Knowledge Graphs and Ontologies

Outline

References

Books (freely available online):

- The Description Logic Handbook: Theory, Implementation, and Applications. Baader et al., Cambridge University Press
- Foundations of Semantic Web Technologies. Hitzler et al., Chapman & Hall / CRC
- Web Data Management. Abiteboul et al., Cambridge University Press

W3C standards and documentation:

- RDF: `https://www.w3.org/TR/rdf11-concepts/`
- SPARQL: `https://www.w3.org/TR/rdf-sparql-query/`
- OWL: `https://www.w3.org/TR/owl2-overview`
- OWL 2 profiles: `https://www.w3.org/TR/owl2-profiles/`

References

Courses:

- Markus Krötzsch's course on Knowledge Graphs, especially the lecture on Wikidata
  `https://iccl.inf.tu-dresden.de/web/Knowledge_Graphs_(WS2019/20)/en`
- Meghyn Bienvenu's course on Description Logics
  `https://www.labri.fr/perso/meghyn/papers/intro-dls.pdf`
- Franz Baader's course on Description Logics
  `https://tu-dresden.de/ing/informatik/thi/lat/studium/lehrveranstaltungen/sommersemester-2019/description-logic`

What is a Knowledge Graph ?

"Since Google started an initiative called Knowledge Graph in 2012, a substantial amount of research has used the phrase knowledge graph as a generalized term.
Although there is no clear definition for the term knowledge graph, it is sometimes used as synonym for ontology.
One common interpretation is that a knowledge graph represents a collection of interlinked descriptions of entities – real-world objects, events, situations or abstract concepts.
Unlike ontologies, knowledge graphs, such as Google's Knowledge Graph, often contain large volumes of factual information with less formal semantics.
In some contexts, the term knowledge graph is used to refer to any knowledge base that is represented as a graph."
(Wikipedia, 28 Oct. 2019)

Examples of Knowledge Graphs

https://www.wikidata.org/wiki/Q937

https://tinyurl.com/2p9cnsta

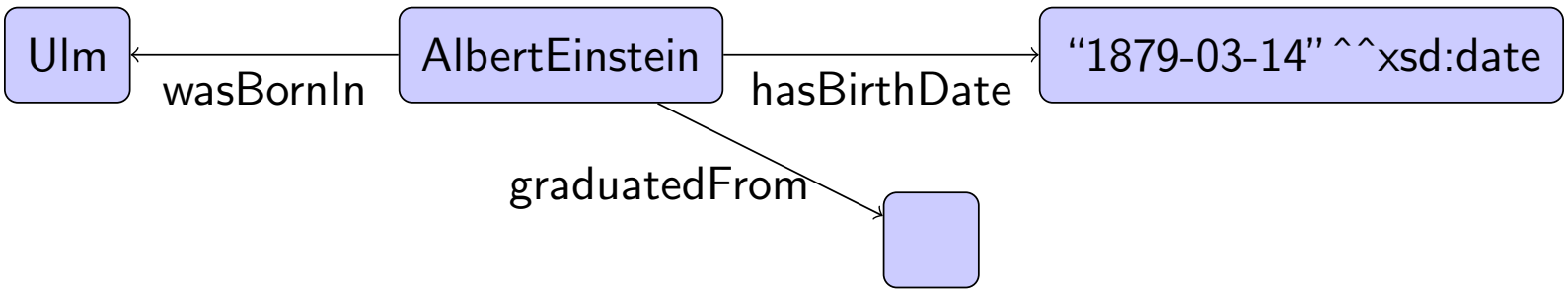
http://fr.dbpedia.org/page/Albert_Einstein

Google Knowledge Graph, Microsoft Bing Satori
https://tinyurl.com/y53nzrqv

Reminders: RDF (Resource Description Framework)

RDF graph : set of triples of the form (subject, predicate, object)
- subject : IRI (Internationalized Resource Identifier) or blank node
- predicate : IRI
- object : IRI, blank node, or literal

- W3C standard for exchanging graphs
- Directed labelled (multi-) graphs
- Nodes are entities (vertices labelled with IRIs), data values (vertices labelled with literals), or blank nodes (vertices without labels)



Reminders: SPARQL (SPARQL Protocol and RDF Query Language)

W3C standard query (and update) language for RDF data
We focus on SELECT queries

- PREFIX declaration: specifies namespaces
- SELECT clause: output variables (strings that begin with ?)
- WHERE clause:
  - basic graph patterns (BGP): sets of triple patterns: $\langle s, p, o \rangle$ where s and o are RDF terms or variables and p is an IRI or variable, written as a whitespace-separated list in the query
  - possibly property path patterns instead of triple patterns: p can be a property path $\sim$ regular expression built on IRIs
  - possibly FILTER, UNION, OPTIONAL
- Solution set modifiers (DISTINCT, LIMIT, ORDER BY...)

SPARQL Examples

SELECT Query

```
PREFIX dc10:  <http://purl.org/dc/elements/1.0/>
PREFIX dc11:  <http://purl.org/dc/elements/1.1/>

SELECT ?title ?author
   WHERE  {
     { ?book dc10:title ?title .
       ?book dc10:creator ?author }
     UNION
     { ?book dc11:title ?title .
       ?book dc11:creator ?author }
   }
```

SPARQL Examples

Property paths

Some property paths constructors

path1/path2	path1 followed by path 2
^path1	backwards path (object to subject)
path1|path2	path1 or path2
path1*	path1 repeated zero or more times
path1+	path1 repeated one or more times

SPARQL Examples

Property paths

```
{
  ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows/foaf:name ?name .
 }

{
  ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows+/foaf:name ?name .
 }


 { ?x rdf:type/rdfs:subClassOf* ?type }
```

Answers to SPARQL Queries

Evaluating a query pattern P over an RDF graph G generates an unordered collection of solutions, each solution being a function μ from variables of P to RDF terms such that there is a mapping σ from blank nodes to RDF terms such that $\mu(\sigma(P)) \subseteq G$.

Using Knowledge Graphs to Answer Complex Questions

Which European citizens were married to Zsa Zsa Gabor?

`https://tinyurl.com/ybp9ljyd`
`https://www.wikidata.org/wiki/Q207405`
`https://tinyurl.com/y9c9f8mo`

Which family members of the president of America were born outside of America?

In which Asian restaurants can you eat vegetarian food in Paris?

What are the inhibitors of enzymes produced by genes on the Y chromosome?

Hands-on Session: Querying Wikidata

Wikidata

- Free knowledge base that anyone can edit
- Wikipedia's knowledge graph
- Large graph: >1.2 billions statements on >90M entities on Feb. 2021
- Large, active community (thousands of active contributors)
- Launched in 2012
- Many applications
  - Wikipedia: inter-language links, auto-generated info boxes, article placeholders...
  - Application-specific data-excerpts
  - Data integration and quality control
  - ...

Hands-on Session: Querying Wikidata

Principles of Wikidata

- Open editing: Anyone can extend or modify content;
- Community control: The users decide what is stored and how it is represented;
- Plurality: There might not be one truth but several co-existing views; such complexity must be supported;
- Secondary data: All content should be supported by external, primary sources;
- Multi-lingual data: One site serves all languages; labels are translated: content is the same for all;
- Easy access: Technical and legal barriers for data re-use are minimized;
- Continuous evolution: Incompleteness of content and technology are embraced; Wikidata remains work in progress.

Hands-on Session: Querying Wikidata

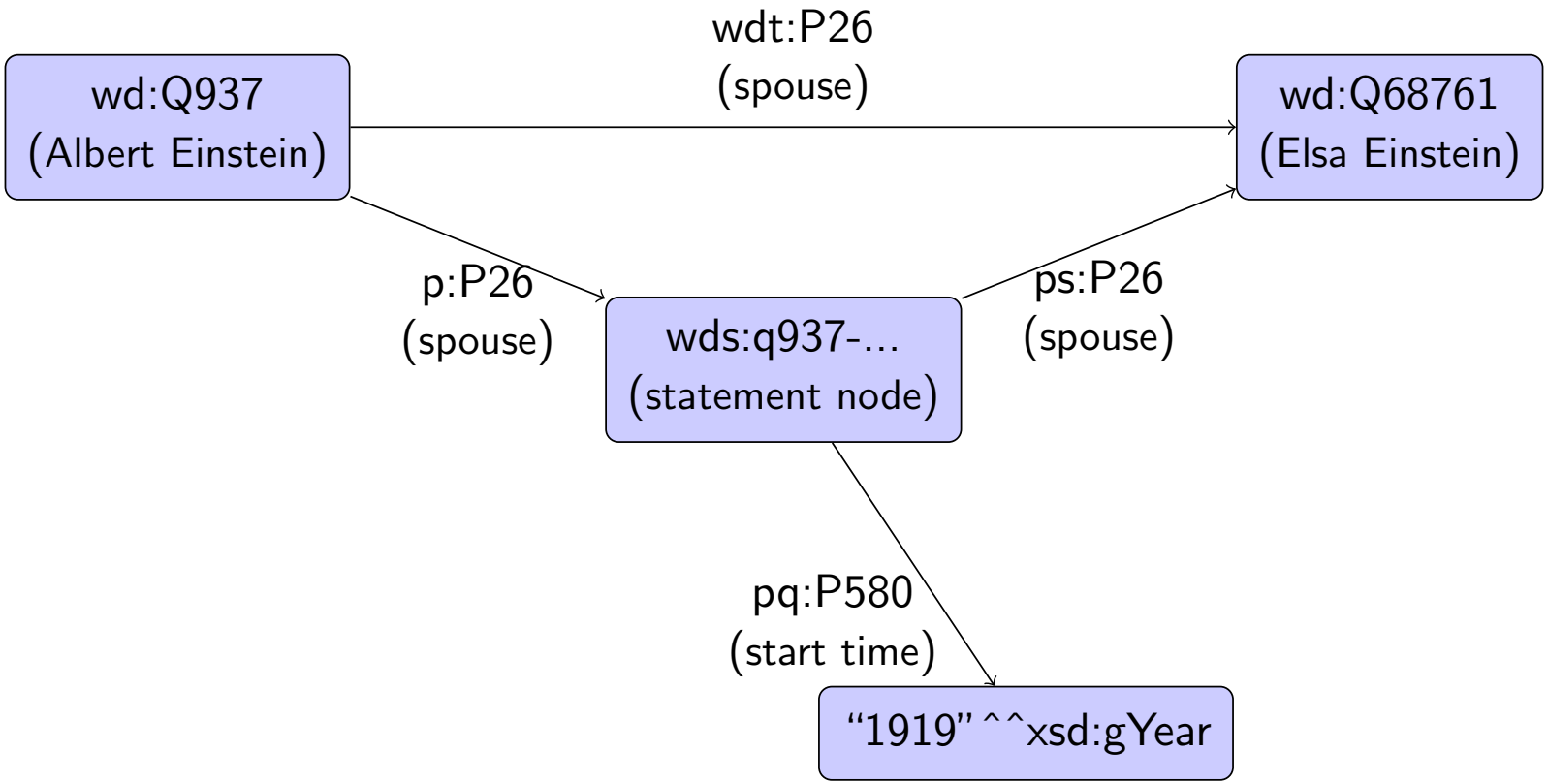
Wikidata data (simplified)

- Statements: Wikidata's basic information units, sourced claims for several properties that an entity might have
  - Built from Wikidata items("Albert Einstein"), Wikidata properties ("date of birth", "spouse"), and data values ("1879")
  - Items and properties can be subjects/values in statements
  - Annotated with property-value pairs ("start time: 1919")
- Entities identified by language-independent ids, starting by Q for items and P for properties (e.g. Q937, P40)
- Wikidata is internally stored in the document-centric form using a JSON format but is converted in RDF for several purposes, and in particular for export for external use and for importing data into Wikidata's SPARQL query service

Hands-on Session: Querying Wikidata

RDF encoding of Wikidata statements

We present the basics needed for today's hands-on session.



Hands-on Session: Querying Wikidata

RDF encoding of Wikidata statements

Summary

- Each statement is represented by a resource in RDF ("wds:q937-881C4FA7-075C-4D48-8182-77D69CA6309C")
- Direct single-triple links from subject to value are added ("wd:Q937 wdt:P26 wd:Q68761")
- Each Wikidata property turns into several RDF properties for different uses in encoding ("wdt:P26", "wd:P26"...)
- Order of qualifiers or statements is not represented in RDF

The complete Wikidata-to-RDF documentation is available online:
`https://www.mediawiki.org/wiki/Wikibase/Indexing/RDF_Dump_Format`

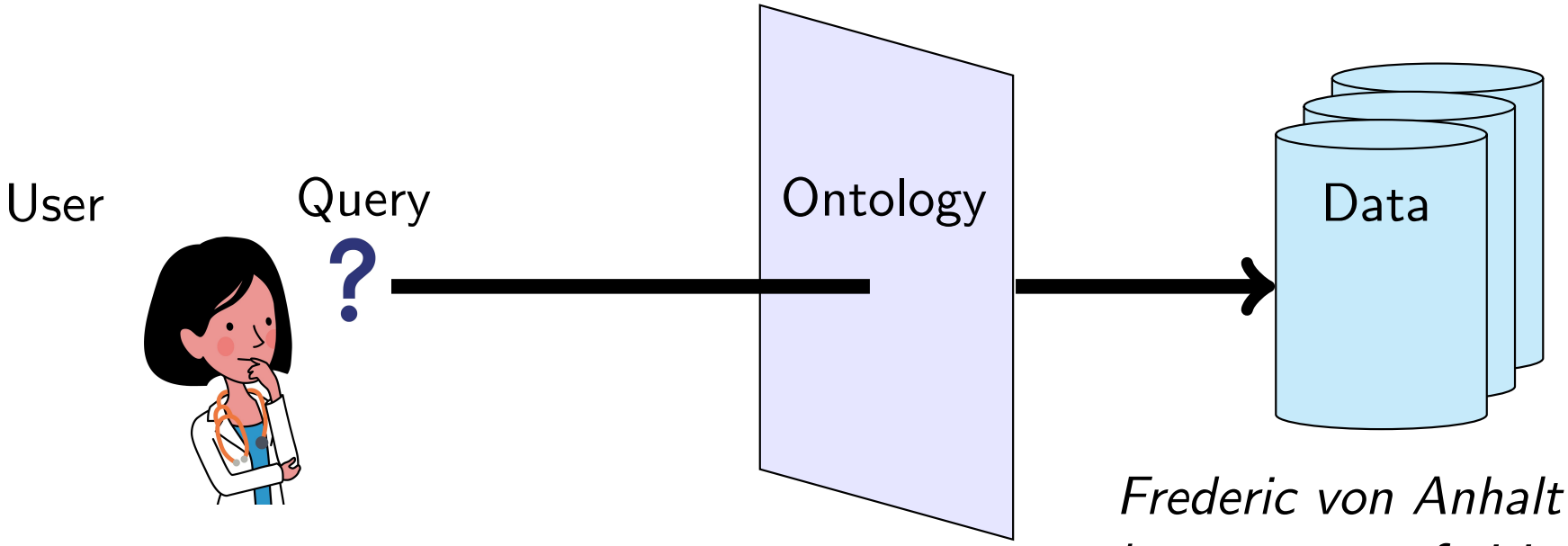
Hands-on Session: Querying Wikidata

- Wikidata: `https://www.wikidata.org/wiki/Wikidata:Main_Page`
- Wikidata Query Service: `https://query.wikidata.org/`
- Queries:
  - List of Albert Einstein's children with their birth date and place.
  - Subproperties of the property student.
  - List of students of Einstein, of the students of his students, etc.
  - List of singers (occupation singer) having French and German citizenship.
  - List of singers having French or German citizenship.
  - List of paintings from European painters that are located in France.
  - List of French presidents with the start date of their presidency.
  - List of presidents of the French Fifth Republic with the start date of their presidency.
  - Number of presidents of the French Fifth Republic.
  - List of proteins encoded by some gene located on chromosome Y.

Accessing Data through an Ontology

- Knowledge graphs and SPARQL queries allow us to get answers to complex queries
- But SPARQL queries may become very (too) complex
- Need for a way of formulating simpler queries, closer to the natural language of the user, and still get all the answers from the data
- Ontologies allow to formalize knowledge and delegate the reasoning to the machine

Accessing Data through an Ontology



Which European citizens were married to Zsa Zsa Gabor?

Someone with country of citizenship in Europe is a European citizen

Frederic von Anhalt has country of citizenship Germany

Germany is in Europe

Frederic von Anhalt was married to Zsa Zsa Gabor

Accessing Data through an Ontology

```
SELECT DISTINCT ?spouse WHERE
{   ZsaZsaGabor hasSpouse ?spouse.
    ?spouse hasCountryOfCitizenship ?country.
    ?country hasContinent Europe. }
```

Express relationships : $\mathit{hasContinent}(x, \mathit{Europe}) \implies \mathit{EuropeanCountry}(x)$

$\mathit{hasCountryOfCitizenship}(x, y) \wedge \mathit{EuropeanCountry}(y) \implies \mathit{EuropeanCitizen}(x)$

as an OWL (Web Ontology Language) ontology

```
EuropeanCountry rdfs:subClassOf owl:Restriction
EuropeanCountry owl:onProperty hasContinent
EuropeanCountry owl:ObjectHasValue Europe
EuropeanCitizen rdfs:subClassOf owl:Restriction
EuropeanCitizen owl:onProperty hasCountryOfCitizenship
EuropeanCitizen owl:someValuesFrom EuropeanCountry
```

```
SELECT DISTINCT ?spouse WHERE
{   ZsaZsaGabor hasSpouse ?spouse.
    ?spouse rdf:type EuropeanCitizen. }
```

Ontologies

An ontology is a formal conceptualization of a domain of interest. Ontologies can be seen as logical theories, thereby making knowledge available for machine processing.

Ontologies

An ontology is a formal conceptualization of a domain of interest. Ontologies can be seen as logical theories, thereby making knowledge available for machine processing.

An ontology defines the terminology (vocabulary) of the domain and the semantics relationships between terms.

Example (family domain)

- Terms: parent, mother, sister, sibling, ...
- Relationships between terms: "mother" is a subclass of "parent", "sister" is both in the domain and in the range of "has sibling", "parent" is the disjoint union of "father" and "mother"...

Reasons for Using Ontologies

- Standardize the terminology of an application domain : make it easy to share information – well-defined syntax and formal logic-based semantics (i.e. meaning)
  - complex industrial systems description, scientific knowledge (medicine, life science...)

Reasons for Using Ontologies

- Standardize the terminology of an application domain : make it easy to share information – well-defined syntax and formal logic-based semantics (i.e. meaning)
  - complex industrial systems description, scientific knowledge (medicine, life science...)
- Present an intuitive and unified view of data sources: make it easy to formulate queries
  - data integration, semantic web

Reasons for Using Ontologies

- Standardize the terminology of an application domain : make it easy to share information – well-defined syntax and formal logic-based semantics (i.e. meaning)
  - complex industrial systems description, scientific knowledge (medicine, life science...)
- Present an intuitive and unified view of data sources: make it easy to formulate queries
  - data integration, semantic web
- Support automated reasoning: logical inferences allow us to take advantage of implicit knowledge to answer queries – computational aspects can be studied to design ontology languages and tools that allow for efficient reasoning
  - expert systems, semantic web, ontology-based data access

Ontology Languages

- You already heard about RDFS (RDF Schema) and OWL (Web Ontology Language), two W3C standards for RDF data and the Semantic Web
  - RDFS has low expressivity
  - OWL Full is undecidable
- Ontology languages design: trade-off between expressive power and complexity of reasoning
- The formal basis of OWL is first-order logic (FOL).
  We will study interesting decidable fragments of FOL
  - Description Logics (integrated in OWL 2 profiles)
  - Existential Rules

Description Logics: Syntax

Basic building blocks

- atomic concepts (unary predicates)
  - Mother, Sister ...
- atomic roles (binary predicates)
  - hasChild, isMarriedTo ...
- individuals (constants)
  - *alice*, *bob* ...

Description Logics: Syntax

Basic building blocks

- atomic concepts (unary predicates)
  - Mother, Sister ...
- atomic roles (binary predicates)
  - hasChild, isMarriedTo ...
- individuals (constants)
  - *alice*, *bob* ...

Complex concepts

- concept constructors: $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists R.C$ ...
  - Mother $\sqcup$ Father : "mothers or fathers"
  - Mother $\sqcap \neg \exists$hasChild.Male : "mothers who don't have any male child"

Complex roles

- role constructors: R^- (inverse), RoS (composition) ...

Description Logics: Syntax

DL knowledge base = TBox (ontology) + ABox (data)

TBox (terminological box) specifies knowledge at intensional level

- describes general knowledge about the domain
- defines a set of conceptual elements (concepts, roles) and states constraints describing the relationships between them

ABox (assertional box) specifies knowledge at extensional level

- contains facts about specific individuals
- specifies a set of instances of the conceptual elements described at the intensional level

Note: the term ontology is sometimes used to refer to the whole knowledge base rather than to the TBox alone.

Description Logics: Syntax

The TBox contains concept inclusions, role inclusions and possibly properties about roles (transitivity, functionality...).

- Mother $\sqsubseteq$ Parent : "all mothers are parents"
- Spouse $\sqsubseteq \exists$isMarriedTo : "spouses are married to someone"
- hasParent $\sqsubseteq$ hasChild$^-$: "if x has parent y, then y has child x"

Description Logics: Syntax

The TBox contains concept inclusions, role inclusions and possibly properties about roles (transitivity, functionality...).

- Mother $\sqsubseteq$ Parent : "all mothers are parents"
- Spouse $\sqsubseteq \exists$isMarriedTo : "spouses are married to someone"
- hasParent $\sqsubseteq$ hasChild$^-$: "if x has parent y, then y has child x"

The ABox contains concept assertions and role assertions.

- Mother(*alice*) : "alice is a mother"
- hasParent(*bob*, *alice*) : "bob has parent alice"

Description Logics: Semantics

- Declarative, model-theoretic semantics:
  - maps symbolic representations to entities of an abstraction of the real-world (interpretation)
  - notion of truth that allows us to determine whether a symbolic expression is true in the world under consideration (model)
- Not procedural semantics: not defined by how certain algorithms behave
- Results depend only on the semantics, not on the syntactic representation: semantically equivalent knowledge bases lead to the same results

Description Logics: Semantics

Interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$

- $\Delta^{\mathcal{I}}$ is a non-empty set called domain
- $\cdot^{\mathcal{I}}$ is a function which associates
  - each constant a with an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
  - each atomic concept A with a unary relation $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
  - each atomic role R with a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

Description Logics: Semantics

Interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$

- $\Delta^{\mathcal{I}}$ is a non-empty set called domain
- $\cdot^{\mathcal{I}}$ is a function which associates
  - each constant a with an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
  - each atomic concept A with a unary relation $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
  - each atomic role R with a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

Example:

$\Delta^{\mathcal{I}} = \{a, b, c, d, e, f, g\}$

$alice^{\mathcal{I}} = a$, $bob^{\mathcal{I}} = b$

$\text{Mother}^{\mathcal{I}} = \{a, c\}$

$\text{Father}^{\mathcal{I}} = \{b, e\}$

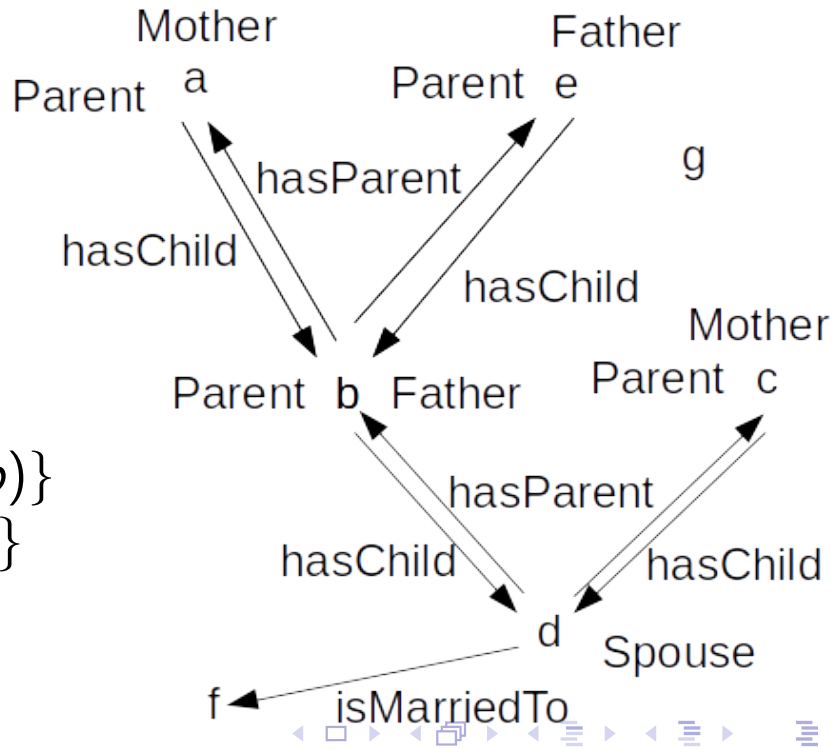
$\text{Parent}^{\mathcal{I}} = \{a, b, c, e\}$

$\text{Spouse}^{\mathcal{I}} = \{d\}$

$\text{hasParent}^{\mathcal{I}} = \{(b, a), (b, e), (d, c), (d, b)\}$

$\text{hasChild}^{\mathcal{I}} = \{(a, b), (e, b), (c, d), (b, d)\}$

$\text{isMarriedTo}^{\mathcal{I}} = \{(d, f)\}$



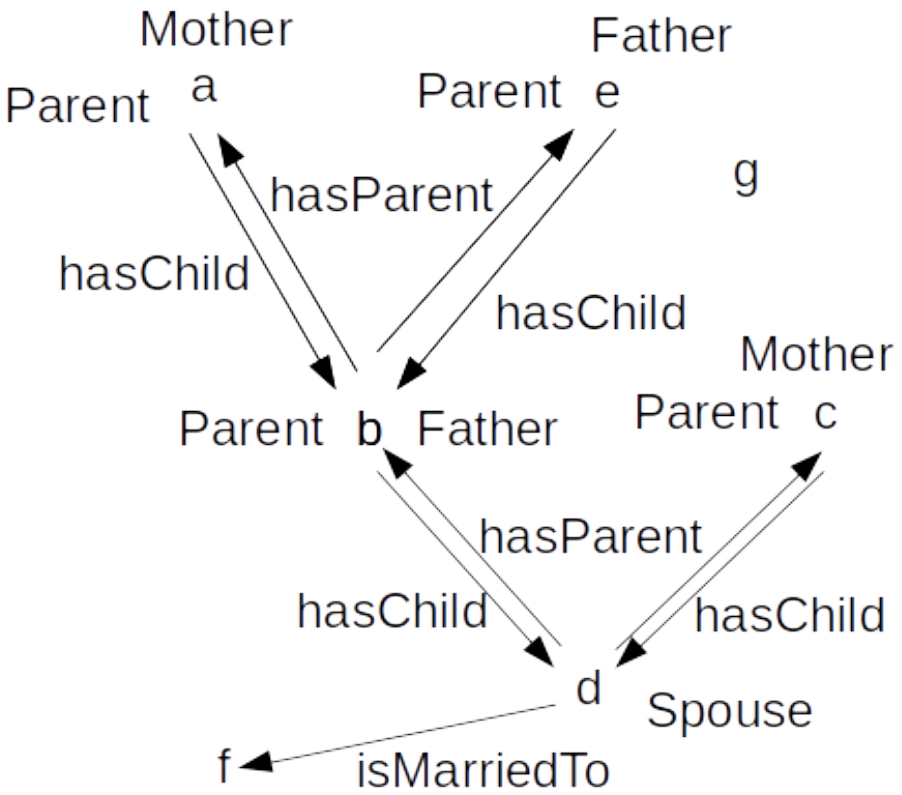
Description Logics: Semantics

The function $\cdot^{\mathcal{I}}$ is extended to complex concepts and roles to formalize the meaning of the constructors:

- $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = \emptyset$
- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(R^{-})^{\mathcal{I}} = \{(u, v) \mid (v, u) \in R^{\mathcal{I}}\}$
- $(\exists R.C)^{\mathcal{I}} = \{u \mid$ there exists $(u, v) \in R^{\mathcal{I}}$ such that $v \in C^{\mathcal{I}}\}$
- $(\forall R.C)^{\mathcal{I}} = \{u \mid$ for every v, if $(u, v) \in R^{\mathcal{I}}$ then $v \in C^{\mathcal{I}}\}$
- ...

Description Logics: Semantics

Example



$(\neg \text{Parent})^{\mathcal{I}} = ?$

$(\exists \text{hasParent}.\top)^{\mathcal{I}} = ?$

$(\text{isMarriedTo}^{-})^{\mathcal{I}} = ?$

$(\text{Spouse} \sqcup \text{Mother})^{\mathcal{I}} = ?$

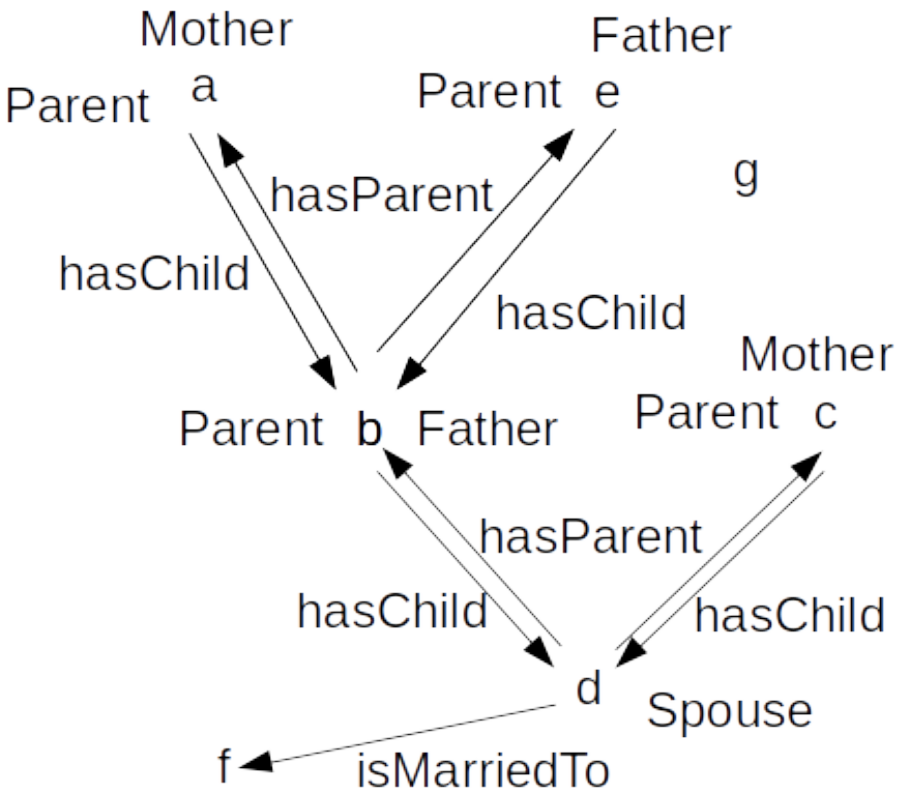
$(\forall \text{hasChild}.\text{Spouse})^{\mathcal{I}} = ?$

$((\forall \text{hasChild}.\text{Spouse}) \sqcap (\exists \text{hasChild}.\top))^{\mathcal{I}} = ?$

$(\text{Mother} \sqcap (\exists \text{hasChild}.\exists \text{hasChild}^{-}.\exists \text{hasParent}.\text{Father}))^{\mathcal{I}} = ?$

Description Logics: Semantics

Example



$$(\neg \text{Parent})^{\mathcal{I}} = \{d, f, g\}$$

$$(\exists \text{hasParent}.\top)^{\mathcal{I}} = \{b, d\}$$

$$(\text{isMarriedTo}^{-})^{\mathcal{I}} = \{(f, d)\}$$

$$(\text{Spouse} \sqcup \text{Mother})^{\mathcal{I}} = \{a, c, d\}$$

$$(\forall \text{hasChild}.\text{Spouse})^{\mathcal{I}} = \{b, c, d, f, g\}$$

$$((\forall \text{hasChild}.\text{Spouse}) \sqcap (\exists \text{hasChild}.\top))^{\mathcal{I}} = \{b, c\}$$

$$(\text{Mother} \sqcap (\exists \text{hasChild}.\exists \text{hasChild}^{-}.\exists \text{hasParent}.\text{Father}))^{\mathcal{I}} = \{c\}$$

Description Logics: Semantics

Satisfaction of TBox axioms

- $\mathcal{I}$ satisfies a concept inclusion $C \sqsubseteq D$, written $\mathcal{I} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I}$ satisfies a role inclusion $R \sqsubseteq S$, written $\mathcal{I} \models R \sqsubseteq S$, if $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
- $\mathcal{I}$ satisfies $(\mathit{func}\ R)$, written $\mathcal{I} \models (\mathit{func}\ R)$, if $R^{\mathcal{I}}$ is a functional relation
- ...

Description Logics: Semantics

Satisfaction of TBox axioms

- $\mathcal{I}$ satisfies a concept inclusion $C \sqsubseteq D$, written $\mathcal{I} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I}$ satisfies a role inclusion $R \sqsubseteq S$, written $\mathcal{I} \models R \sqsubseteq S$, if $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
- $\mathcal{I}$ satisfies $(\mathit{func}\ R)$, written $\mathcal{I} \models (\mathit{func}\ R)$, if $R^{\mathcal{I}}$ is a functional relation
- ...

Satisfaction of ABox assertions

- $\mathcal{I}$ satisfies a concept assertion $C(a)$, written $\mathcal{I} \models C(a)$, if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- $\mathcal{I}$ satisfies a role assertion $R(a, b)$, written $\mathcal{I} \models R(a, b)$, if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

Description Logics: Semantics

Satisfaction of TBox axioms

- $\mathcal{I}$ satisfies a concept inclusion $C \sqsubseteq D$, written $\mathcal{I} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I}$ satisfies a role inclusion $R \sqsubseteq S$, written $\mathcal{I} \models R \sqsubseteq S$, if $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
- $\mathcal{I}$ satisfies $(\mathit{func}\ R)$, written $\mathcal{I} \models (\mathit{func}\ R)$, if $R^{\mathcal{I}}$ is a functional relation
- ...

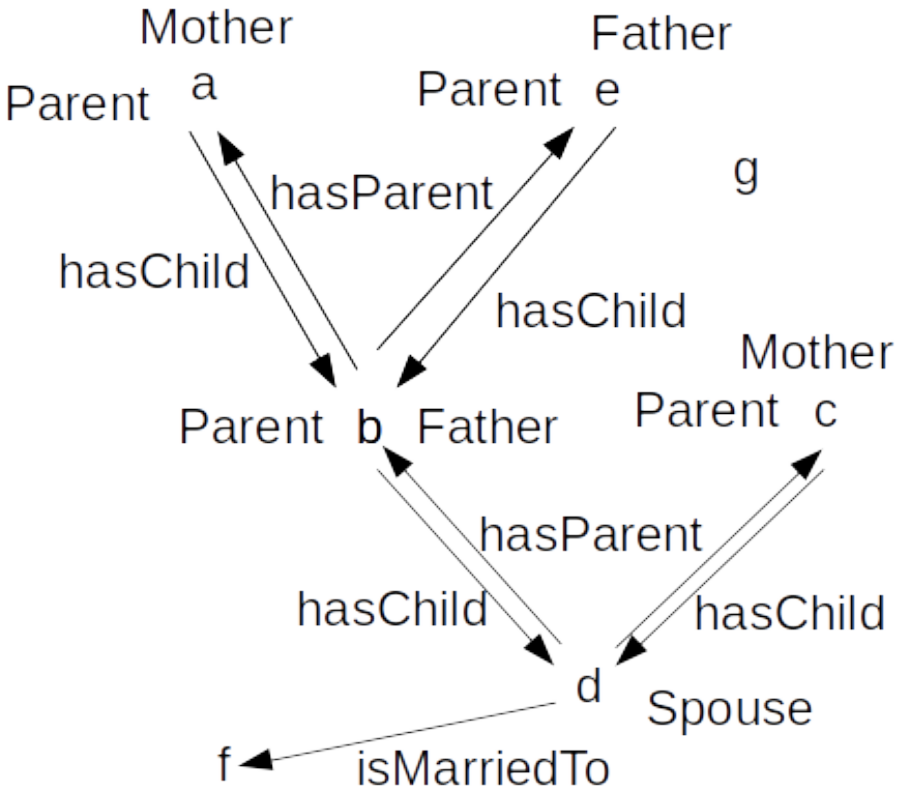
Satisfaction of ABox assertions

- $\mathcal{I}$ satisfies a concept assertion $C(a)$, written $\mathcal{I} \models C(a)$, if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- $\mathcal{I}$ satisfies a role assertion $R(a, b)$, written $\mathcal{I} \models R(a, b)$, if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

Open-world assumption: the absence of an assertion does not mean that it is false (different from closed-world assumption used for databases)

Description Logics: Semantics

Example



Assuming that $alice^{\mathcal{I}} = a$ and $bob^{\mathcal{I}} = b$:

$\mathcal{I} \models \text{Mother} \sqsubseteq \text{Parent}$?

$\mathcal{I} \models \exists \text{hasChild}.\top \sqsubseteq \exists \text{hasParent}.\top$?

$\mathcal{I} \models \text{Mother} \sqsubseteq \neg \text{Father}$?

$\mathcal{I} \models (\mathit{func}\ \text{hasChild})$?

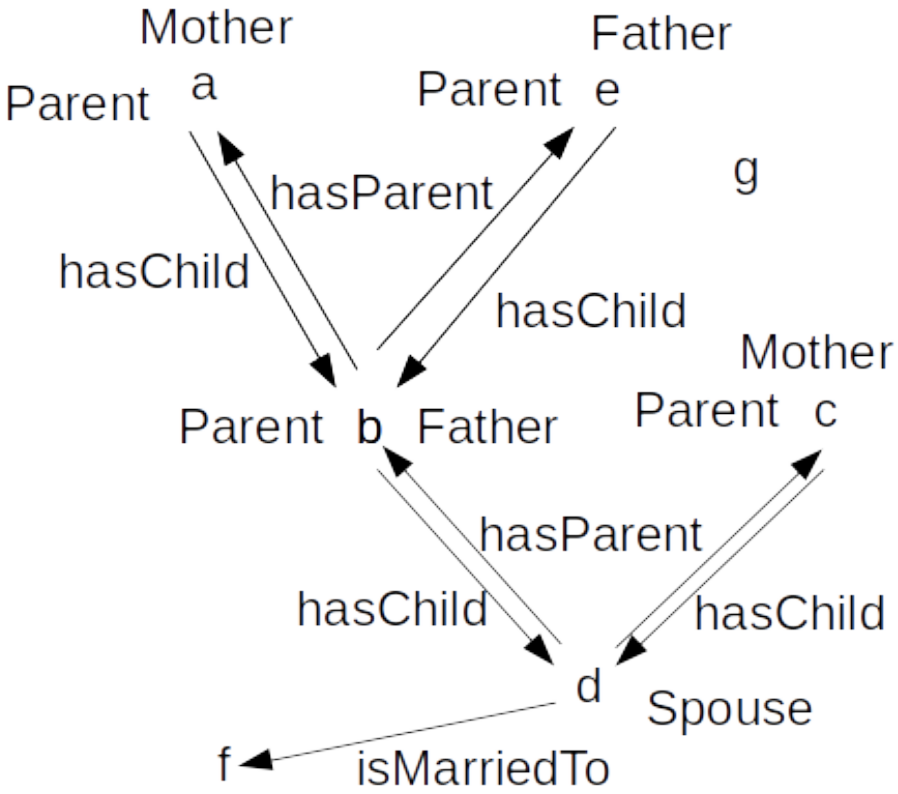
$\mathcal{I} \models \text{hasParent}(\mathit{bob}, \mathit{alice})$?

$\mathcal{I} \models \exists \text{hasChild}.(\text{Father} \sqcap \exists \text{hasChild}.\text{Spouse})(\mathit{alice})$?

$\mathcal{I} \models \forall \text{hasChild}.(\text{Father} \sqcap \forall \text{isMarriedTo}.\text{Spouse})(\mathit{alice})$?

Description Logics: Semantics

Example



Assuming that $alice^{\mathcal{I}} = a$ and $bob^{\mathcal{I}} = b$:

$\mathcal{I} \models \text{Mother} \sqsubseteq \text{Parent}$ ✓

$\mathcal{I} \models \exists \text{hasChild}.\top \sqsubseteq \exists \text{hasParent}.\top$ ✗

$\mathcal{I} \models \text{Mother} \sqsubseteq \neg \text{Father}$ ✓

$\mathcal{I} \models (\mathit{func}\ \text{hasChild})$ ✓

$\mathcal{I} \models \text{hasParent}(\mathit{bob}, \mathit{alice})$ ✓

$\mathcal{I} \models \exists \text{hasChild}.(\text{Father} \sqcap \exists \text{hasChild}.\text{Spouse})(\mathit{alice})$ ✓

$\mathcal{I} \models \forall \text{hasChild}.(\text{Father} \sqcap \forall \text{isMarriedTo}.\text{Spouse})(\mathit{alice})$ ✓

Description Logics: Semantics

Models

- $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ if it satisfies every axiom in $\mathcal{T}$
- $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if it satisfies every assertion in $\mathcal{A}$
- $\mathcal{I}$ is a model of a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ if it is a model of $\mathcal{T}$ and $\mathcal{A}$
- Two KBs are equivalent if they have the same models

Description Logics: Semantics

Models

- $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ if it satisfies every axiom in $\mathcal{T}$
- $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if it satisfies every assertion in $\mathcal{A}$
- $\mathcal{I}$ is a model of a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ if it is a model of $\mathcal{T}$ and $\mathcal{A}$
- Two KBs are equivalent if they have the same models

Satisfiability

- A KB $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, or consistent, if it has at least one model
- A concept C is satisfiable w.r.t. a TBox $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$

Description Logics: Semantics

Models

- $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ if it satisfies every axiom in $\mathcal{T}$
- $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if it satisfies every assertion in $\mathcal{A}$
- $\mathcal{I}$ is a model of a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ if it is a model of $\mathcal{T}$ and $\mathcal{A}$
- Two KBs are equivalent if they have the same models

Satisfiability

- A KB $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, or consistent, if it has at least one model
- A concept C is satisfiable w.r.t. a TBox $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$

Entailment

- A TBox $\mathcal{T}$ entails an axiom α, written $\mathcal{T} \models \alpha$, if every model of $\mathcal{T}$ satisfies α
- A KB $\langle \mathcal{T}, \mathcal{A} \rangle$ entails an assertion α, written $\langle \mathcal{T}, \mathcal{A} \rangle \models \alpha$, if every model of $\langle \mathcal{T}, \mathcal{A} \rangle$ satisfies α

Description Logics: Semantics

Example

$$\mathcal{T} = \{\, \mathsf{Female} \sqcap \exists \mathsf{hasChild}.\top \sqsubseteq \mathsf{Mother},$$
$$\mathsf{Father} \sqsubseteq \mathsf{Male} \sqcap \exists \mathsf{hasChild}.\top,$$
$$\mathsf{Parent} \equiv \exists \mathsf{hasChild}.\top,$$
$$\top \sqsubseteq \exists \mathsf{hasChild}^{-}.\top \,\}$$
$$\mathcal{A} = \{\, \mathsf{hasChild}(\mathit{alice}, \mathit{bob}),\ \mathsf{Female}(\mathit{alice}),\ \mathsf{Father}(\mathit{bob}) \,\}$$

Note: $A \equiv B$ is a shorthand for $A \sqsubseteq B$ and $B \sqsubseteq A$

- $\mathcal{T} \models \mathsf{Father} \sqsubseteq \mathsf{Parent}$?
- $\mathcal{T} \models \mathsf{Mother} \sqsubseteq \mathsf{Parent}$?
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \mathsf{Mother}(\mathit{alice})$?
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \mathsf{Male}(\mathit{bob})$?
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \forall \mathsf{hasChild}.\mathsf{Male}(\mathit{alice})$?
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \exists \mathsf{hasChild}^{-}.\exists \mathsf{hasChild}^{-}.\top(\mathit{alice})$?

Description Logics: Semantics

Example

$$\mathcal{T} = \{\, \mathsf{Female} \sqcap \exists \mathsf{hasChild}.\top \sqsubseteq \mathsf{Mother},$$
$$\mathsf{Father} \sqsubseteq \mathsf{Male} \sqcap \exists \mathsf{hasChild}.\top,$$
$$\mathsf{Parent} \equiv \exists \mathsf{hasChild}.\top,$$
$$\top \sqsubseteq \exists \mathsf{hasChild}^{-}.\top \,\}$$
$$\mathcal{A} = \{\, \mathsf{hasChild}(\mathit{alice}, \mathit{bob}),\ \mathsf{Female}(\mathit{alice}),\ \mathsf{Father}(\mathit{bob}) \,\}$$

Note: $A \equiv B$ is a shorthand for $A \sqsubseteq B$ and $B \sqsubseteq A$

- $\mathcal{T} \models \mathsf{Father} \sqsubseteq \mathsf{Parent}$ ✓
- $\mathcal{T} \models \mathsf{Mother} \sqsubseteq \mathsf{Parent}$ ✗
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \mathsf{Mother}(\mathit{alice})$ ✓
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \mathsf{Male}(\mathit{bob})$ ✓
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \forall \mathsf{hasChild}.\mathsf{Male}(\mathit{alice})$ ✗
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \exists \mathsf{hasChild}^{-}.\exists \mathsf{hasChild}^{-}.\top(\mathit{alice})$ ✓

Description Logics: Semantics

Example

$$\mathcal{T} = \{\, \top \sqsubseteq \text{Male} \sqcup \text{Female},\ \text{Male} \sqcap \text{Female} \sqsubseteq \bot,\ \exists \text{friend}.(\text{Female} \sqcap \exists \text{loves}.\text{Male}) \sqsubseteq A \,\}$$

$$\mathcal{A} = \{\, \text{friend}(\mathit{john}, \mathit{susan}),\ \text{friend}(\mathit{john}, \mathit{andrea}),\ \text{loves}(\mathit{susan}, \mathit{andrea}),\ \text{loves}(\mathit{andrea}, \mathit{bill}),\ \text{Female}(\mathit{susan}),\ \text{Male}(\mathit{bill}) \,\}$$

$$\langle \mathcal{T}, \mathcal{A} \rangle \models A(\mathit{john})\,?$$

Description Logics: Semantics

Example

$$\mathcal{T} = \{\, \top \sqsubseteq \text{Male} \sqcup \text{Female},\ \text{Male} \sqcap \text{Female} \sqsubseteq \bot,\ \exists \text{friend}.(\text{Female} \sqcap \exists \text{loves}.\text{Male}) \sqsubseteq A \,\}$$

$$\mathcal{A} = \{\, \text{friend}(\mathit{john}, \mathit{susan}),\ \text{friend}(\mathit{john}, \mathit{andrea}),\ \text{loves}(\mathit{susan}, \mathit{andrea}),\ \text{loves}(\mathit{andrea}, \mathit{bill}),\ \text{Female}(\mathit{susan}),\ \text{Male}(\mathit{bill}) \,\}$$

$$\langle \mathcal{T}, \mathcal{A} \rangle \models A(\mathit{john}) \checkmark$$

Defining a Particular DL

To define a particular DL, we need to specify

- which concept and role constructors can be used
- what types of statements can appear in the TBox

Defining a Particular DL

To define a particular DL, we need to specify

- which concept and role constructors can be used
- what types of statements can appear in the TBox

For example, the $\mathcal{ALC}$ DL ("Attributive Concept Language with Complements") is defined as follows:

- if A is an atomic concept, then A is an $\mathcal{ALC}$ concept
- if C, D are $\mathcal{ALC}$ concepts and R is an atomic role, then the following are $\mathcal{ALC}$ concepts:
  - $C \sqcap D$ (conjunction)
  - $C \sqcup D$ (disjunction)
  - $\neg C$ (negation)
  - $\exists R.C$ (existential restriction)
  - $\forall R.C$ (value restriction)
- an $\mathcal{ALC}$ TBox contains only concept inclusions

Note that $A \sqcap \neg A$ can be abbreviated by $\bot$ and $A \sqcup \neg A$ by $\top$.

Relationship with First-Order Logic

DL KBs can be translated into first-order logic (FOL):

- atomic concepts and roles are unary and binary predicates
- complex concepts are FOL formula with one free variable
  - $\text{Female} \sqcap \exists \text{hasChild}.\top$ $\qquad$ $\text{Female}(x) \wedge \exists y \text{hasChild}(x, y)$
- TBox and ABox axioms are FOL sentences
  - $\exists \text{hasChild}.\top \sqsubseteq \text{Parent}$ $\qquad$ $\forall x(\exists y \text{hasChild}(x, y) \Rightarrow \text{Parent}(x))$

Relationship with First-Order Logic

Example: Translation of an $\mathcal{ALC}$ TBox

Concept C is translated into FOL formula with one free variable $\pi_x(C)$ inductively defined as follows

- $\pi_x(A) = A(x)$ for A atomic concept
- $\pi_x(C \sqcap D) = \pi_x(C) \wedge \pi_x(D)$
- $\pi_x(C \sqcup D) = \pi_x(C) \vee \pi_x(D)$
- $\pi_x(\neg C) = \neg \pi_x(C)$
- $\pi_x(\exists R.C) = \exists y(R(x, y) \wedge \pi_y(C))$, y different from x
- $\pi_x(\forall R.C) = \forall y(R(x, y) \Rightarrow \pi_y(C))$, y different from x

Concept inclusion $C \sqsubseteq D$ is translated into FOL sentence $\pi(C \sqsubseteq D) = \forall x(\pi_x(C) \Rightarrow \pi_x(D))$

Relationship between DLs and OWL

OWL adopts different terminology and syntax(es) than DLs but OWL axioms can be translated into DLs axioms

- OWL classes are concepts in DLs
- OWL properties are roles in DLs

Examples of OWL expressions and their DL counterparts

OWL	DL
owl : Thing	$\top$
owl : intersectionOf(C_1, C_2)	$C_1 \sqcap C_2$
owl : complementOf(C)	$\neg C$
owl : Restriction(R owl : someValuesFrom(C))	$\exists R.C$
rdfs : subClassOf(C_1, C_2)	$C_1 \sqsubseteq C_2$
owl : disjointWith(C_1, C_2)	$C_1 \sqsubseteq \neg C_2$
owl : ClassAssertion(C a)	$C(a)$
owl : ObjectPropertyAssertion(R a b)	$R(a, b)$
...	

Relationship between DLs and OWL

OWL 2 DL (decidable fragment of OWL Full) corresponds to the DL $\mathcal{SROIQ}(D)$

- $\mathcal{S}$ stands for $\mathcal{ALC}$ extended with transitive roles ($(trans\ R)$),
- $\mathcal{R}$: regular role hierarchies (role inclusions with some suitable acyclicity conditions) ($R_1 \sqsubseteq R_2$),
- $\mathcal{O}$: nominals (possibility of using individuals in the TBox: concept $\{o\}$ where o is an individual),
- $\mathcal{I}$: inverse roles (R^-),
- $\mathcal{Q}$: qualified number restrictions ($\leq n\ R.C$, $\geq n\ R.C$),
- (D) : data types.

Relationship between DLs and OWL

Mapping (sub-languages of) OWL to equivalent DLs provides a well-defined semantics and allows us to use results of DL research

- One of the two semantics of OWL 2 is directly based on DLs
- Complexity results, algorithms and implemented reasoners
- OWL 2 profiles (OWL 2 EL, OWL 2 QL, and OWL 2 RL) correspond to DL languages with interesting computational properties, targeted towards a specific use (we will see some of them in more detail later)

Existential Rules

Even very expressive DLs cannot express some useful (and simple) relationships

- $\forall x(\text{Boss}(x) \Rightarrow \text{supervisorOf}(x, x))$
- $\forall sxyd_1d_2(\text{marriageSp}_1(s, x) \wedge \text{marriageSp}_2(s, y) \wedge \text{marriageStart}(s, d_1) \wedge \text{marriageEnd}(s, d_2) \Rightarrow \exists p\, \text{marriageSp}_1(p, y) \wedge \text{marriageSp}_2(p, x) \wedge \text{marriageStart}(p, d_1) \wedge \text{marriageEnd}(p, d_2))$
- $\forall xyd_1d_2(\text{marriedFromTo}(x, y, d_1, d_2) \Rightarrow \text{marriedFromTo}(y, x, d_1, d_2))$

Another family of FOL fragments overcomes this limitations: Existential Rules (a.k.a. Datalog$^{\pm}$ rules, or tuple-generating dependencies)

Existential Rules: Syntax

Basic building blocks

- a countable set C of constants
  - *alice*, *bob* ...
- a countable set N of (labeled) nulls
  - $\perp_1$, $\perp_2$ ...
- a countable set V of variables
  - x, y ...
- a countable set of predicates (of any arity)
  - Person, Employee, Customer, Order...

Existential Rules: Syntax

Basic building blocks

- a countable set C of constants
  - *alice*, *bob* ...
- a countable set N of (labeled) nulls
  - $\perp_1$, $\perp_2$ ...
- a countable set V of variables
  - x, y ...
- a countable set of predicates (of any arity)
  - Person, Employee, Customer, Order...

A term is a constant, null or variable

An atom has the form $P(t_1, \ldots, t_n)$ where P is an n-ary predicate and the t_i are terms

- Person(*bob*, *smith*, 1985.11.03), Order(*smith*, x), Customer(*smith*, *smith@mail.com*, $\perp_{15}$)...

Existential Rules: Syntax

Knowledge Base

=

ontology (set of existential rules) + database (set of facts)

An existential rule is an expression of the form

$$\forall \vec{X} \forall \vec{Y} \; (\phi(\vec{X}, \vec{Y}) \rightarrow \exists \vec{Z} \psi(\vec{X}, \vec{Z}))$$

- $\vec{X}$, $\vec{Y}$ and $\vec{Z}$ are tuples of variables
- $\phi(\vec{X}, \vec{Y})$ and $\psi(\vec{X}, \vec{Z})$ are conjunctions of atoms with terms in $\vec{X} \cup \vec{Y}$ and $\vec{X} \cup \vec{Z}$ respectively
- $\phi(\vec{X}, \vec{Y})$ is called the body of the rule
- $\psi(\vec{X}, \vec{Z})$ is called the head of the rule

Quantifiers may be left implicit: $\phi(\vec{X}, \vec{Y}) \rightarrow \psi(\vec{X}, \vec{Z})$

A fact is a variable-free atom

Existential Rules: Semantics

Homomorphisms

- We could use FOL semantics ($\sim$ DL semantics) but ER semantics is traditionally defined via the notion of homomorphisms, that will also be useful later

- A homomorphism from a set of atoms A to a set of atoms B is a substitution $h : \mathsf{C} \cup \mathsf{N} \cup \mathsf{V} \rightarrow \mathsf{C} \cup \mathsf{N} \cup \mathsf{V}$ such that
  - $h(t) = t$ for all $t \in \mathsf{C}$ - unique name assumption
    Note: not mandatory in FOL / DL semantics
  - $P(t_1, \ldots, t_n) \in \mathsf{A} \Rightarrow P(h(t_1), \ldots, h(t_n)) \in \mathsf{B}$

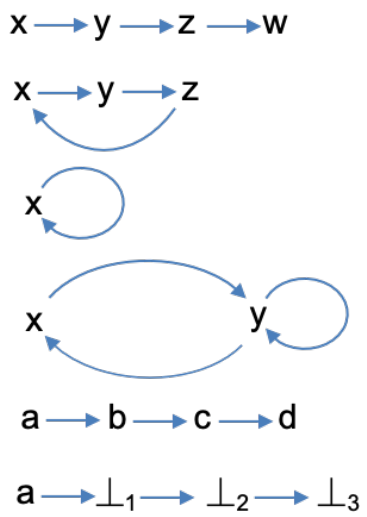
- Extended to conjunctions of atoms seen as set of atoms

Existential Rules: Semantics

Homomorphisms: Examples

Find the homomorphisms (x, y, z, w variables, a, b, c, d constants)

$$A_1 = \{P(x,y), P(y,z), P(z,w)\}$$
$$A_2 = \{P(x,y), P(y,z), P(z,x)\}$$
$$A_3 = \{P(x,x)\}$$
$$A_4 = \{P(x,y), P(y,x), P(y,y)\}$$
$$A_5 = \{P(a,b), P(b,c), P(c,d)\}$$
$$A_6 = \{P(a,\bot_1), P(\bot_1,\bot_2), P(\bot_2,\bot_3)\}$$

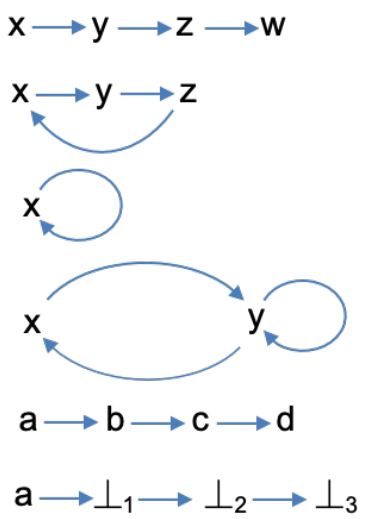


Existential Rules: Semantics

Homomorphisms: Examples

Find the homomorphisms (x, y, z, w variables, a, b, c, d constants)

$$\mathrm{A}_1 = \{P(x,y), P(y,z), P(z,w)\}$$
$$\mathrm{A}_2 = \{P(x,y), P(y,z), P(z,x)\}$$
$$\mathrm{A}_3 = \{P(x,x)\}$$
$$\mathrm{A}_4 = \{P(x,y), P(y,x), P(y,y)\}$$
$$\mathrm{A}_5 = \{P(a,b), P(b,c), P(c,d)\}$$
$$\mathrm{A}_6 = \{P(a,\bot_1), P(\bot_1,\bot_2), P(\bot_2,\bot_3)\}$$



h from A_1 to A_2

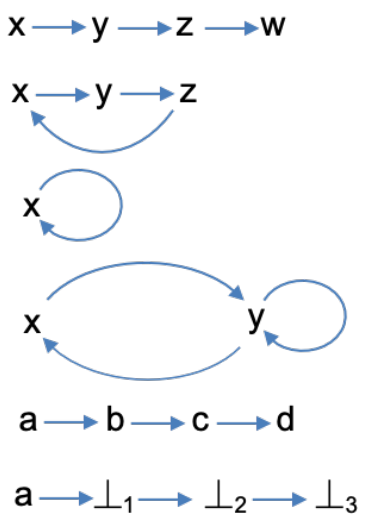
$$h(x) = x$$
$$h(y) = y$$
$$h(z) = z$$
$$h(w) = x$$

Existential Rules: Semantics

Homomorphisms: Examples

Find the homomorphisms (x, y, z, w variables, a, b, c, d constants)

$$\mathrm{A}_1 = \{P(x,y), P(y,z), P(z,w)\}$$
$$\mathrm{A}_2 = \{P(x,y), P(y,z), P(z,x)\}$$
$$\mathrm{A}_3 = \{P(x,x)\}$$
$$\mathrm{A}_4 = \{P(x,y), P(y,x), P(y,y)\}$$
$$\mathrm{A}_5 = \{P(a,b), P(b,c), P(c,d)\}$$
$$\mathrm{A}_6 = \{P(a,\bot_1), P(\bot_1,\bot_2), P(\bot_2,\bot_3)\}$$



h from A_1 to A_3

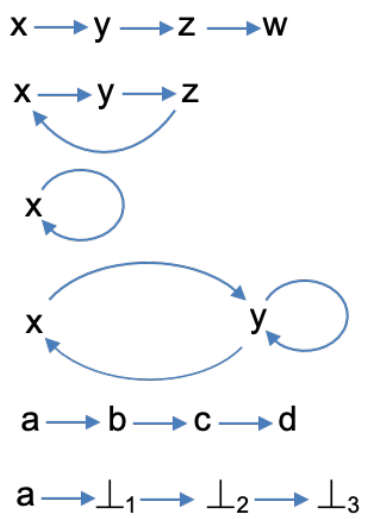
$$h(x) = x$$
$$h(y) = x$$
$$h(z) = x$$
$$h(w) = x$$

Existential Rules: Semantics

Homomorphisms: Examples

Find the homomorphisms (x, y, z, w variables, a, b, c, d constants)

$$\mathrm{A}_1 = \{P(x,y), P(y,z), P(z,w)\}$$
$$\mathrm{A}_2 = \{P(x,y), P(y,z), P(z,x)\}$$
$$\mathrm{A}_3 = \{P(x,x)\}$$
$$\mathrm{A}_4 = \{P(x,y), P(y,x), P(y,y)\}$$
$$\mathrm{A}_5 = \{P(a,b), P(b,c), P(c,d)\}$$
$$\mathrm{A}_6 = \{P(a,\bot_1), P(\bot_1,\bot_2), P(\bot_2,\bot_3)\}$$



h from A_1 to A_4

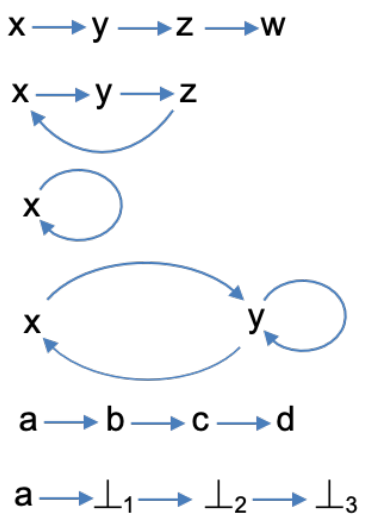
$$h(x) = x$$
$$h(y) = y$$
$$h(z) = y$$
$$h(w) = x$$

Existential Rules: Semantics

Homomorphisms: Examples

Find the homomorphisms (x, y, z, w variables, a, b, c, d constants)

$$\mathrm{A}_1 = \{P(x,y), P(y,z), P(z,w)\}$$
$$\mathrm{A}_2 = \{P(x,y), P(y,z), P(z,x)\}$$
$$\mathrm{A}_3 = \{P(x,x)\}$$
$$\mathrm{A}_4 = \{P(x,y), P(y,x), P(y,y)\}$$
$$\mathrm{A}_5 = \{P(a,b), P(b,c), P(c,d)\}$$
$$\mathrm{A}_6 = \{P(a,\bot_1), P(\bot_1,\bot_2), P(\bot_2,\bot_3)\}$$



h from A_1 to A_5

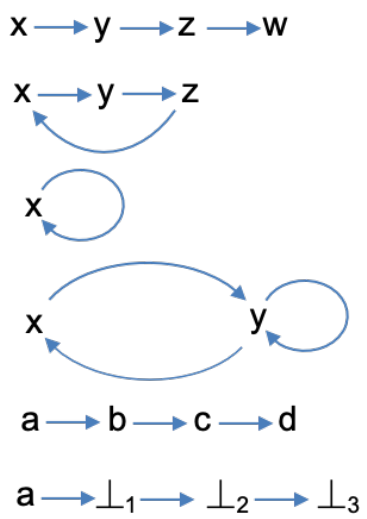
$$h(x) = a$$
$$h(y) = b$$
$$h(z) = c$$
$$h(w) = d$$

Existential Rules: Semantics

Homomorphisms: Examples

Find the homomorphisms (x, y, z, w variables, a, b, c, d constants)

$$A_1 = \{P(x,y), P(y,z), P(z,w)\}$$
$$A_2 = \{P(x,y), P(y,z), P(z,x)\}$$
$$A_3 = \{P(x,x)\}$$
$$A_4 = \{P(x,y), P(y,x), P(y,y)\}$$
$$A_5 = \{P(a,b), P(b,c), P(c,d)\}$$
$$A_6 = \{P(a,\bot_1), P(\bot_1,\bot_2), P(\bot_2,\bot_3)\}$$



h from A_1 to A_6

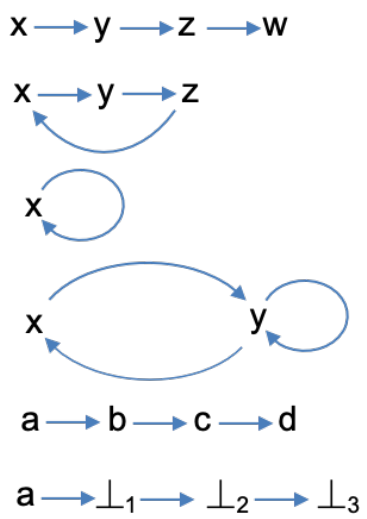
$$h(x) = a$$
$$h(y) = \bot_1$$
$$h(z) = \bot_2$$
$$h(w) = \bot_3$$

Existential Rules: Semantics

Homomorphisms: Examples

Find the homomorphisms (x, y, z, w variables, a, b, c, d constants)

$$\mathrm{A}_1 = \{P(x,y), P(y,z), P(z,w)\}$$
$$\mathrm{A}_2 = \{P(x,y), P(y,z), P(z,x)\}$$
$$\mathrm{A}_3 = \{P(x,x)\}$$
$$\mathrm{A}_4 = \{P(x,y), P(y,x), P(y,y)\}$$
$$\mathrm{A}_5 = \{P(a,b), P(b,c), P(c,d)\}$$
$$\mathrm{A}_6 = \{P(a,\bot_1), P(\bot_1,\bot_2), P(\bot_2,\bot_3)\}$$



h from A_2 to A_3

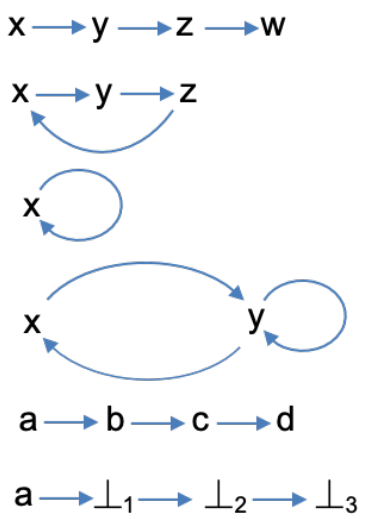
$$h(x) = x$$
$$h(y) = x$$
$$h(z) = x$$

Existential Rules: Semantics

Homomorphisms: Examples

Find the homomorphisms (x, y, z, w variables, a, b, c, d constants)

$$A_1 = \{P(x,y), P(y,z), P(z,w)\}$$
$$A_2 = \{P(x,y), P(y,z), P(z,x)\}$$
$$A_3 = \{P(x,x)\}$$
$$A_4 = \{P(x,y), P(y,x), P(y,y)\}$$
$$A_5 = \{P(a,b), P(b,c), P(c,d)\}$$
$$A_6 = \{P(a,\bot_1), P(\bot_1,\bot_2), P(\bot_2,\bot_3)\}$$



h from A_2 to A_4

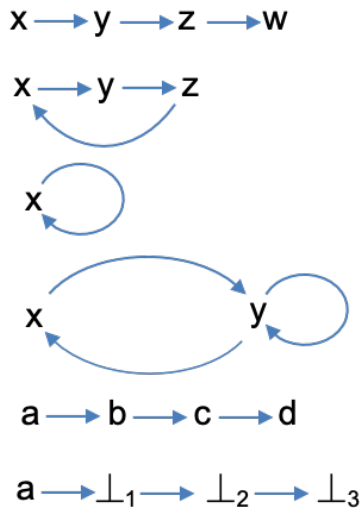
$$h(x) = x$$
$$h(y) = y$$
$$h(z) = y$$

Existential Rules: Semantics

Homomorphisms: Examples

Find the homomorphisms (x, y, z, w variables, a, b, c, d constants)

$$\mathrm{A}_1 = \{P(x,y), P(y,z), P(z,w)\}$$
$$\mathrm{A}_2 = \{P(x,y), P(y,z), P(z,x)\}$$
$$\mathrm{A}_3 = \{P(x,x)\}$$
$$\mathrm{A}_4 = \{P(x,y), P(y,x), P(y,y)\}$$
$$\mathrm{A}_5 = \{P(a,b), P(b,c), P(c,d)\}$$
$$\mathrm{A}_6 = \{P(a,\bot_1), P(\bot_1,\bot_2), P(\bot_2,\bot_3)\}$$



h from A_3 to A_4

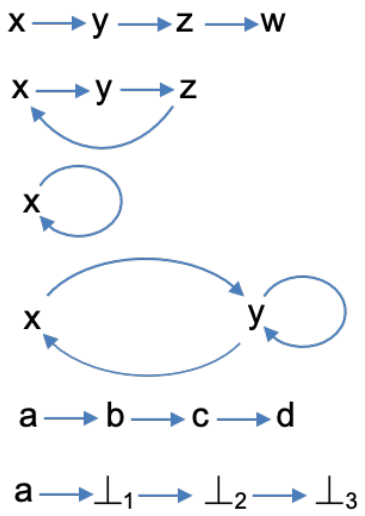
$$h(x) = y$$

Existential Rules: Semantics

Homomorphisms: Examples

Find the homomorphisms (x, y, z, w variables, a, b, c, d constants)

$$\mathrm{A}_1 = \{P(x,y), P(y,z), P(z,w)\}$$
$$\mathrm{A}_2 = \{P(x,y), P(y,z), P(z,x)\}$$
$$\mathrm{A}_3 = \{P(x,x)\}$$
$$\mathrm{A}_4 = \{P(x,y), P(y,x), P(y,y)\}$$
$$\mathrm{A}_5 = \{P(a,b), P(b,c), P(c,d)\}$$
$$\mathrm{A}_6 = \{P(a,\bot_1), P(\bot_1,\bot_2), P(\bot_2,\bot_3)\}$$



h from A_4 to A_3

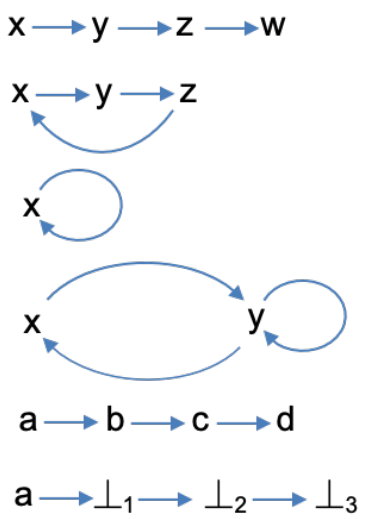
$$h(x) = x$$
$$h(y) = x$$

Existential Rules: Semantics

Homomorphisms: Examples

Find the homomorphisms (x, y, z, w variables, a, b, c, d constants)

$$\mathrm{A}_1 = \{P(x,y), P(y,z), P(z,w)\}$$
$$\mathrm{A}_2 = \{P(x,y), P(y,z), P(z,x)\}$$
$$\mathrm{A}_3 = \{P(x,x)\}$$
$$\mathrm{A}_4 = \{P(x,y), P(y,x), P(y,y)\}$$
$$\mathrm{A}_5 = \{P(a,b), P(b,c), P(c,d)\}$$
$$\mathrm{A}_6 = \{P(a,\bot_1), P(\bot_1,\bot_2), P(\bot_2,\bot_3)\}$$



h from A_6 to A_5

$$h(a) = a$$
$$h(\bot_1) = b$$
$$h(\bot_2) = c$$
$$h(\bot_3) = d$$

Existential Rules: Semantics

Models

- A set of facts J is a model of the rule
$$\sigma = \forall \vec{X} \forall \vec{Y} \ (\phi(\vec{X}, \vec{Y}) \rightarrow \exists \vec{Z} \psi(\vec{X}, \vec{Z})),$$
written $J \models \sigma$, if whenever there exists a homomorphism h such that $h(\phi(\vec{X}, \vec{Y})) \subseteq J$, then there exists a homomorphism g such that $g(t) = h(t)$ for every $t \in \vec{X}$ and $g(\psi(\vec{X}, \vec{Z})) \subseteq J$
- J is a model of an ontology Σ if it models all rules in Σ
- J is a model of a database D if $D \subseteq J$

Existential Rules: Semantics

Models

- A set of facts J is a model of the rule
$$\sigma = \forall \vec{X} \forall \vec{Y} \ (\phi(\vec{X}, \vec{Y}) \rightarrow \exists \vec{Z} \psi(\vec{X}, \vec{Z})),$$
written $J \models \sigma$, if whenever there exists a homomorphism h such that $h(\phi(\vec{X}, \vec{Y})) \subseteq J$, then there exists a homomorphism g such that $g(t) = h(t)$ for every $t \in \vec{X}$ and $g(\psi(\vec{X}, \vec{Z})) \subseteq J$
- J is a model of an ontology Σ if it models all rules in Σ
- J is a model of a database D if $D \subseteq J$

Entailment

- $\langle \Sigma, D \rangle$ entails a rule σ if every model of $\langle \Sigma, D \rangle$ is a model of σ
- $\langle \Sigma, D \rangle$ entails a fact if every model of $\langle \Sigma, D \rangle$ contains the fact

Existential Rules: Semantics

Example

$$\Sigma = \{\text{Order}(c, o) \rightarrow \text{Customer}(c, m, p) \wedge \text{Product}(o)$$
$$\text{Customer}(n, m, p) \rightarrow \text{HasEmail}(n, m) \wedge \text{HasPhone}(n, p)$$
$$\text{HasEmail}(n, m) \rightarrow \text{HasContact}(n)$$
$$\text{HasPhone}(n, m) \rightarrow \text{HasContact}(n)$$
$$\text{Product}(x) \rightarrow \text{HasPrice}(x, y)$$
$$\text{Book}(x) \wedge \text{HasPrice}(x, y) \rightarrow \text{HasDiscount}(y, z)\}$$
$$D = \{\text{Order}(\mathit{smith}, \mathit{hamlet}), \text{Book}(\mathit{hamlet}), \text{HasPrice}(\mathit{hamlet}, 10)\}$$

- $\langle \Sigma, D \rangle \models \text{HasContact}(\mathit{smith})$?
- $\langle \Sigma, D \rangle \models (\text{Order}(x, y) \rightarrow \text{HasPrice}(y, z))$?
- $\langle \Sigma, D \rangle \models (\text{Order}(x, y) \rightarrow \text{HasDiscount}(y, z))$?
- $\langle \Sigma, D \rangle \models \text{HasDiscount}(10, 2)$?

Existential Rules: Semantics

Example

$$\Sigma = \{\text{Order}(c, o) \rightarrow \text{Customer}(c, m, p) \wedge \text{Product}(o)$$
$$\text{Customer}(n, m, p) \rightarrow \text{HasEmail}(n, m) \wedge \text{HasPhone}(n, p)$$
$$\text{HasEmail}(n, m) \rightarrow \text{HasContact}(n)$$
$$\text{HasPhone}(n, m) \rightarrow \text{HasContact}(n)$$
$$\text{Product}(x) \rightarrow \text{HasPrice}(x, y)$$
$$\text{Book}(x) \wedge \text{HasPrice}(x, y) \rightarrow \text{HasDiscount}(y, z)\}$$
$$D = \{\text{Order}(\mathit{smith}, \mathit{hamlet}), \text{Book}(\mathit{hamlet}), \text{HasPrice}(\mathit{hamlet}, 10)\}$$

- $\langle \Sigma, D \rangle \models \text{HasContact}(\mathit{smith})$ ✓
- $\langle \Sigma, D \rangle \models (\text{Order}(x, y) \rightarrow \text{HasPrice}(y, z))$ ✓
- $\langle \Sigma, D \rangle \models (\text{Order}(x, y) \rightarrow \text{HasDiscount}(y, z))$ ✗
- $\langle \Sigma, D \rangle \models \text{HasDiscount}(10, 2)$ ✗

Existential Rules vs Description Logics

Two families of fragments of FOL: similar formalisms, DL and ER ontologies can be translated to FOL

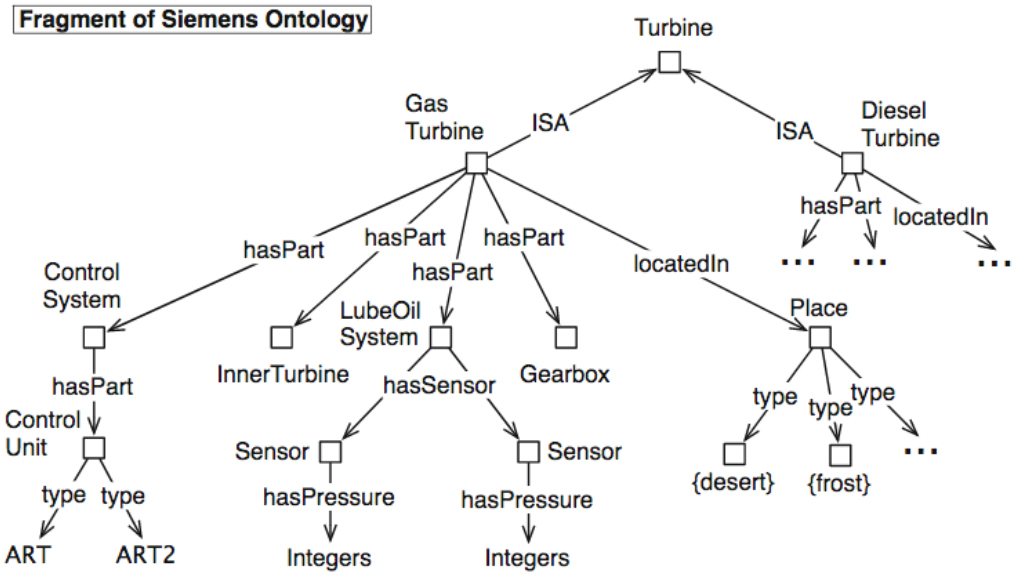
Incomparable expressiveness

- not expressible in DL
  - $\forall x\ (A(x) \rightarrow R(x,x))$
  - $\forall wxyz\ (R(w,x) \wedge R(w,y) \wedge R(w,z) \rightarrow \exists v\ S(v,x) \wedge S(v,y) \wedge S(v,z))$
  - $\forall xyz\ (R(x,y,z) \rightarrow R(x,y,z))$
- not expressible in existential rules
  - $A \sqsubseteq B \sqcup C$
  - $A \sqsubseteq \bot$

However, existential rules generalize several widely used Horn DLs (without disjunction)

Examples of Applications of Ontologies

Ontologies for Industry



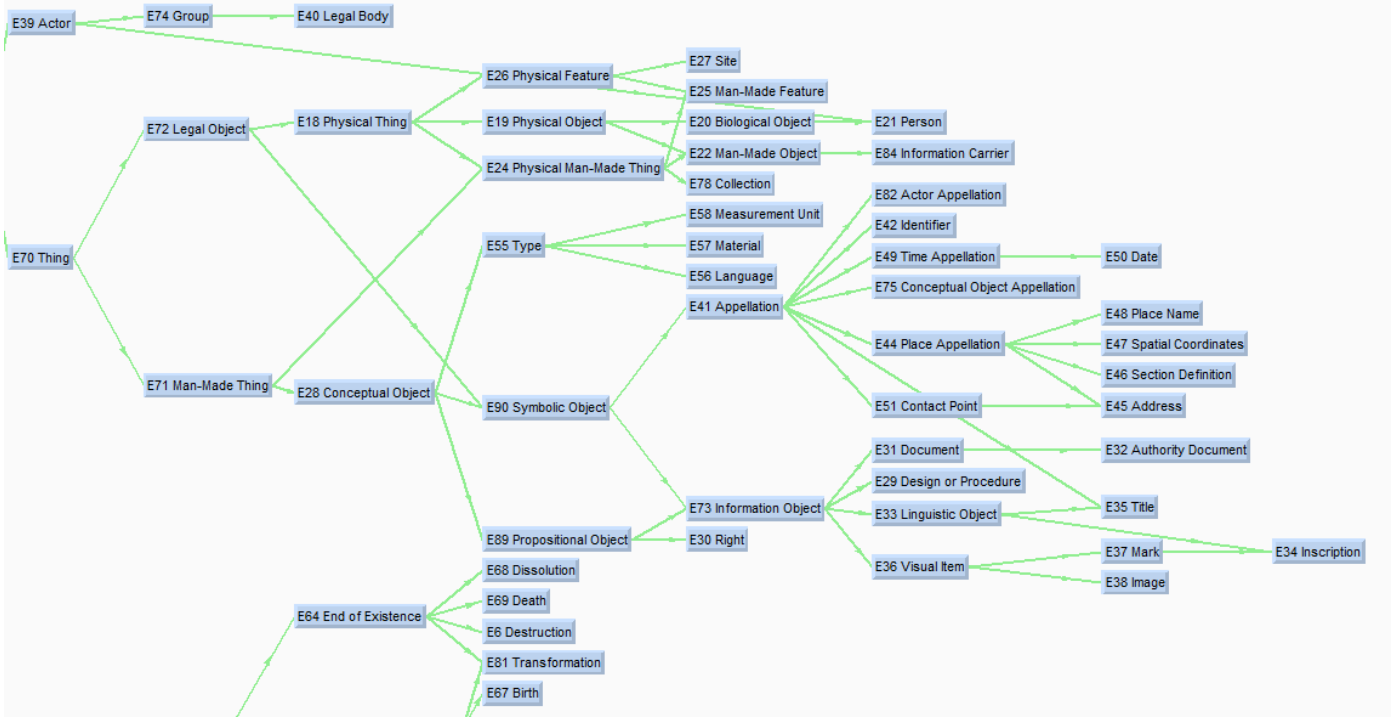
From: How Semantic Technologies Can Enhance Data Access at Siemens Energy, Kharlamov et al., ISWC 2014

- Energy sector: Optique EU project (several universities involved)
  - Siemens: turbines diagnostics
  - StatOil: find exploitable accumulations of oil or gas
- Aeronautics sector
  - Collaboration between Thales and Univ. Paris Sud on ontology-based solutions for avionics maintenance
  - NASA Air Traffic Management Ontology

Examples of Applications of Ontologies

Ontologies for Public Policies

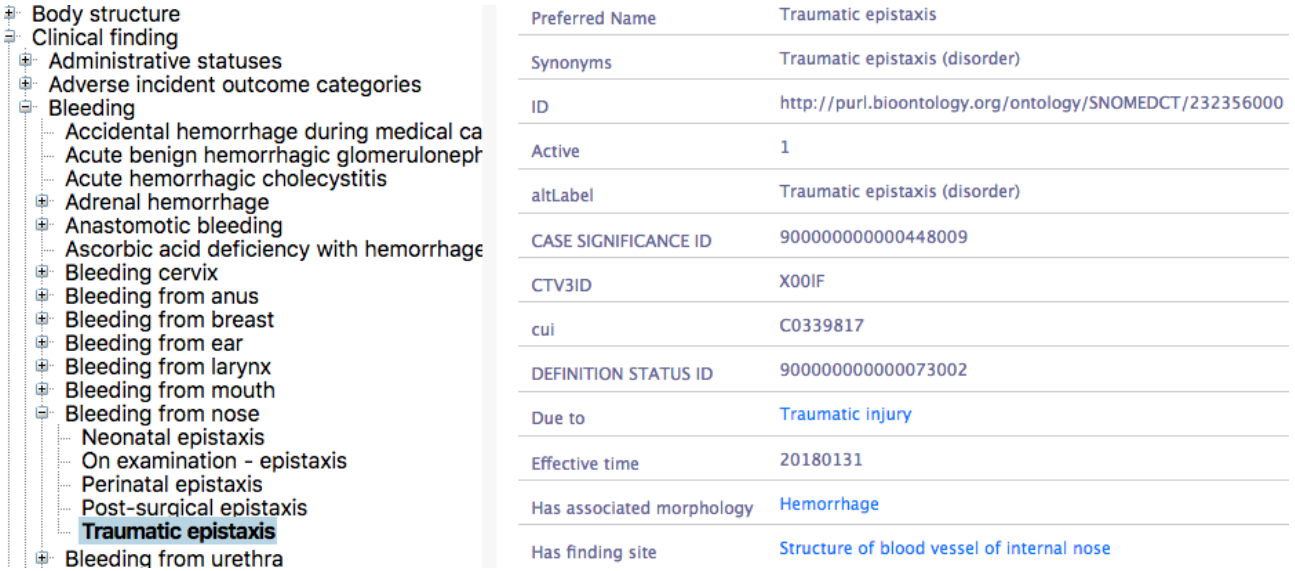
- Collaboration between Sapienza Univ. & Italian Department of Treasury on ontology-based data management of public debt
- CIDOC CRM (Comité International pour la DOCumentation Conceptual Reference Model): ontology for concepts and information in cultural heritage and museum documentation



Part of CIDOC CRM Class Hierarchy from www.cidoc-crm.org

Examples of Applications of Ontologies

Medical Ontologies



- SNOMED CT: general medical ontology ($>$ 350 000 concepts)
  - multilingual, mapped to other international standards
  - used for recording medical information : information sharing, decision-making assistance systems, gathering data for clinical research, monitoring population health and clinical practices...
- NCI (National Cancer Institute Thesaurus), FMA (Foundational Model of Anatomy), GO (Gene Ontology) ...

Examples of Applications of Ontologies

Ontologies for Life Sciences

- Bioportal repository contains hundreds of ontologies about biology and chemistry (`http://bioportal.bioontology.org/`)