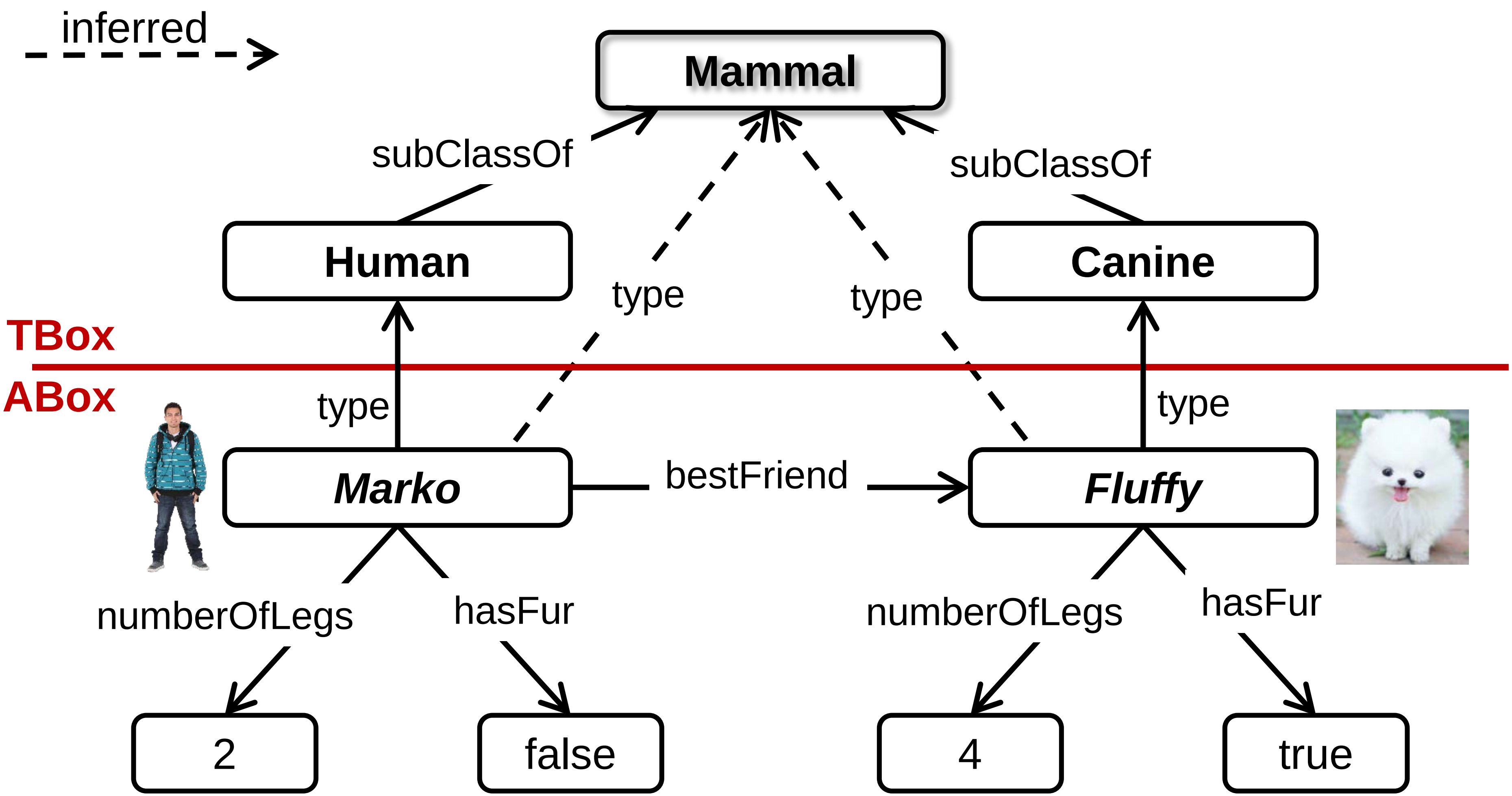
knowledge base

reasoning component

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

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

An Example


inferred
Mammal
subClassOf
subClassOf
Human
Canine
type
type
TBox
ABox
type
type
Marko
bestFriend
Fluffy
numberOfLegs
hasFur
numberOfLegs
hasFur
2
false
4
true

Syntax of $\mathcal{ALC}$

Let $\mathbf{C}$ and $\mathbf{R}$ 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)
  - $\forall r.C$ (universal role value restriction)
  - $\exists r.C$ (existential role value 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

- Person ⊓ Female
- Participant ⊓ ∃attends.Talk
- Participant ⊓ ∀attends.(Talk ⊓ ¬Boring)
- Speaker ⊓ ∃gives.(Talk ⊓ ∀topic.DL)
- Speaker ⊓ ∀gives.(Talk ⊓ ∃topic.(DL ⊔ FuzzyLogic))

Notation

- Concept names are called **atomic**
- All other descriptions are called **complex**
- Instead of $\mathcal{ALC}$-concept description we often say $\mathcal{ALC}$-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 $\mathcal{ALC}$

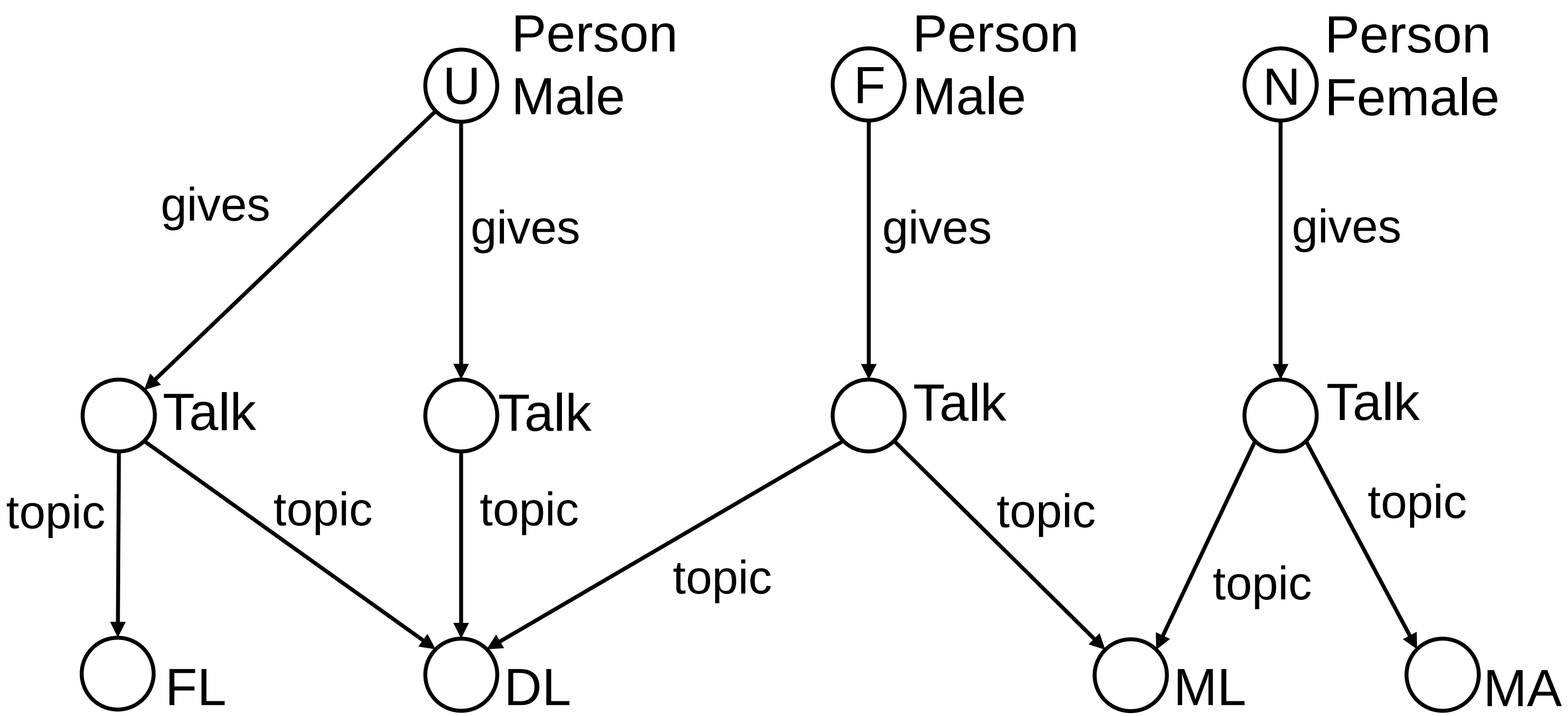
An interpretation $I = (\Delta^I, \cdot^I)$ consists of a non-empty domain Δ^I and an extension mapping $\cdot^I$:

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

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

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

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 corresponds to first-order formulae with one free variable
- Given such a formula $\varphi(x)$ with the free variable x and an interpretation I, the extension of φ w.r.t. I is given by
- $\varphi^I := \{d \in \Delta^I \mid I \vDash \varphi(d)\}$
- We can translate $\mathcal{ALC}$-concepts C into first-order formulae $\tau_x(C)$ such that their extensions coincide

The TBox

- A general concept inclusion (GCI) is of the form $C \sqsubseteq D$ where C, D are concept descriptions

- A TBox is a finite set of GCIs

- An interpretation I satisfies a GCI $C \sqsubseteq D$ iff $C^I \subseteq D^I$

- An interpretation I is a model of the TBox T iff it satisfies all GCIs in T

➢ Two TBoxes are equivalent if they have the same models

Acyclic TBox

An acyclic TBox is a finite set of concept definitions, which

- do **not** contain multiple definitions

$$A \equiv C$$
$$A \equiv D$$
for $C \neq D$

- do **not** contain cyclic definitions

$$A \equiv P \sqcap \forall r.(\exists r.A) \sqcap \forall r.P$$

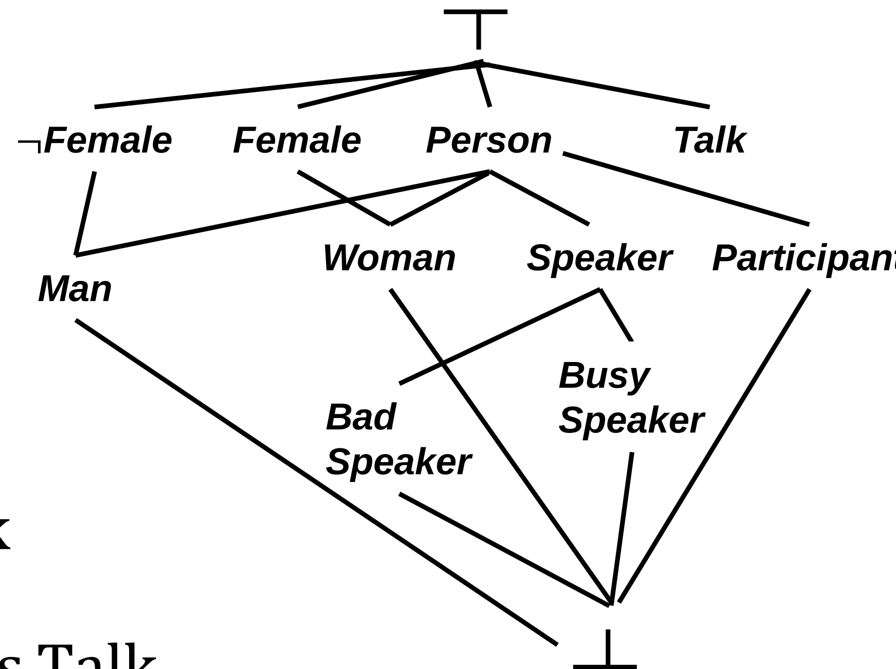
$$A \equiv B \sqcap \forall r.P$$
$$B \equiv P \sqcap \forall r.C$$
$$C \equiv \exists r.A$$

A TBox T does not contain cyclic definitions iff there is no sequence $A_1 \equiv C_1, \ldots, A_n \equiv C_n \in T$ $(n \geq 1)$ such that

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

Concept Definitions in an Acyclic Tbox and GCI

- $\text{Woman} \equiv \text{Person} \sqcap \text{Female}$
- $\text{Man} \equiv \text{Person} \sqcap \neg\text{Female}$
- $\text{Talk} \equiv \exists\text{topic}.\top$
- $\text{Speaker} \equiv \text{Person} \sqcap \exists\text{gives}.\text{Talk}$
- $\text{Participant} \equiv \text{Person} \sqcap \exists\text{attends}.\text{Talk}$
- $\text{BusySpeaker} \equiv \text{Speaker} \sqcap (\geq 3\ \text{gives}.\text{Talk})$
- $\text{BadSpeaker} \equiv \text{Speaker} \sqcap \forall\text{gives}.(\forall\text{attends}^{-}.(\text{Bored} \sqcup \text{Sleeping}))$



if r is a role, then r^{-} denotes its inverse: $(r^{-})^{I} := \{(e,d) | (d,e) \in r^{I}\}$

The ABox

An **ABox A** is a finite set of assertions

- An assertion is of the form
  $a : C$ (concept assertion) or $(a, b) : r$ (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 $\mathbf{C}, \mathbf{R}$
- I assigns elements a^I of Δ^I to individual names $a \in \mathbf{I}$
- An interpretation I is a model of an ABox $\boldsymbol{A}$ if it satisfies all its assertions:
  $a^I \in C^I$ for all $a : C \in \boldsymbol{A}$
  $(a^I, b^I) \in r^I$ for all $(a, b) : r \in \boldsymbol{A}$

Example of an ABox

FRANZ : Lecturer

TU03 : Tutorial

REASONINGinDL : DL

(FRANZ, TU03) : teaches

(TU03, REASONINGinDL) : topic

Knowledge Bases

A knowledge base $KB = (T, A)$ consists of a TBox T and an ABox A

The interpretation I is a model of the knowledge base $KB = (T, A)$ iff it is a model of T and a model of 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 T be a TBox

Satisfiability:

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

Subsumption:

C is subsumed by D w.r.t. T ($C \sqsubseteq_T D$) iff
$C^I \subseteq D^I$ for all models I of the TBox T

Equivalence:

C is equivalent to D w.r.t. T ($C \equiv_T D$) iff
$C^I = D^I$ for all models I of the TBox T

Examples

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

Formalization of Assertional Reasoning

Let $KB = (T, A)$ be a knowledge base

Consistency:

KB is consistent iff there exists a model of KB

Instance:

a is an instance of C w.r.t. KB iff $a^I \in C^I$ for all models I of KB

Realization

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

$$
\begin{array}{lcl}
\mathrm{Woman} & \equiv & \mathrm{Person} \sqcap \mathrm{Female} \\
\mathrm{Man} & \equiv & \mathrm{Person} \sqcap \neg \mathrm{Female} \\
\mathrm{Mother} & \equiv & \mathrm{Woman} \sqcap \exists \mathrm{has.Child}.\top \\
\mathrm{Person} & \equiv & \mathrm{Man} \sqcup \mathrm{Woman} \\
\bot & \equiv & \mathrm{Male} \sqcap \mathrm{Female}
\end{array}
$$

Person(JOHN), Person(MARC), Person(STEPHEN), Person(JASON), Person(MICHELLE), Person(ANNA), Person(MARIA)

hasChild(STEPHEN, MARC)
hasChild(MARC, ANNA)
hasChild(JOHN, MARIA)
hasChild(ANNA, JASON)

Male(JOHN), Male(MARC), Male(STEPHEN), Male(JASON), Female(MICHELLE), Female(ANNA), Female(MARIA)

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

More Equivalences

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

$C \equiv_T D$ iff $C \sqsubseteq_T D$ and $D \sqsubseteq_T C$

$C \sqsubseteq_T D$ iff $C \equiv_T C \sqcap D$

$C \sqsubseteq_T D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. T

C is satisfiable w.r.t T iff $C \not\sqsubseteq_T \bot$

C is satisfiable w.r.t T iff $(T, \{a : C\})$ is consistent

a is an instance of C w.r.t KB iff $(T, A \cup \{a: \neg C\})$ is inconsistent

KB is consistent iff a is not an instance of $\bot$ w.r.t. KB

Complexity of Reasoning in $\mathcal{ALC}$

- Satisfiability of a concept description w.r.t. a TBox is decidable for $\mathcal{ALC}$
- Concept satisfiability and subsumption w.r.t. acyclic TBoxes are in PSpace for $\mathcal{ALC}$
- Concept satisfiability and subsumption w.r.t. general TBoxes are in ExpTime for $\mathcal{ALC}$

Complexity classes:

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

Expressivity and Undecidability

- Consider the following part of a TBox about universities:

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

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

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

- 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

Fundamental Challenges in Knowledge Bases

- **Qualification problem**: specifying all exceptions is infeasible
  - *Pepper can follow you unless it cannot detect you, its batteries are empty, its vision system is broken, the ground is slippy, you are too fast,…*
- **Frame problem**: cannot explicitly specify what does <u>not</u> change when an action is executed
  - *When Pepper answers a question, the furniture will stay in place, it will not move outside a certain range, it will not loose any information, …*
- **Ramification problem**: how to represent what happens implicitly due to an action
  - *When Pepper grasps a box and moves, the box will move with the robot, all objects inside the box will also move with the box, the beads above the little box inside the big box will fall into the big box, …*

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 it is 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 in 1985
- Ilkka Niemelä and others developed *Answer Set Programming* (ASP) in 1999

Belief Revision

- Process of changing beliefs in the light of a new piece of information
- To understand how **nonmonotonic reasoning** requires the revision of beliefs, consider a standard example:

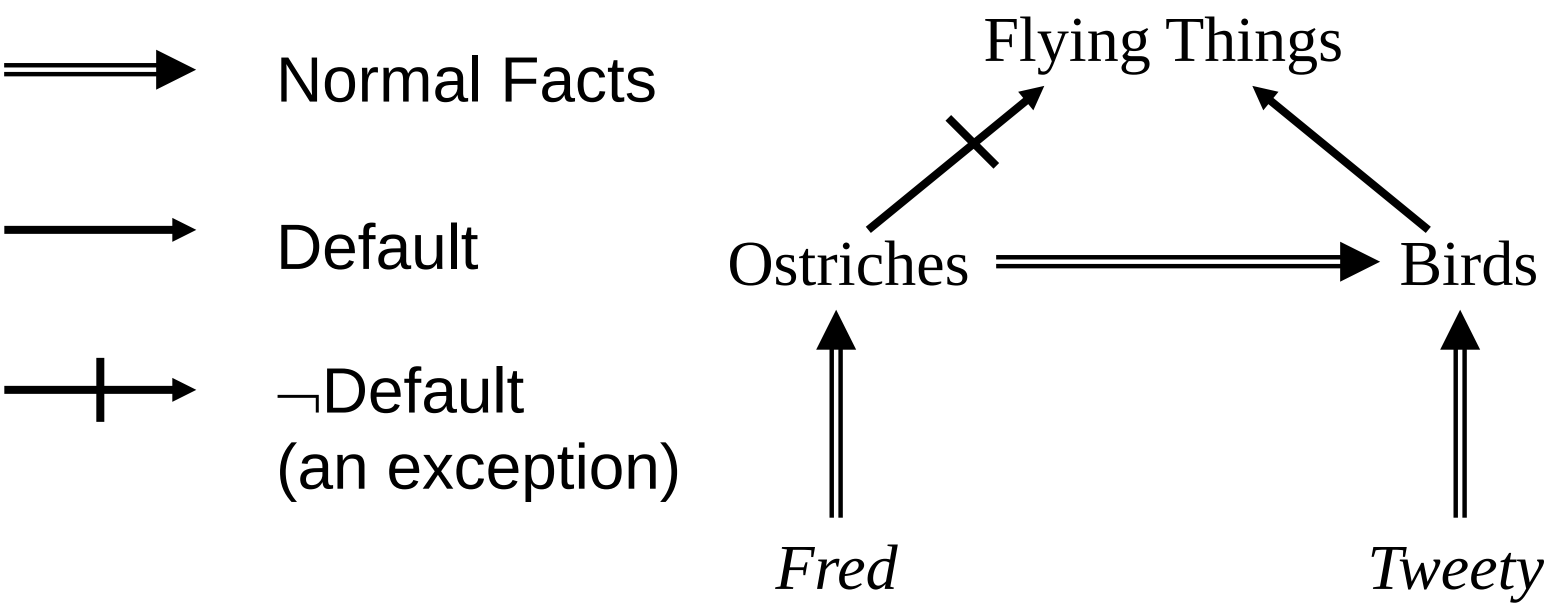
“All birds fly."

"Tweety is a bird."

"Does Tweety fly?“

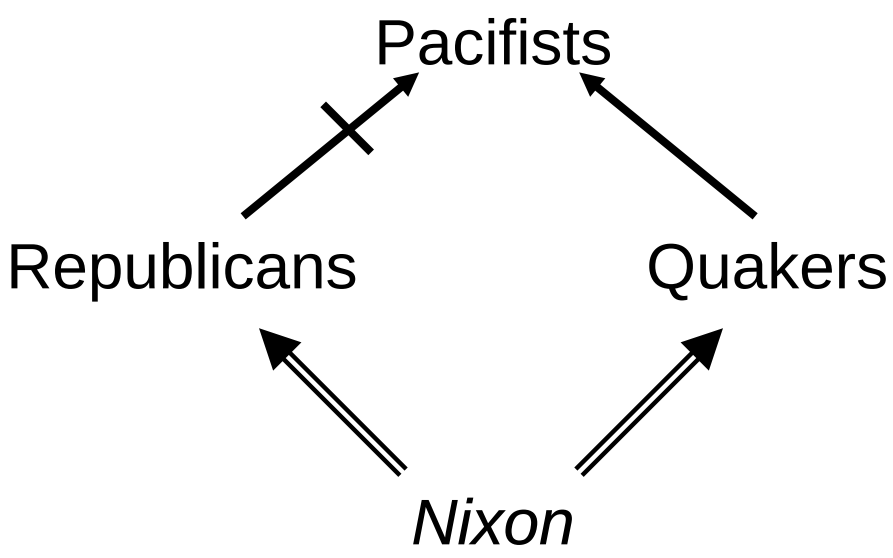
- The obvious answer is yes,
  - however what if later we learned that Tweety had a broken wing, then the answer becomes no,
  - what if we later learned that Tweety was a human airplane pilot and that the information of it being a bird was wrong …

A Historic Formalism: Inheritance Diagrams



The Nixon Diamond

- Quakers are pacifists
- 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?

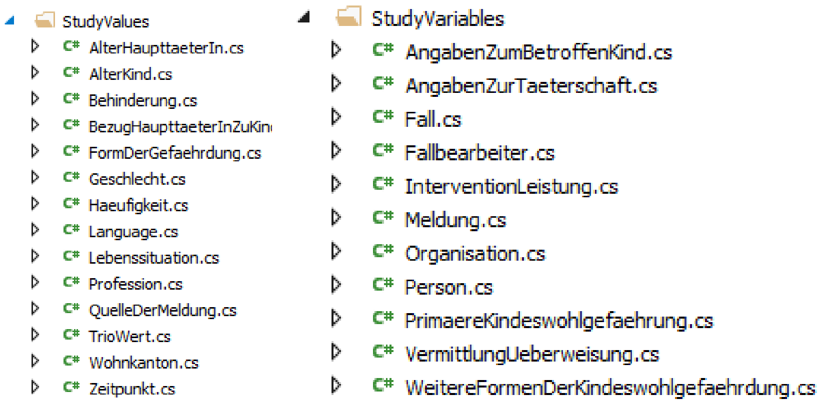
- Pat is a Bat.
- Bats are Mammals.
- Bats can fly.
- Bats have 2 legs.
- Mammals cannot fly.
- Mammals have 4 legs.

Web Ontologies and the W3C OWL Standard

Description Logic and Ontologies

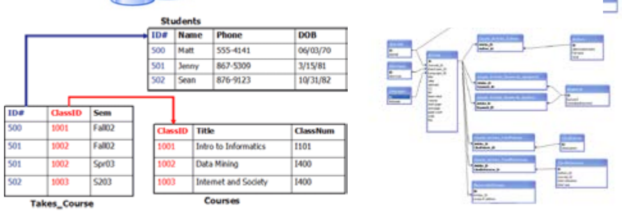
- The W3C standards for OWL "Ontology Web Language" is based on Description Logic
- DBpedia is a famous ontology based on OWL
- Cyc https://www.cyc.com/ is another famous ontology based on description logics [Lenat & Guha, 1990]

The World of Data

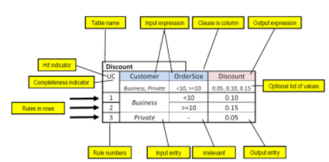


Object-Oriented Programming
Objects, Properties, Methods

Semantic Web
concepts, relationships, ontologies



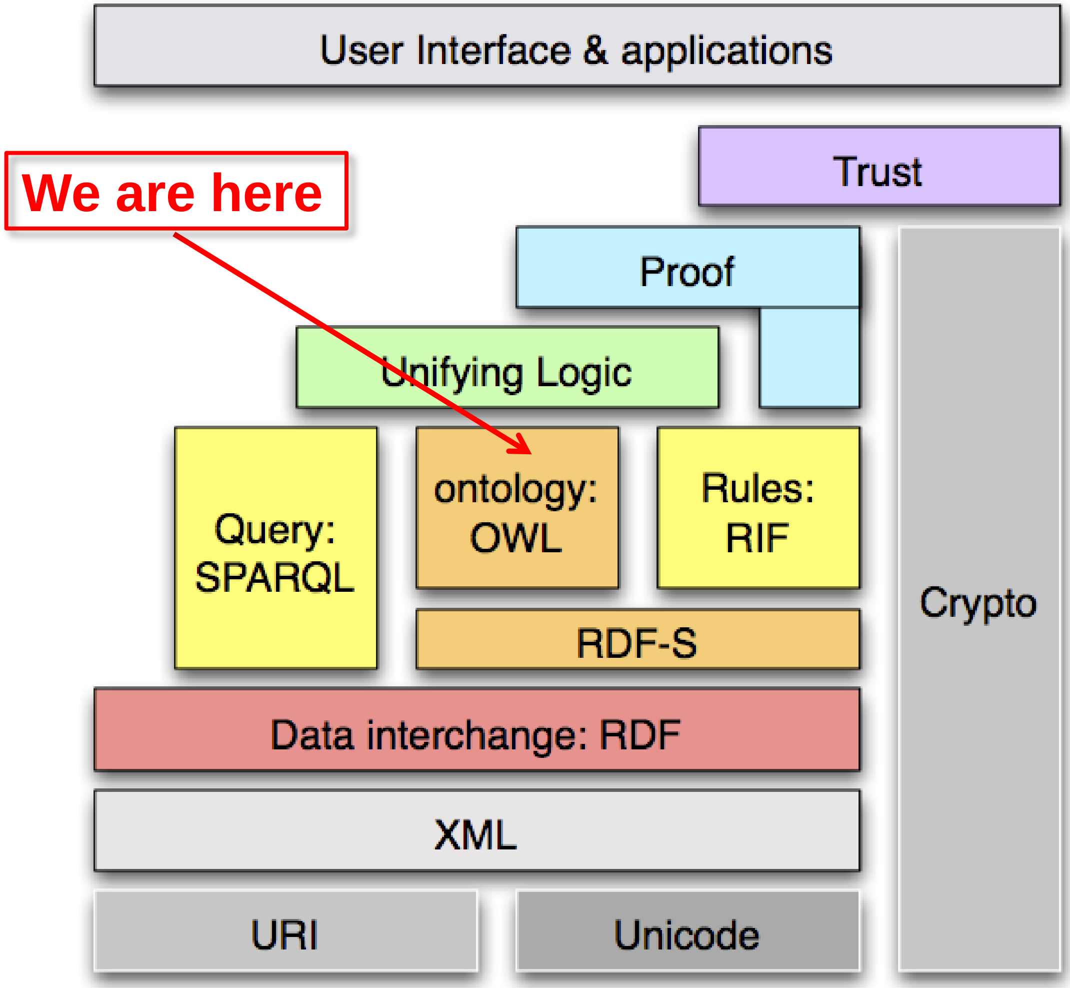
Relational Databases
Entities, Relations, Tables



Deductive Databases
Business Rules …

(Distributed) NoSQL Databases
Attribute-Value-Pairs, Columns/Graphs

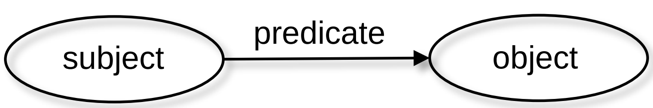
Semantic Web Architecture



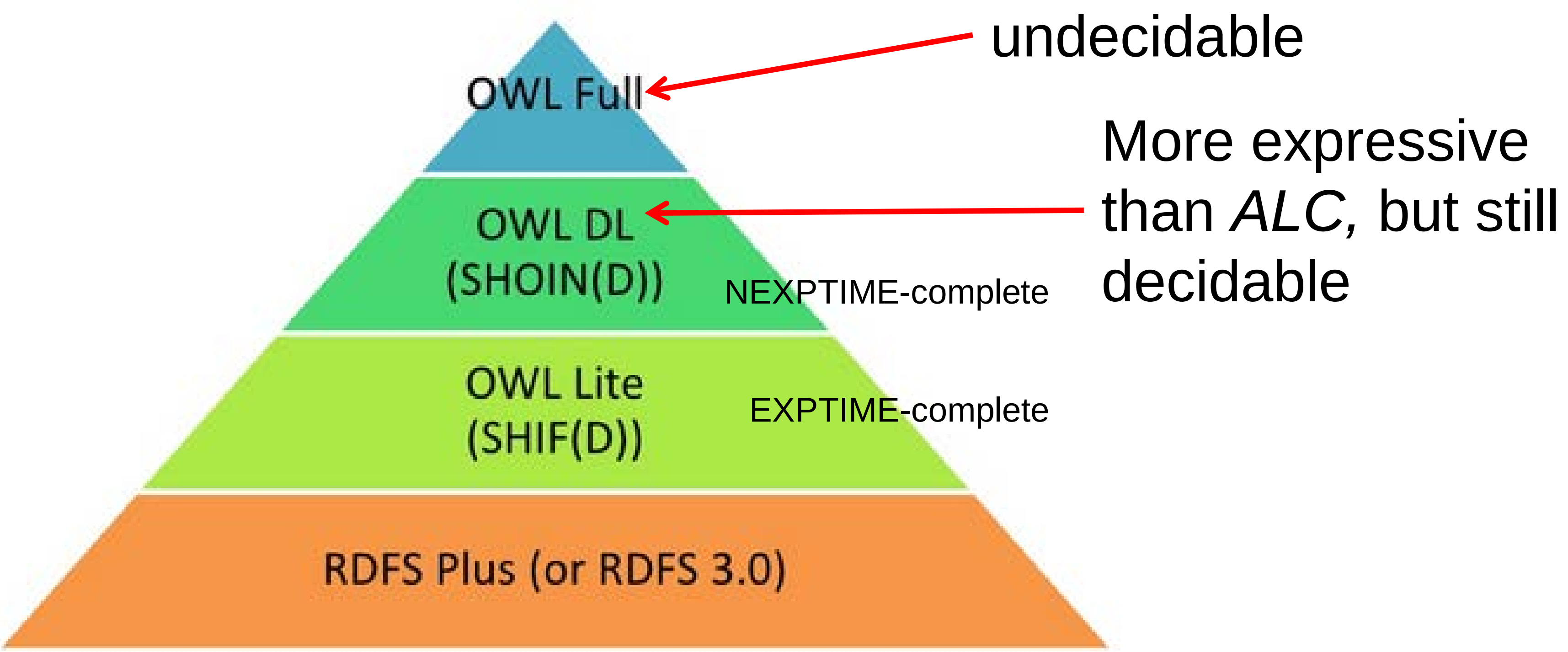
W3C, 2006
http://www.ansta.co.uk/blog/semantic-web-technologies-part-3-94/

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

RDF Triples



The OWL Family



OWL DL is decidable

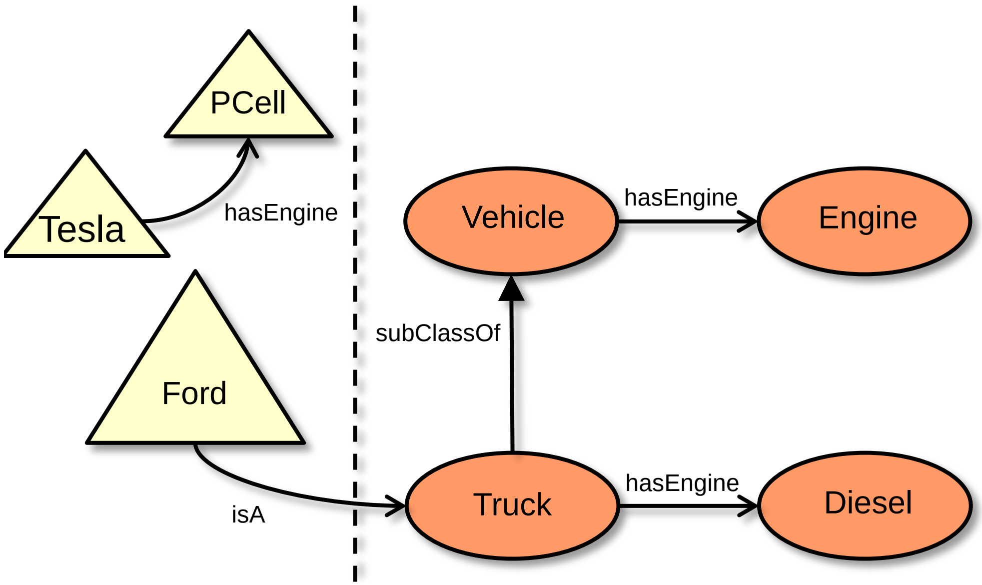
- ➢ The reasoner underlying any application will eventually answer our question!

OWL DL

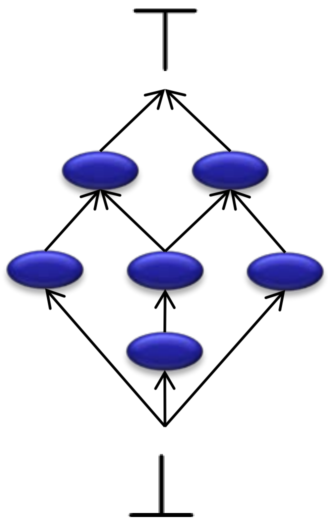
Abstract Syntax	DL Syntax
Descriptions (C)	
A	A
owl:Thing	$\top$
owl:Nothing	$\bot$
intersectionOf($C_1 \ldots C_n$)	$C_1 \sqcap \ldots \sqcap C_n$
unionOf($C_1 \ldots C_n$)	$C_1 \sqcup \ldots \sqcup C_n$
complementOf(C)	$\neg C$
oneOf($o_1 \ldots o_n$)	$\{o_1\} \sqcup \ldots \sqcup \{o_n\}$
restriction(R 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 maxCardinality(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

ABox

TBox



Reasoning in OWL-DL: Class-Class Relationships



- **Class subsumption**

  Given classes $\boldsymbol{C}$ and $\boldsymbol{D}$, determine if $\boldsymbol{C}$ is a subclass of $\boldsymbol{D}$ in the given ontology

  - build the class/subsumption hierarchy

- **Class satisfiability**

  Given a class $\boldsymbol{C}$, determine if $\boldsymbol{C}$ is satisfiable (consistent) in the given ontology

  – $\boldsymbol{C}$ is satisfiable iff $C \not\sqsubseteq \bot$

Reasoning in OWL-DL: Class-Instance Memberships

- **Class-instance membership**
  - **ground**

    Given a class $\boldsymbol{C}$ and an individual $\boldsymbol{a}$, is $\boldsymbol{a}$ an instance of $\boldsymbol{C}$ in knowledge base KB?

  - **open**

    Given a class $\boldsymbol{C}$, determine all the individuals $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \ldots$ in KB that are instances of $\boldsymbol{C}$.

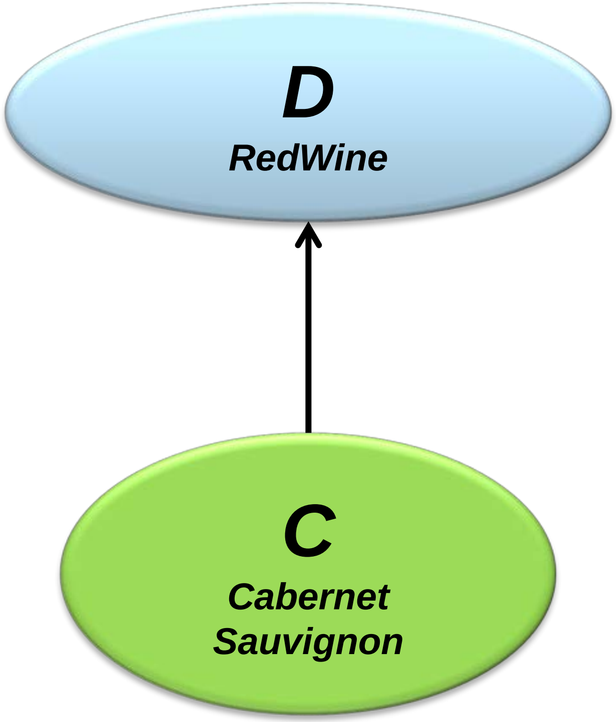
  - **``all-classes"**

    Given an individual $\boldsymbol{a}$, determine all the (named) classes $\boldsymbol{C}, \boldsymbol{D}, \boldsymbol{E}, \ldots$ in KB of which $\boldsymbol{a}$ is an instance of.

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

$$C^I \subseteq D^I$$



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

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

Computing Subsumption By Structural Comparison

1. Put class descriptions into a **normal form representation** exploiting equivalences
   – similar to the computation of normal forms (CNF/DNF) in Propositional and 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

Given two concepts: $C \equiv \neg(\neg A \sqcup \neg B)$ $\qquad D \equiv A$

$Does\ C \sqsubseteq D? \equiv C \rightarrow D?$

We use the following logical equivalences:

$\neg(A \sqcup B) \equiv \neg A \sqcap \neg B$
$\neg(\neg A) \equiv A$

$C \equiv \neg(\neg A \sqcup \neg B) \equiv \neg\neg A \sqcap \neg\neg B$

$\equiv A \sqcap B$

$A \sqcap B \rightarrow A$ which also means that $A \sqcap B \sqsubseteq A$ and thus $C \sqsubseteq D$

Example of Structural Comparison Rules for OWL-DL

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

$R \sqsubseteq S$ role subsumption and role hierarchies

$$\{(x,y)|(x,y) \in R^I\} \subseteq \{(w,z)|(w,z) \in S^I\}$$

$$married \sqsubseteq loves \quad equivalent\ to \quad \forall x \forall y(married(x,y) \rightarrow loves(x,y))$$

$$dogowner \sqsubseteq petowner$$

An Example

RBox $\mathcal{R}$

$$\texttt{owns} \sqsubseteq \texttt{caresFor}$$

"If somebody owns something, they care for it."

TBox $\mathcal{T}$

$$\texttt{Healthy} \sqsubseteq \neg\texttt{Dead}$$

"Healthy beings are not dead."

$$\texttt{Cat} \sqsubseteq \texttt{Dead} \sqcup \texttt{Alive}$$

"Every cat is dead or alive."

$$\texttt{HappyCatOwner} \sqsubseteq \exists\texttt{owns.Cat} \sqcap \forall\texttt{caresFor.Healthy}$$

"A happy cat owner owns a cat and all beings he cares for are healthy."

ABox $\mathcal{A}$

$$\texttt{HappyCatOwner}\,(\texttt{schrödinger})$$

"Schrödinger is a happy cat owner."

From Sebastian Rudolph: Foundations of Description Logics https://www.aifb.kit.edu/images/1/19/DL-Intro.pdf

An Example

$$\mathrm{WellRoundedCo} \equiv$$
$$[\mathbf{AND}\ \mathrm{Company}\ [\mathbf{ALL} : \mathrm{Manager}\ [\mathbf{AND}\ \mathrm{B{-}SchoolGrad}$$
$$[\mathbf{EXISTS}\ 1 \colon \mathrm{TechnicalDegree}]]]]$$

$$\mathrm{HighTechCo} \equiv$$
$$[\mathbf{AND}\ \mathrm{Company}\ [\mathbf{FILLS} : \mathrm{Exchange\ nasdaq}]\ [\mathbf{ALL} : \mathrm{Manager\ Techie}]]$$

$$\mathrm{Techie} \equiv$$
$$[\mathbf{EXISTS}\ 2 : \mathrm{TechnicalDegree}]$$

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?

$$
\begin{aligned}
&\text{CoolTecCo} \equiv \\
&\qquad [\mathbf{AND}\ \text{Company} \\
&\qquad [\mathbf{ALL} : \text{Manager}\ [\mathbf{AND}\ \text{B}-\text{SchoolGrad}\ [\mathbf{EXISTS}\ 2 : \text{TechnicalDegree}]]] \\
&\qquad [\mathbf{FILLS} : \text{Exchange nasdaq}]]
\end{aligned}
$$

$\uparrow$ **No**

$$
\begin{aligned}
&\text{WellRoundedCo} \equiv \\
&\qquad [\mathbf{AND}\ \text{Company}\ [\mathbf{ALL} : \text{Manager}\ [\mathbf{AND} : \text{B}-\text{SchoolGrad} \\
&\qquad\qquad\qquad\qquad [\mathbf{EXISTS}\ 1 : \text{TechnicalDegree}]]]]
\end{aligned}
$$

$$
\begin{aligned}
&\text{HighTechCo} \equiv \\
&\qquad [\mathbf{AND}\ \text{Company}\ [\mathbf{FILLS} : \text{Exchange nasdaq}\ [\mathbf{ALL} : \text{Manager Techie}]]
\end{aligned}
$$

$$\text{Techie} \equiv [\mathbf{EXISTS}\ 2 : \text{TechnicalDegree}]$$

- How about the **intersection** of WellRoundedCo and HighTechCo?

Join and Expand Definitions

WellRoundedCo ≡
[**AND** Company [**ALL** : Manager [**AND** : B−SchoolGrad
[**EXISTS** 1 : TechnicalDegree]]]]

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

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

We expand the definitions of WellRoundedCo and HighTechCo, and then Techie yielding:

[**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

$$
\begin{array}{l}
[\textbf{AND}\ [\textbf{AND}\ \text{Company} \\
\qquad\qquad [\textbf{ALL} : \text{Manager}\ [\textbf{AND}\ \text{B}{-}\text{SchoolGrad} \\
\qquad\qquad\qquad\qquad [\textbf{EXISTS}\ 1 : \text{TechnicalDegree}]]]] \\
\qquad\quad [\textbf{AND}\ \text{Company} \\
\qquad\qquad [\textbf{FILLS} : \text{Exchange nasdaq}] \\
\qquad\qquad [\textbf{ALL} : \text{Manager}\ [\textbf{EXISTS}\ 2 : \text{TechnicalDegree}]]]]
\end{array}
$$

We flatten the **AND** operators at the top level and then combine the **ALL** operators over :Manager:

$$
\begin{array}{l}
[\textbf{AND}\ \text{Company} \\
\qquad\qquad [\textbf{ALL} : \text{Manager}\ [\textbf{AND}\ \text{B}{-}\text{SchoolGrad} \\
\qquad\qquad\qquad\qquad [\textbf{EXISTS}\ 1 : \text{TechnicalDegree}] \\
\qquad\qquad\qquad\qquad [\textbf{EXISTS}\ 2 : \text{TechnicalDegree}]]] \\
\qquad\quad \text{Company} \\
\qquad\qquad [\textbf{FILLS} : \text{Exchange nasdaq}]
\end{array}
$$

Remove Redundant Concepts and Combine Operators

$$
\begin{array}{l}
[\textbf{AND}\ \text{Company} \\
\qquad [\textbf{ALL} : \text{Manager}\ [\textbf{AND}\ \text{B}{-}\text{SchoolGrad} \\
\qquad\qquad [\textbf{EXISTS}\ 1 : \text{TechnicalDegree}] \\
\qquad\qquad [\textbf{EXISTS}\ 2 : \text{TechnicalDegree}]]] \\
\quad \text{Company} \\
\qquad [\textbf{FILLS} : \text{Exchange nasdaq}]
\end{array}
$$

We remove the redundant Company concept and combine the **EXISTS** operators over :TechnicalDegree, yielding:

$$
\begin{array}{l}
\text{WellRoundedCo} \sqcap \text{HighTechCo} \equiv \\
\quad [\textbf{AND}\ \text{Company} \\
\qquad [\textbf{ALL} : \text{Manager}\ [\textbf{AND}\ \text{B}{-}\text{SchoolGrad}\ [\textbf{EXISTS}\ 2 : \text{TechnicalDegree}]]] \\
\qquad [\textbf{FILLS} : \text{Exchange nasdaq}]]
\end{array}
$$

$$
\begin{array}{l}
\text{CoolTecCo} \equiv \\
\quad [\textbf{AND}\ \text{Company} \\
\qquad [\textbf{ALL} : \text{Manager}\ [\textbf{AND}\ \text{B}{-}\text{SchoolGrad}\ [\textbf{EXISTS}\ 2 : \text{TechnicalDegree}]]] \\
\qquad [\textbf{FILLS} : \text{Exchange nasdaq}]]
\end{array}
$$

Another Example: Applying DL-Specific Structural Rules

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

?

$\boldsymbol{D \sqsubseteq E}$**?**

minCardinality vs. hasValue ?

$D \equiv$ [**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

- $\boldsymbol{D}$ **subsumes** $\boldsymbol{C}$ **if and only if** $\boldsymbol{C}$ **logically implies** $\boldsymbol{D}$

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

$$KB \models C \rightarrow D$$

$$KB \models \neg C \vee D$$

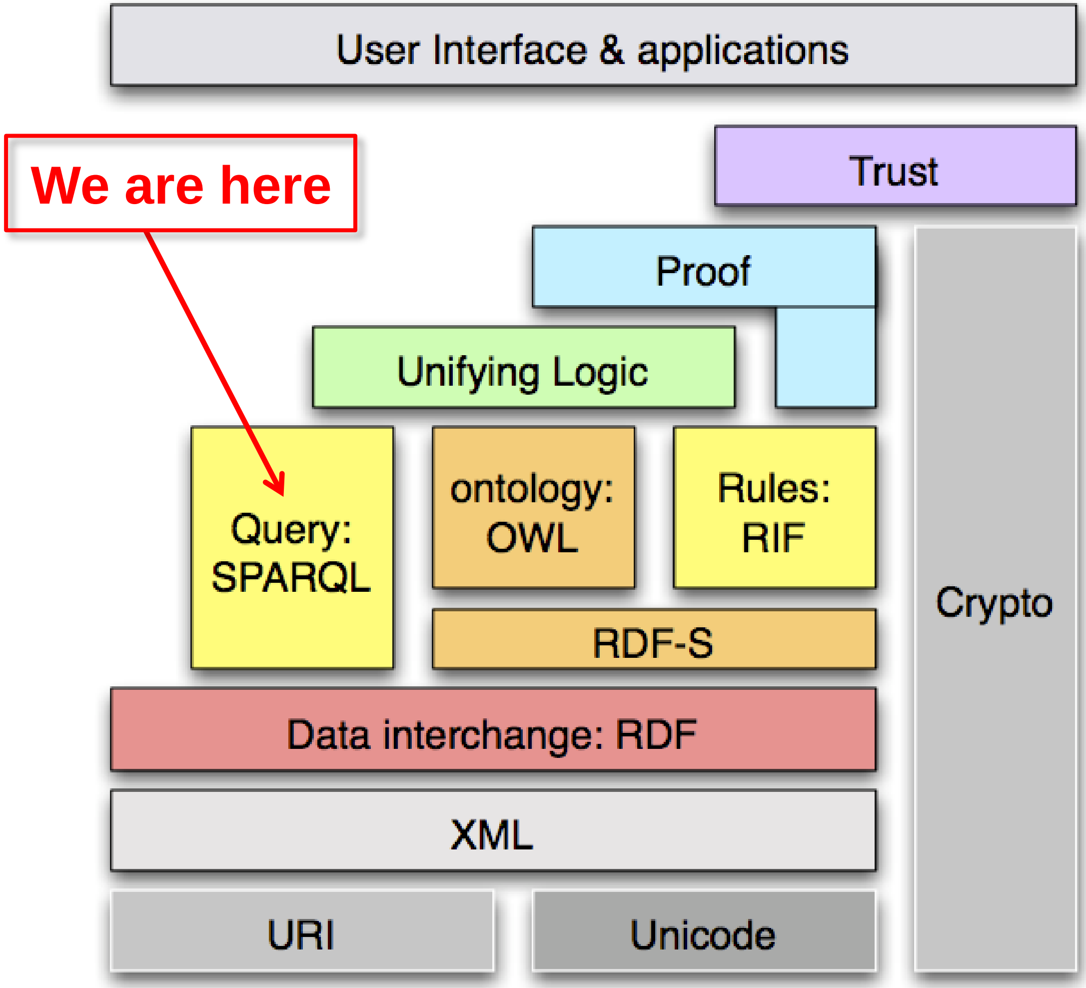
$$KB \not\models \neg(\neg C \vee D)$$

$$KB \wedge C \wedge \neg D \models \boldsymbol{F}$$

Two options:

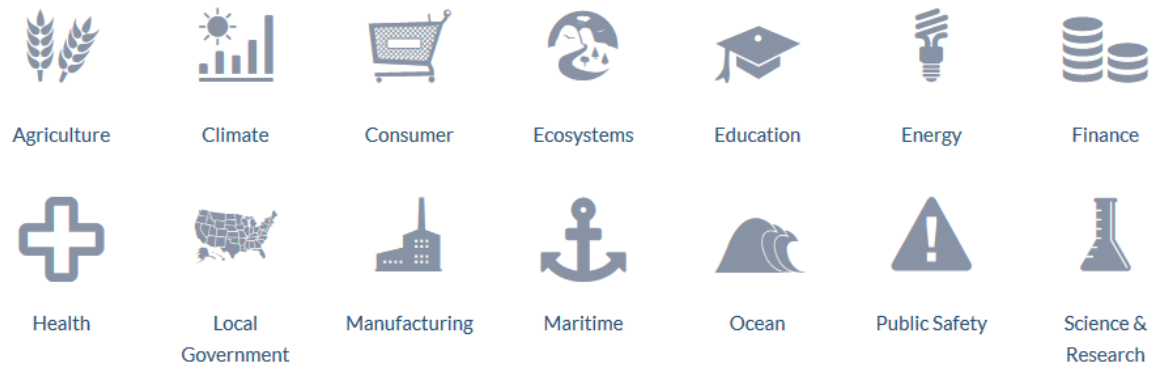
- Use a Tableau theorem prover to construct a satisfying instance
- Use a SAT Checker to prove unsatisfiability

Querying the Semantic Web with SPARQL

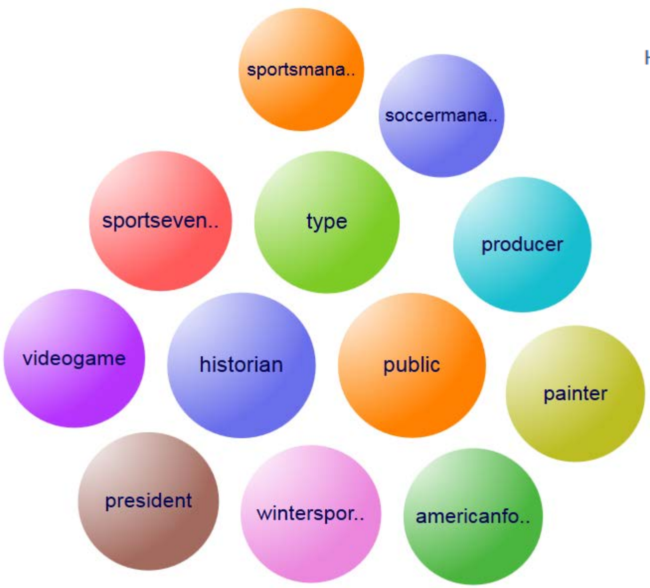


RDF Stores on the Web

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

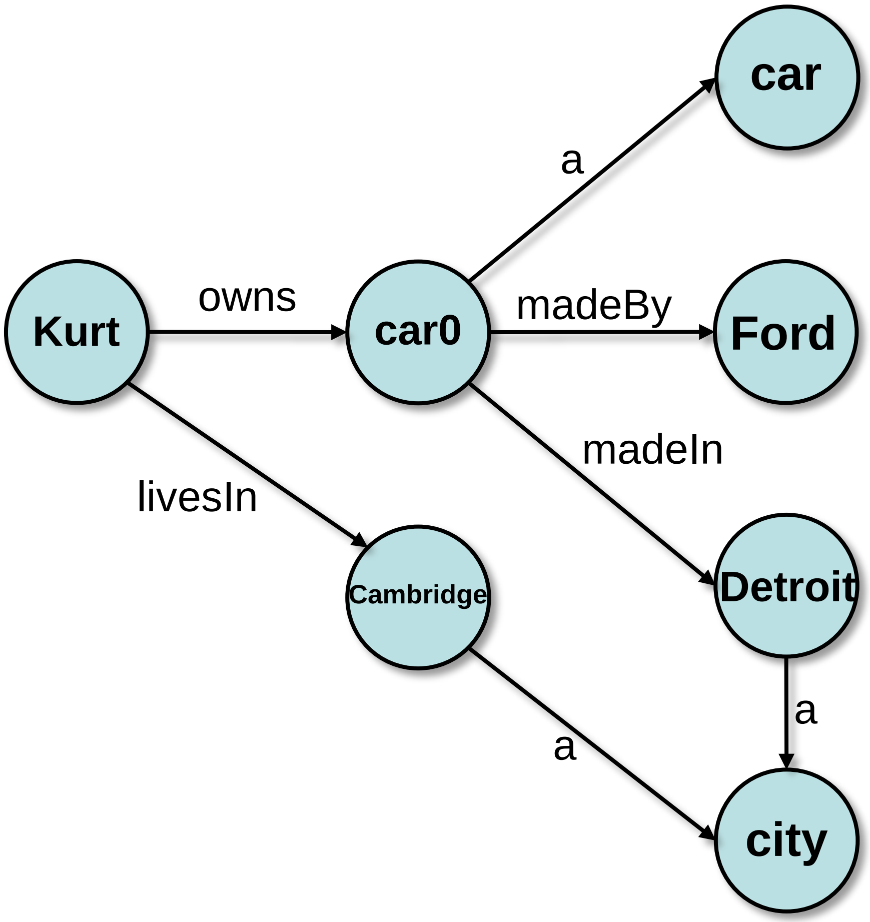


data.gov



DBpedia Bubble Navigator

RDF

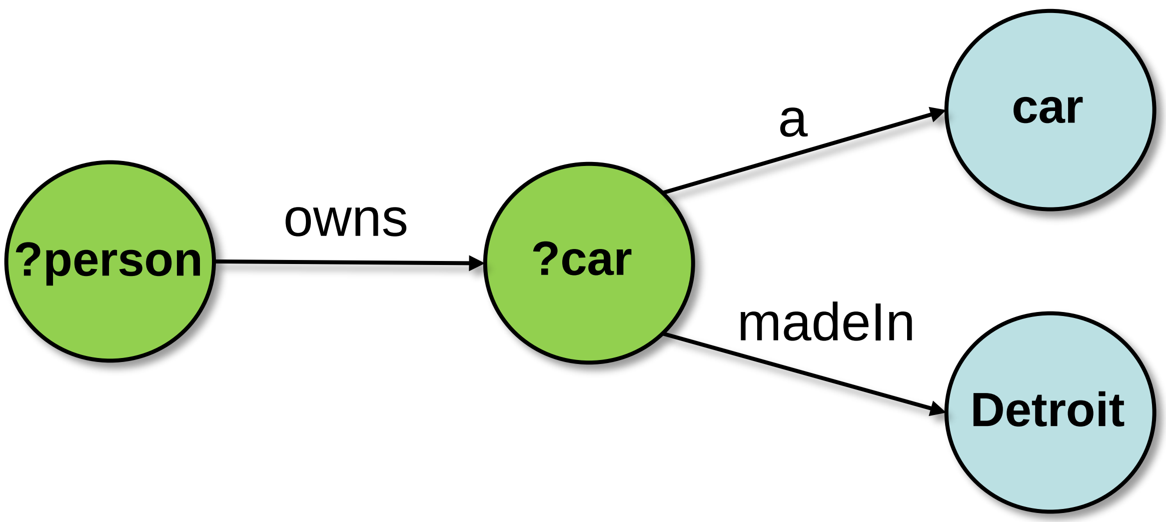


7 triples (in the ABox)

- 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

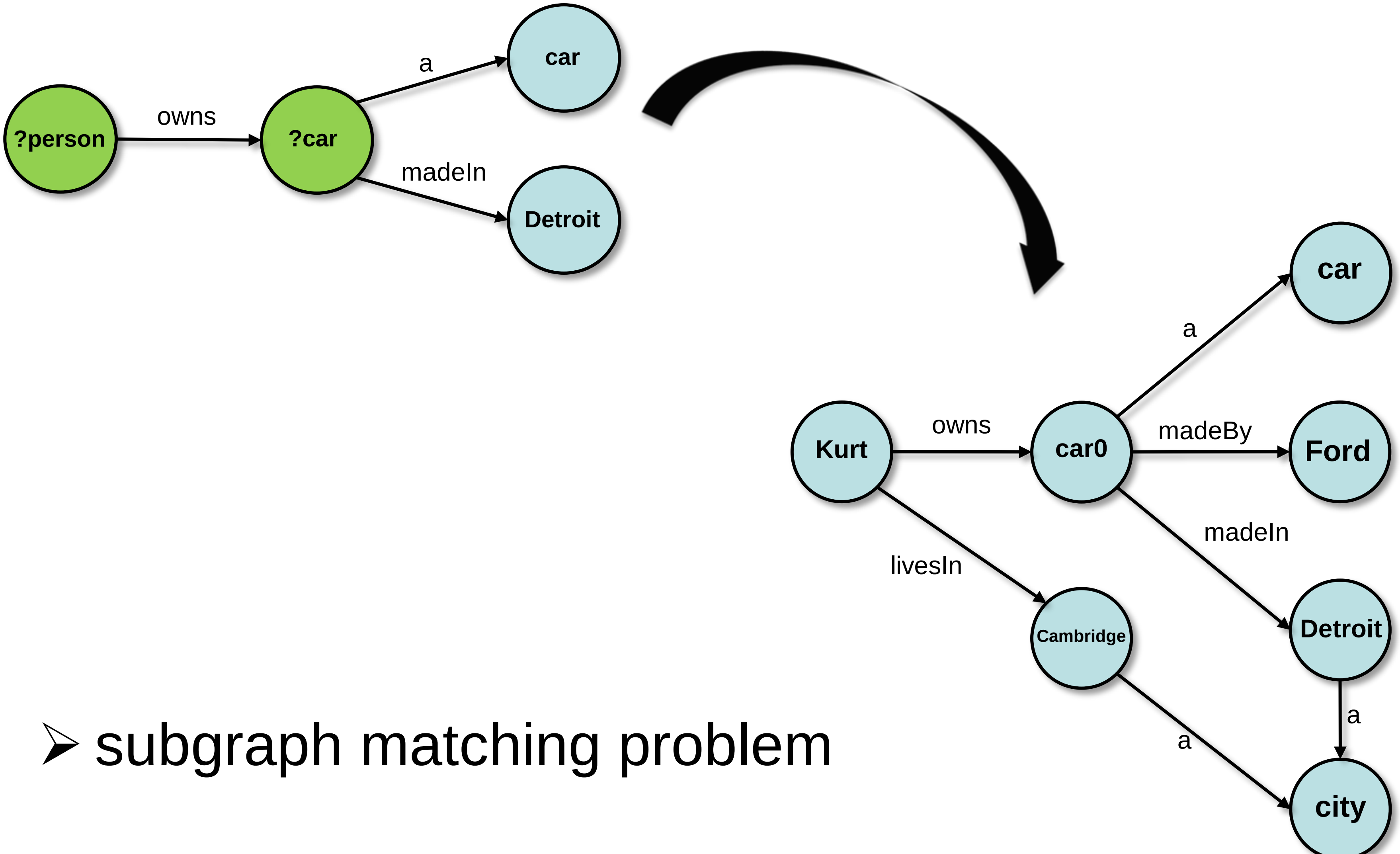
Query an RDF Store - SPARQL

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 $f: V_0 \rightarrow V'$ such that $(v_1, v_2) \in E_0 \Leftrightarrow \big(f(v_1), f(v_2)\big) \in E'$?

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

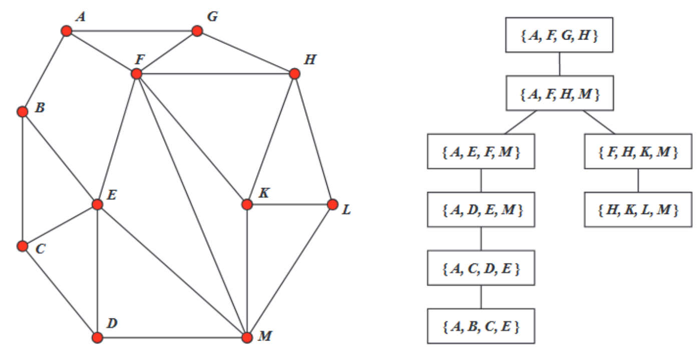
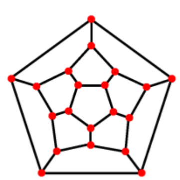
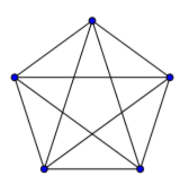


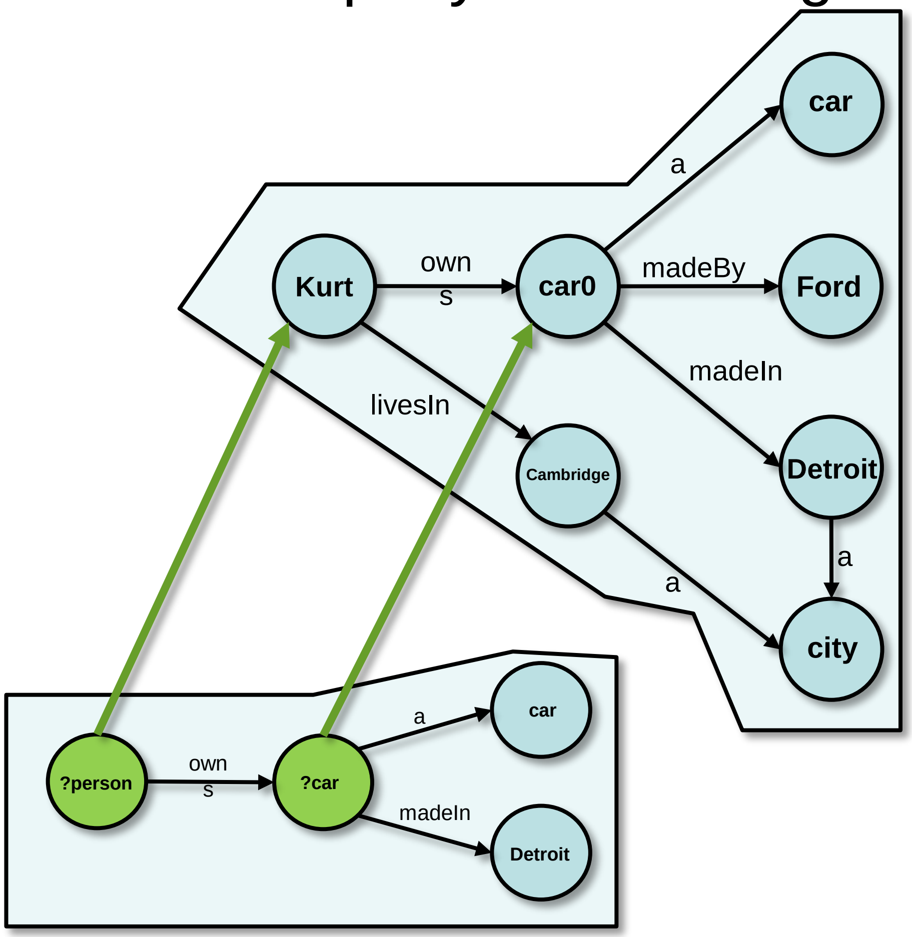
Figure 1: Tree decomposition of a planar graph.





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 …
```

Query result clause

Define the dataset (*optional*)

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

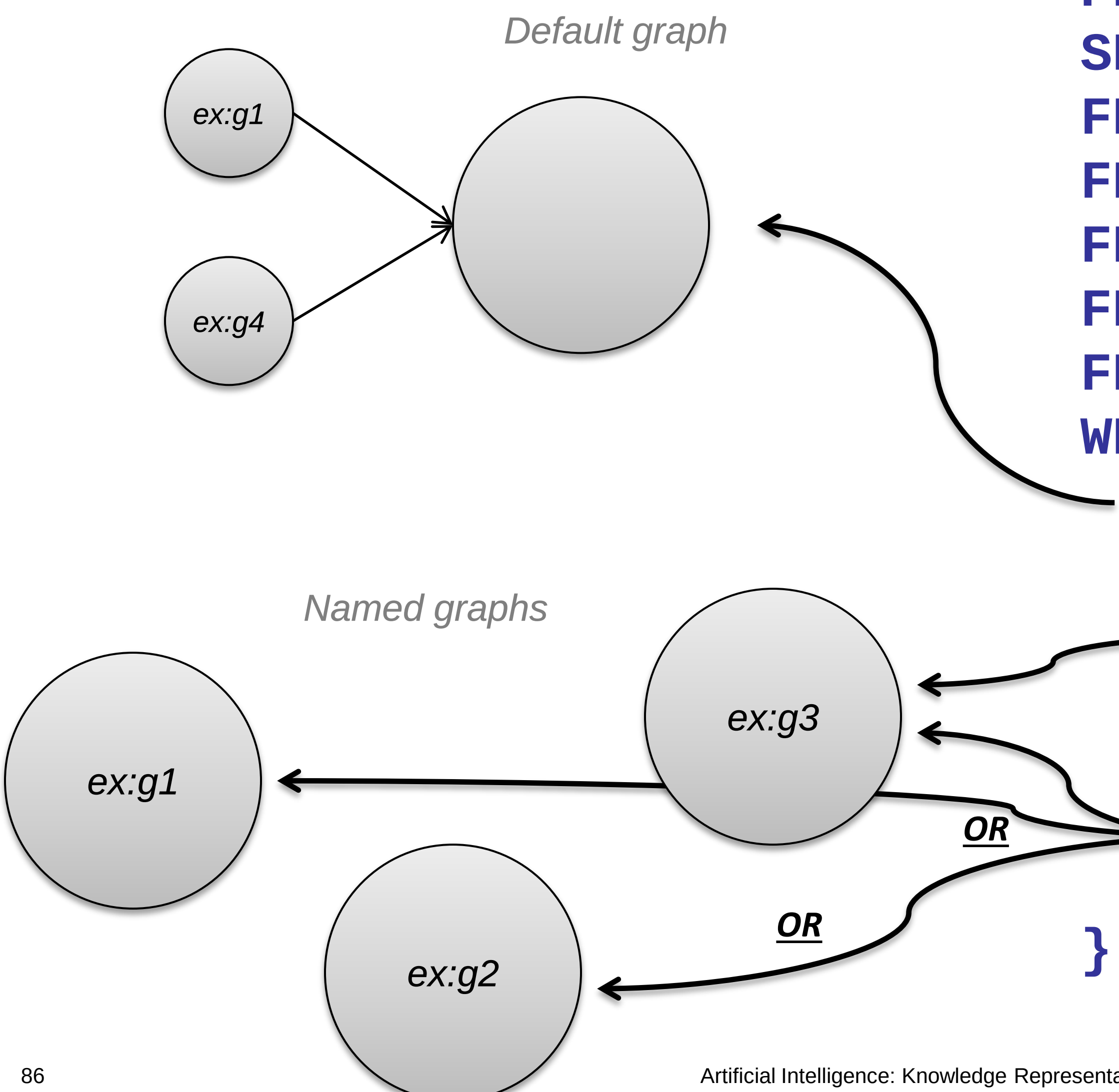
`{ A } UNION { B }` ⇨ **Disjunction**

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 …
   }
}
```

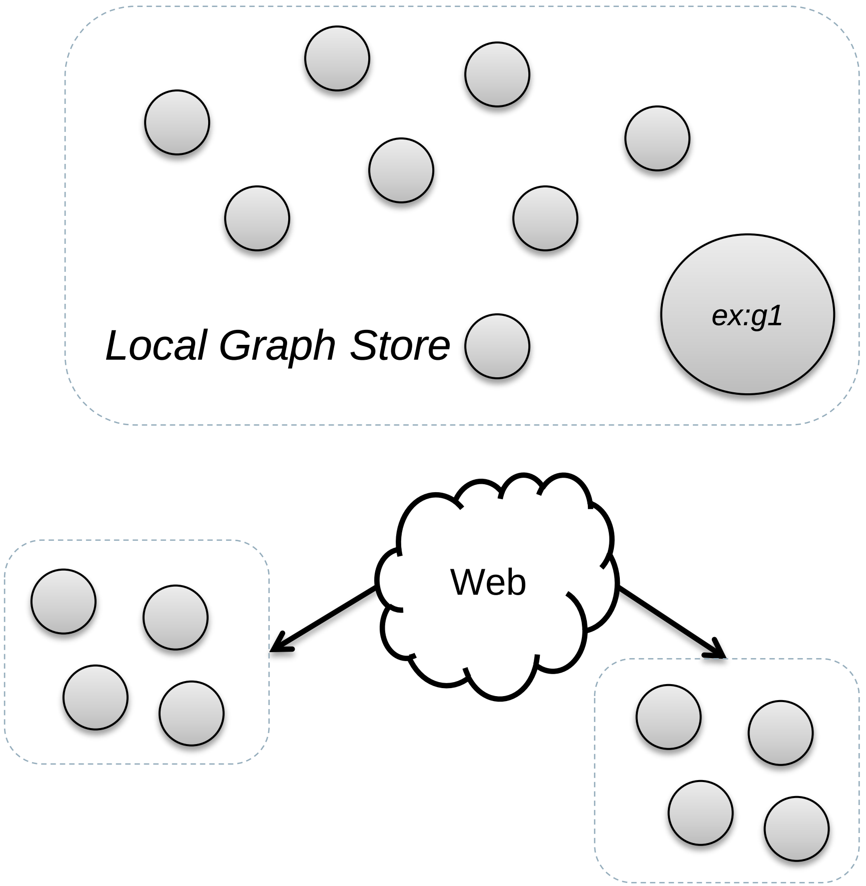
Distributing Queries over Multiple Triplestores

1. Query a local collection of stores
   - Build local store with copies of relevant external stores

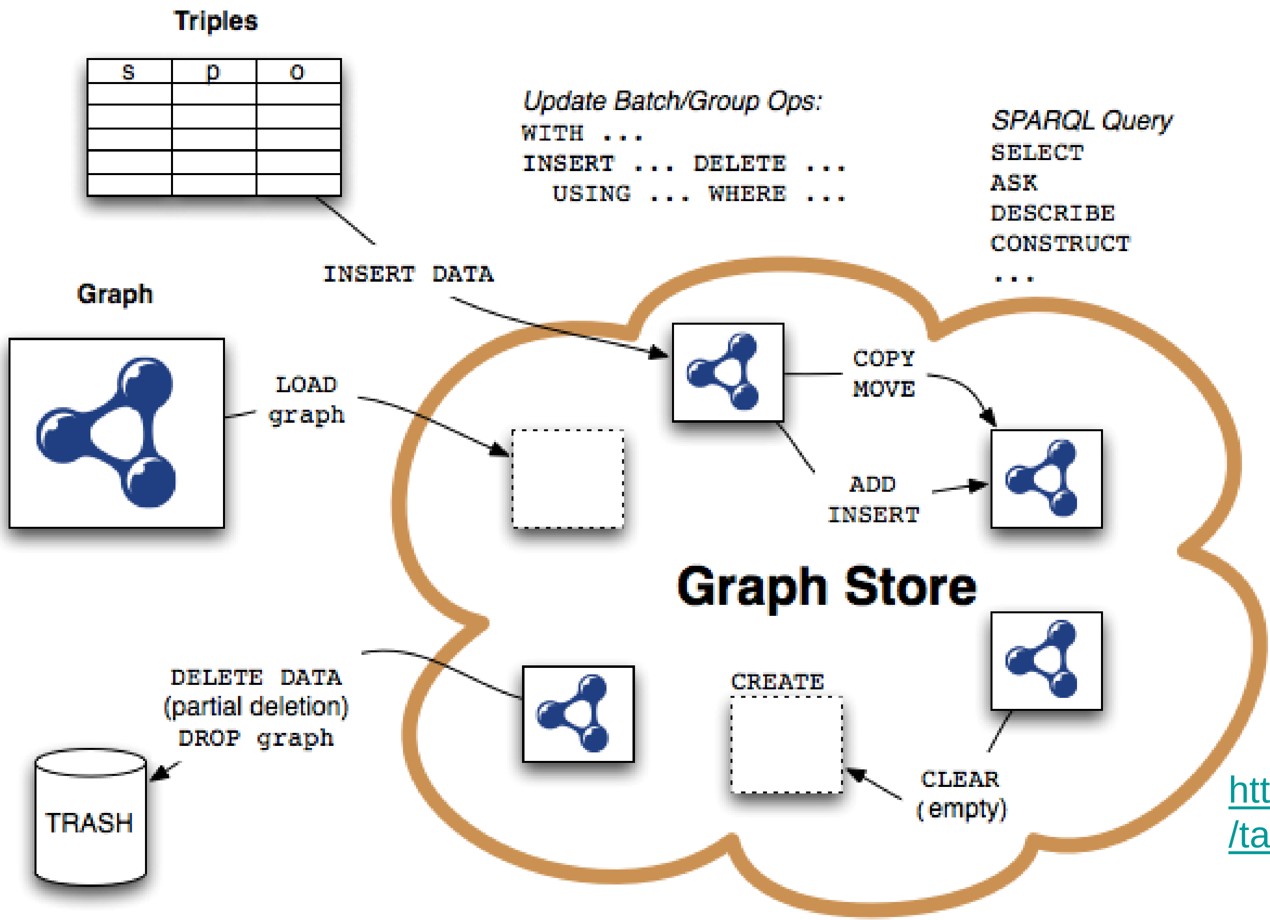
2. Issue follow-up queries to external stores

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

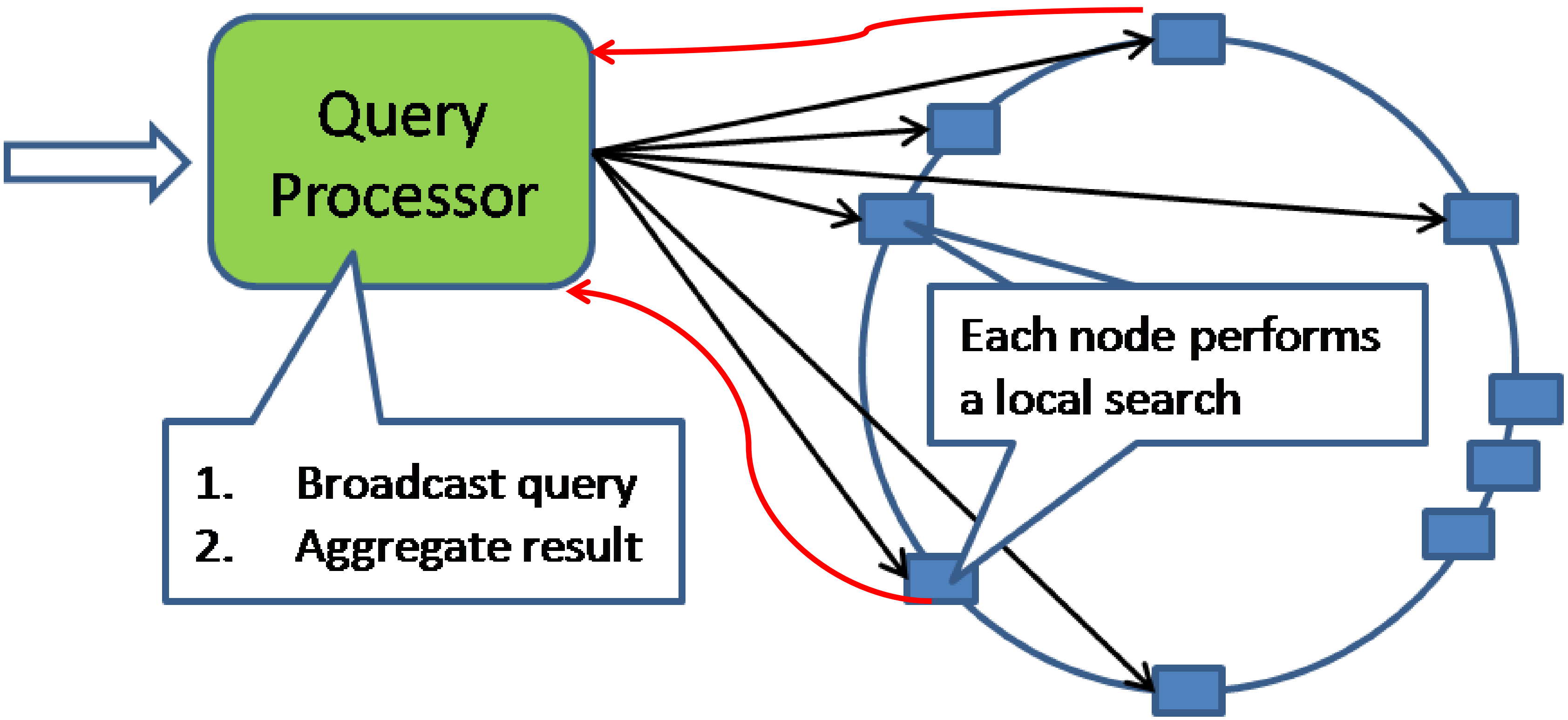
- Take results from a first query to substitute placeholders in subsequent query templates

➢ 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

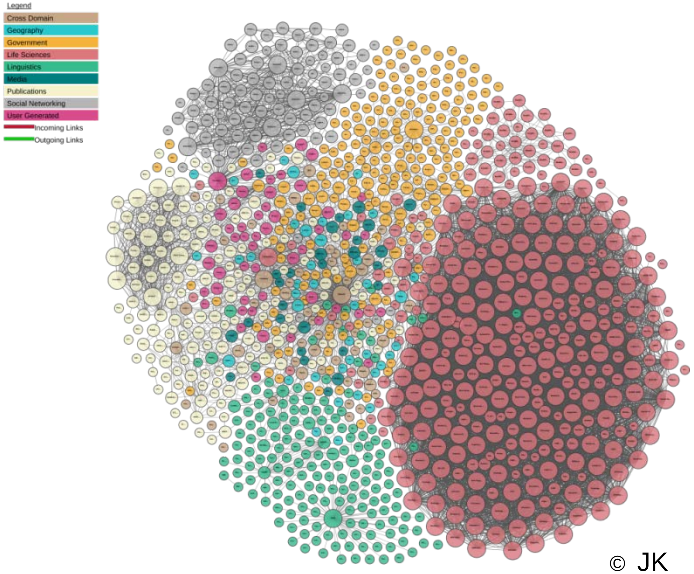
(4) Automatic Link Traversal

- Traverse RDF links during query evaluation
- Link-Traversal based query execution (LTBQE)

Subject: http://dig.csail.mit.edu/data#DIG
Predicate: [http://xmlns.com/foaf/0.1/ member](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

Summary

- Description logics are widely accepted formalisms to represent conceptual knowledge (ontologies)
- We distinguish between abstract concepts in the TBox (terminological knowledge) and concrete instances in the ABox (assertional knowledge)
- ALC is a well-studied decidable fragment of first-order logic and a basis for many description logics
- Subsumption, classification and instance relationships are essential inference services needed for ontology data bases
- Modern knowledge graphs use RDF to represent huge sets of subject-predicate-object triples
- Sparql is a querying language for RDF stores

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 simple 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?