Semantic Data

Chapter 4 : Description logics

Jean-Louis Binot

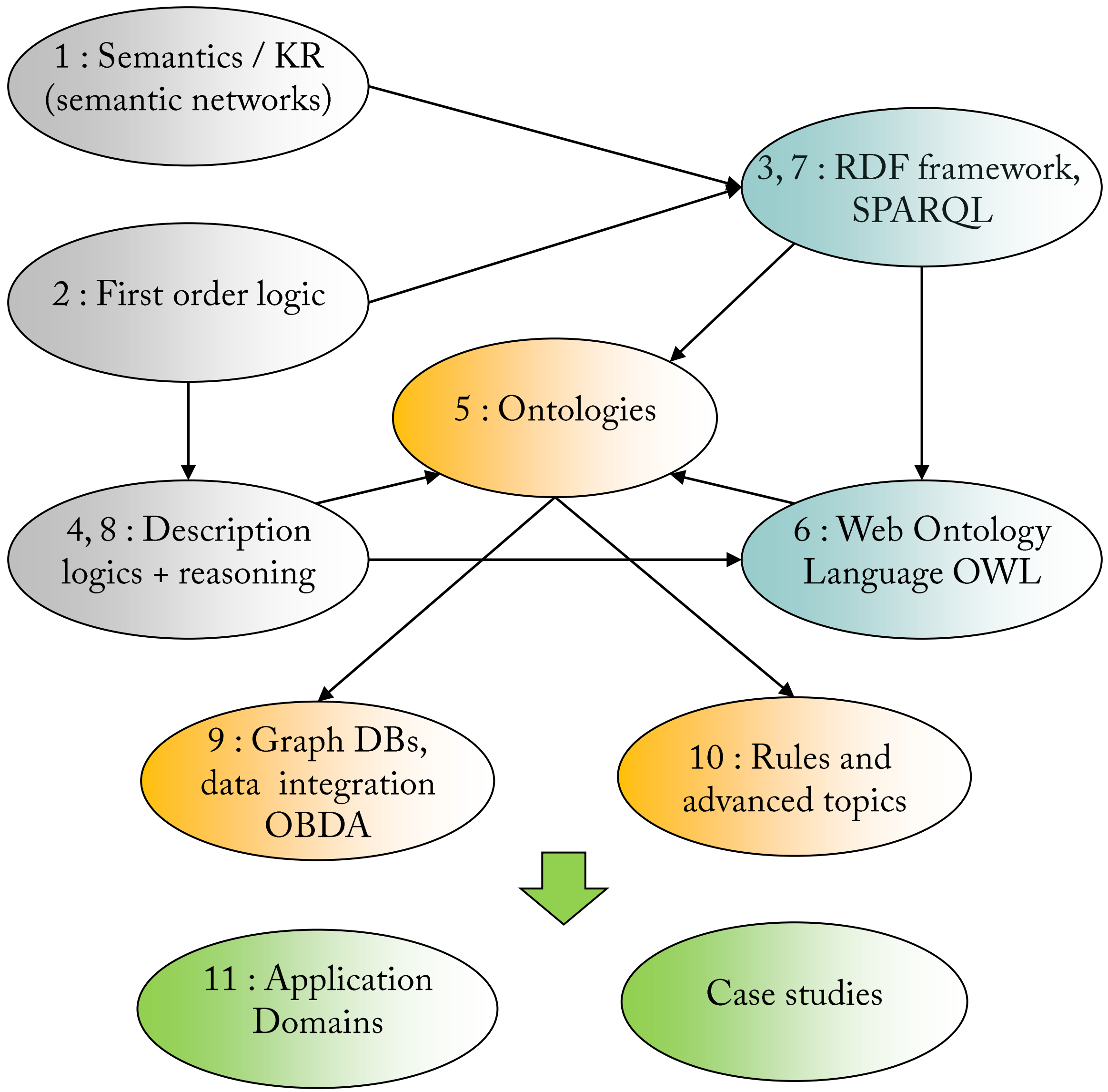
Course content outline

Credits : 5 (theory 25 h, practice 10 h, project 45 h)

Theory (25 h):

1. Semantics and knowledge representation.
2. Introduction to first order logic.
3. The semantic web resource description framework.
4. Description logics.
5. Ontologies and ontology engineering.
6. The Web Ontology Language : OWL.
7. Querying the semantic web : SPARQL.
8. Reasoning with description logics.
9. Data integration and ontology-based data access.
10. Rules and advanced topics.
11. Application domains for semantic data.

Case studies : real cases for genuine business customers; integrated in the relevant theory sessions.



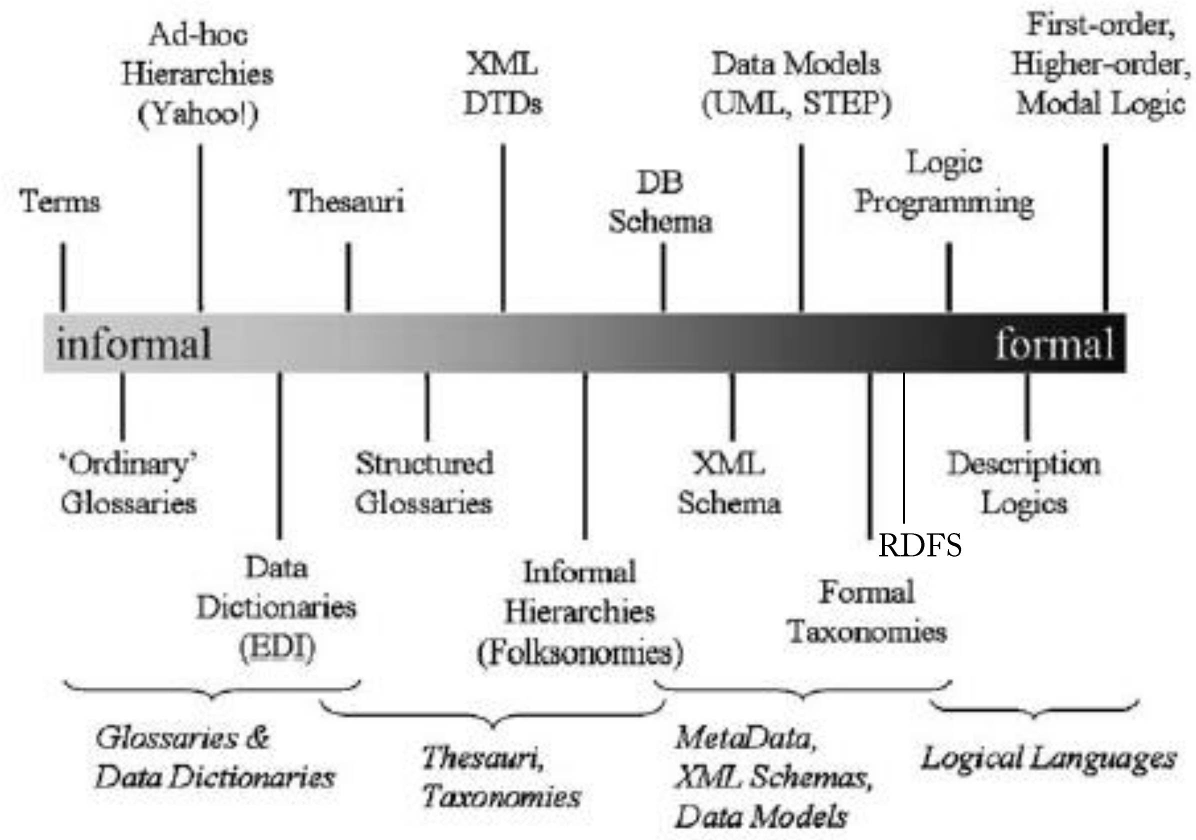
Sources and recommended readings

- There are no additional required references for this chapter.
- Sources and useful additional readings :
  - *Basic description logic* (*Baader and Nutt 2003)* : good starting point for description logic architecture.
  - *An introduction to description logic (Baader et al. 2017)* : comprehensive overview based on logic $\mathcal{ALC}$.
  - *A description logic primer (Krotzsch et al. 2013)* is based on the logic underlying OWL2 ($\mathcal{SROIQ}$).
  - *Foundations of Description Logics (Rudolph 2011)*.
- University courses having partially inspired ideas and examples for this chapter :
  - *Description Logics, a formal foundation for ontology languages and tools, I. Horrocks, Oxford University.*
  - *Introduction aux Logiques de Description, B. Espinasse, Aix-Marseilles University.*
  - *Description Logic, L. Karlsen, University Oslo.*
  - *The Description Logics* $\mathcal{EL}$*, F. Wolter, University of Liverpool.*
  - *Reasoning and Query Answering in Description Logics, Ortiz and Simkus, 8*th *Reasoning Web Summer School, 2012.*

Agenda

1. In search for the right language
2. Architecture and terminology
3. Basic description logic $\mathcal{ALC}$
4. Description logic knowledge bases
5. Extending description logics

A spectrum of languages to specify the meaning of terms



(from Guarino et al. 2009; RDFS added).

- At one extreme : informal lightweight solutions : terms only, with little or no specification of meaning.
- At the other extreme : rigorous formalized logical theories.
- The amount of meaning specified, and the degree of formality, increase along the continuum :
  - Ambiguity is reduced;
  - Support for automated reasoning is increased.

Requirements for ontology languages

Requirements :

1. A well-defined syntax;
2. Well-defined semantics;
3. Efficient reasoning support;
4. Sufficient expressive power;
5. Convenience of expression.

(after Antoniou and Van Harmelen 2004)

Comments

1. Syntax.
   - Easy use both for write and read.
   - Serialization variants (syntactic sugar, terse syntaxes) & development tools help addressing this problem.
2. Semantics.
   - A prerequisite for reasoning support.
   - Supports automatic classification of classes and instances.
   - Supports consistency checking of knowledge an detection of unintended relationships.

Requirements for ontology languages

Requirements :

1. A well-defined syntax;
2. Well-defined semantics;
3. Efficient reasoning support;
4. Sufficient expressive power;
5. Convenience of expression.

Comments

3. Efficient reasoning support.
   - Fast inferences (for knowledge-based systems)
   - Integration and sharing of ontologies from various sources.

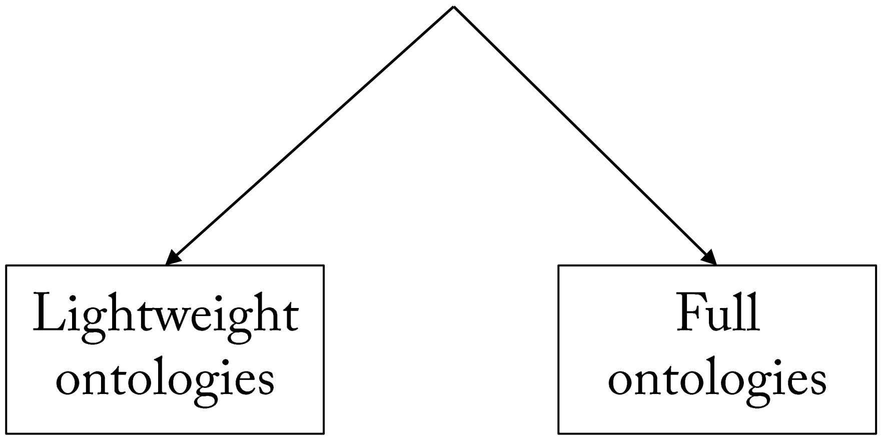
4. Tradeoff expressive power <=> reasoning efficiency.

   Reasoning efficiency : decidability/complexity of the reasoning algorithms.

   Expressive power : capability to express the concepts and models we need.

   - The more expressive a logic is, the more computationally expensive it is to draw conclusions.
   - Some may even become undecidable (cf. first order logic).

Trade-off expressive power vs reasoning support

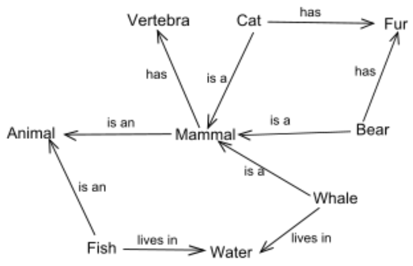


- ❑ Lightweight ontologies
  - Relationships, simple classification.
  - Suitable for terminologies and large scale performance.
- ❑ Heavy or full ontologies
  - Full axiomatization of domain semantics.
  - Suitable for inferences and reasoning.

Formal languages reviewed so far :

- ❑ Propositional logic :
  - Not expressive enough.
- ❑ First order logic :
  - Too expressive, and not decidable.
- ❑ RDF :
  - Lacks specific tools to represent ontologies.
  - Suitable for basic linked data integration.
- ❑ RDFS :
  - Offer only basic tools for taxonomy representation.
  - Suitable for lightweight ontologies.
- ❑ For full ontologies ? Description logics.

A brief history of description logics



(source for image: wikipedia)

Semantic networks

Graph-based inheritance.
Semantics and inference issues.
"Semantic networks have no semantics."

First order logic

$(\forall x\ Human(x) \rightarrow Animal(x))$

Logical deduction; formal semantics.
Complexity and representation issues.
"Computing over all possible worlds is no joke."

→ **Description logics**

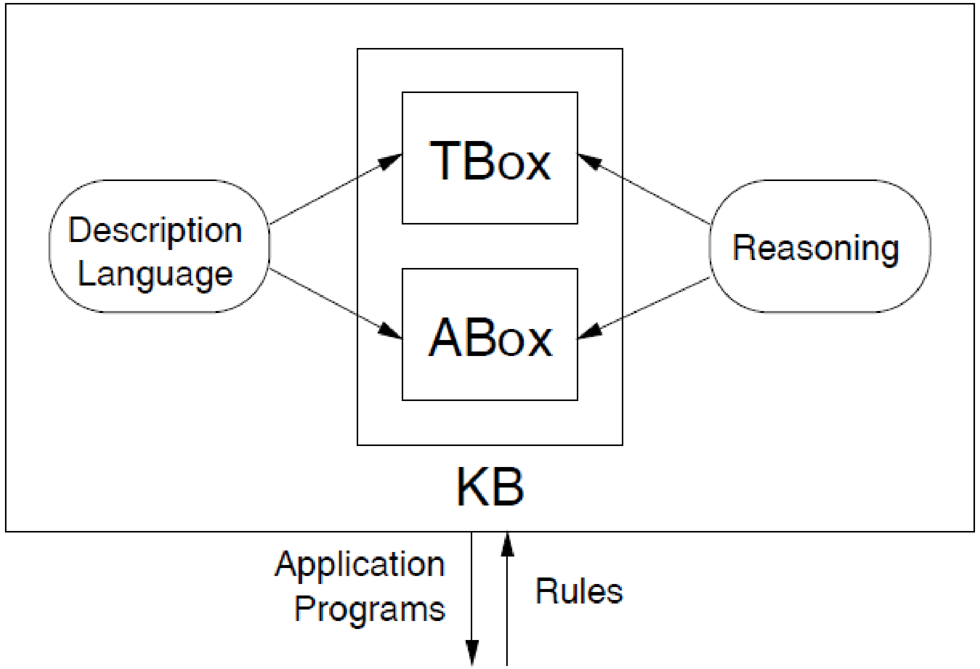
A merge between inheritance concepts of semantic networks and formal semantics of first order logic :

- 1965-80s : Semantic networks and frames are challenged.
- 1980s-90s : Structured inheritance networks attempt to bring formal semantics to semantic networks, using a fragment of first order logic (KL-ONE).
- 1990-now : description logics emerge, with more and more performing tableau-based reasoning algorithms.

Agenda

1	Origins and motivation
2	Architecture and terminology
3	Basic description logic $\mathcal{ALC}$
4	Description logic knowledge bases
5	Extending description logics

Architecture of a description logic knowledge base

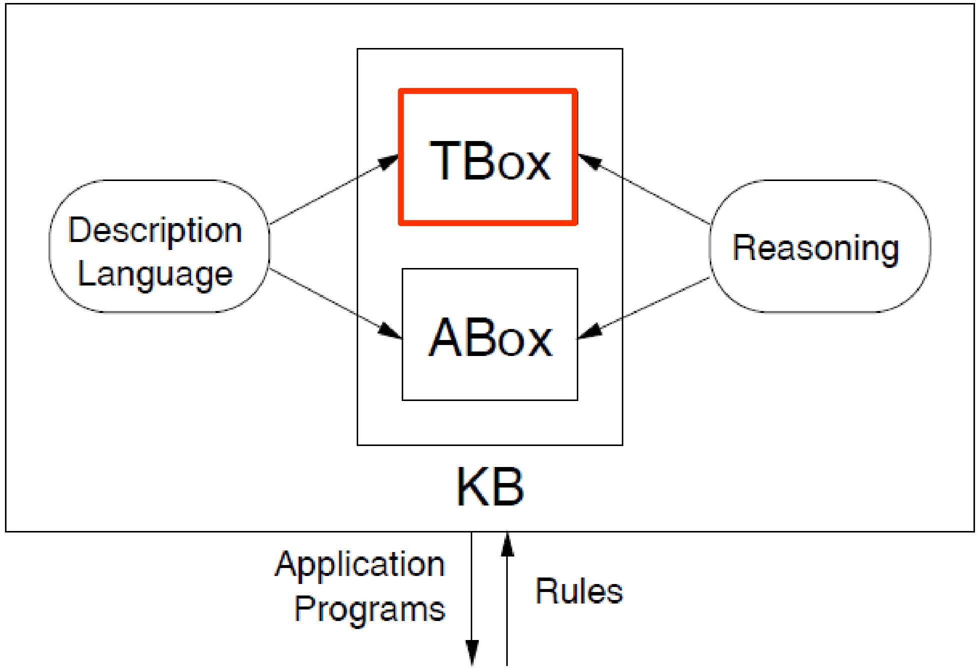


(after Baader and Nutt 2003)

A description logic knowledge base is typically made of the following components :

- The TBox: contains the Terminology of the domain.
- The ABox: contains Assertions about individuals.
- The reasoning services support the inferences available in the DL system.
- Rules provide an (optional) component to add assertions to the logical system.

DL architecture : the TBox



Tbox : contains the Terminology of the domain :

- Concepts (sets of individuals).
- Roles (binary relationships / properties).
- Complex descriptions of concepts and roles, which can be named.
- Terminological axioms (inclusion, equivalence).

Examples :

Person ⊓ Female : concept description.

Parent ⊓ ∀hasChild.(Doctor ⊔ ∃hasChild.Doctor) : complex description.

Woman ≡ Person ⊓ Female : equivalence axiom.

Woman ⊆ Person : inclusion axiom.

Concept descriptions

At the heart of a description logic is a concept description language, using :

- Atomic concepts : Person
- Atomic roles (binary properties) : hasChild, loves…
- Complex expressions combining concepts using set operators :

  $\neg$ (negation or complement), $\sqcap$ (conjunction), $\sqcup$ (disjunction)…

  They express the corresponding operations on the extensions (instance sets) of the concepts.

Person $\sqcap$ $\neg$Female	*a concept expression defining a man*.
Father $\sqcup$ Mother	*a concept expression defining a parent*.

Continued on next slide…

Concept descriptions ./.

- Complex expressions combining concepts with roles :

  - Existential restriction :

    $\exists r.C$ expresses that there exists an individual of class C filling role r.

    $\text{Woman} \sqcap \exists \text{hasChild}.\text{Person}$ *a description of the concept of mother*.

  - Value restriction :

    $\forall r.C$ expresses that all fillers of role r must be of class C.

    $\text{Person} \sqcap \forall \text{hasChild}.\neg \text{Woman}$ *a description of the concept of person without daughter*.

    Note: the definition in last example also includes persons without any children !

Description logic notation is variable-free !

TBox concept equivalence axioms

- Equivalence axiom : $C \equiv D$, where C and D are concept expressions.
  - Concepts C and D have the same extension.
- When C is an atomic concept name, this axiom is called a concept definition.
  - It defines and names a new concept in terms of other previously defined concepts :

  Woman ≡ Person ⊓ Female

  Mother ≡ Woman ⊓ ∃hasChild.Person

  - It is used to build the terminology of the domain.
- Equivalence axioms may include complex concept descriptions on both sides.

  ∃teaches.Course ≡ ¬Student

TBox concept inclusion axioms

- Concept inclusion axiom : $C \sqsubseteq D$, where C and D are concept expressions.
  - All instances of C are also instances of D.

- When C is an atomic concept, this axiom is called a specialization.
  - It is used to build the generalization/specialization hierarchy.

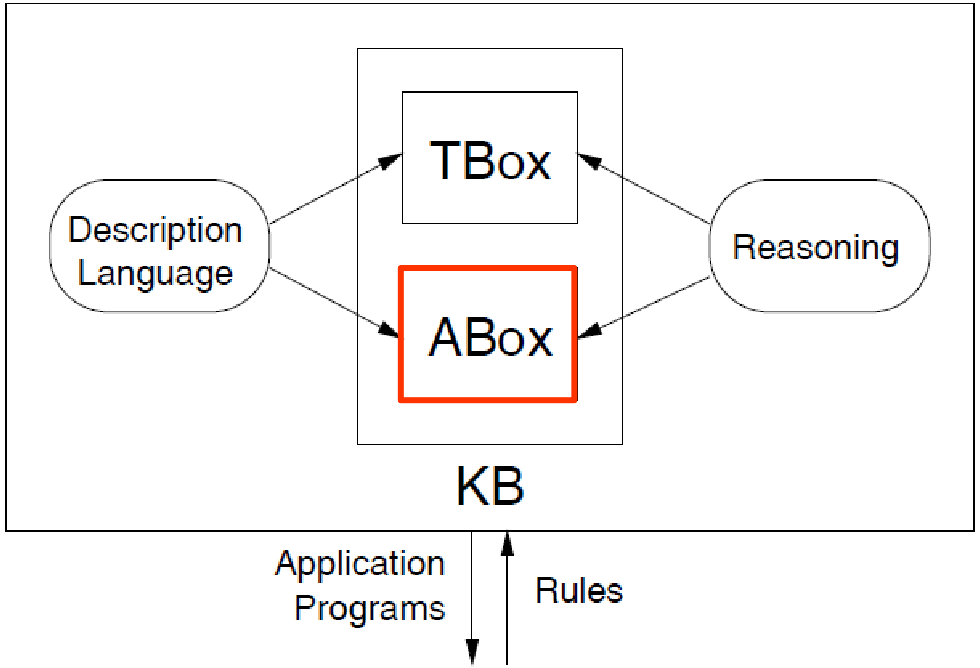
    $\text{Woman} \sqsubseteq \text{Person}$ *Woman is a subclass of Person*, or *Person subsumes Woman*.

- General concept inclusion (GCI) : both C and D can be complex concepts expressions.

    $\exists \text{teaches.Course} \sqsubseteq \text{Person}$

    $\text{Teacher} \sqsubseteq \text{Person} \sqcap \exists \text{teaches.Course}$

DL architecture : the ABox



- ABox: contains Assertions about individuals :
  - Concept assertion : axiom of the form C(a) or a : C (*)

    states that individual a belongs to class C.
  - Role assertion : axiom of the form r(b, c) or (b, c) : r

    states that individual c is a filler of role r for individual b.
- Describes a specific state of affairs in terms of the TBox vocabulary.

Examples :

Father(Peter) : concept assertion.

hasChild(Peter, Harry) : role assertion.

Michel : ∃own.Ferrari : complex role assertion (Michel is a Ferrari owner).

* : the ":" notation makes more readable the use of complex descriptions without name.

Description logic knowledge base : an example

KB

TBOX

Woman ≡ Person ⊓ Female

Man ≡ Person ⊓ ¬Woman

Mother ≡ Woman ⊓ ∃hasChild.Person

Father ≡ Man ⊓ ∃hasChild.Person

Parent ≡ Father ⊔ Mother

Grandmother ≡ Mother ⊓ ∃hasChild.Parent

MotherWithoutDaughter ≡ Mother ⊓ ∀hasChild.¬Woman

Wife ≡ Woman ⊓ ∃hasHusband.Man

ABOX

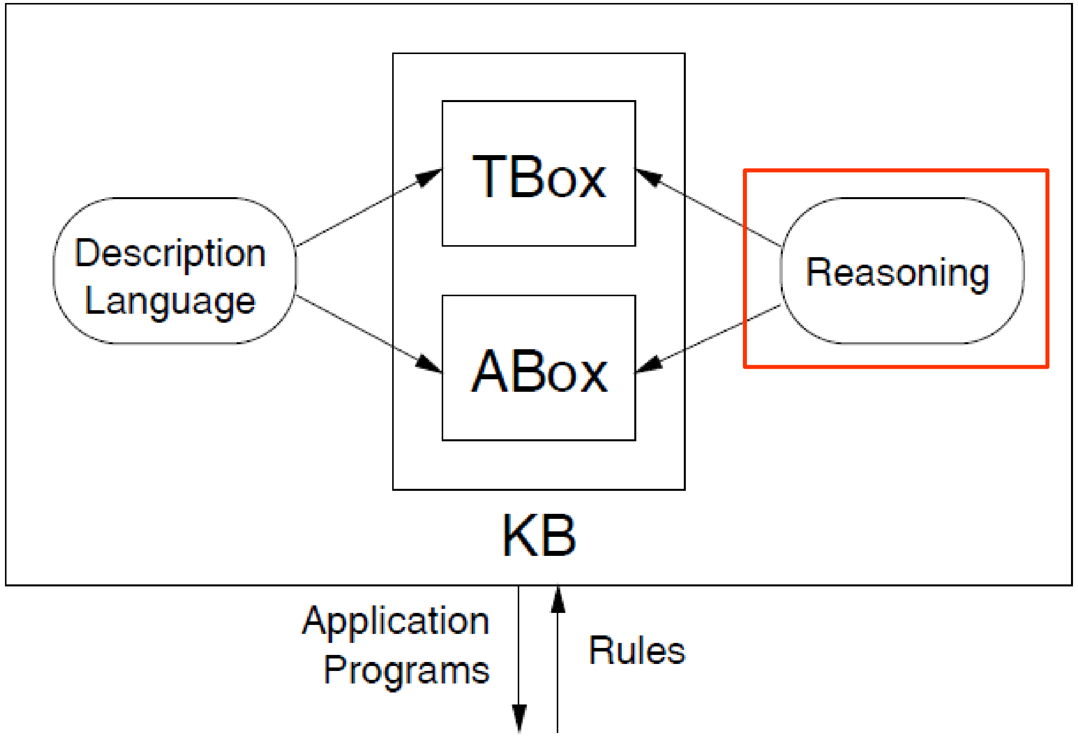
MotherWithoutDaughter(Mary)

Father(Peter)

hasChild(Mary, Paul)

(from Baader and Nutt 2003)

DL architecture : reasoning services



- Reasoning services (inferences) in a DL system :
  - Is a description satisfiable (not contradictory) ?
  - Is a description subsuming another ?
  - Is a set of assertions consistent (does it have a model) ?
  - Is an instance relationship entailed by the ABox ?
- Rules : restricted mechanism to add assertions.

They will be covered in later chapters.

Agenda

1	Origins and motivation
2	Architecture and terminology
3	Basic description logic $\mathcal{ALC}$
4	Description logic knowledge bases
5	Extending description logics

A family of concept description languages

- Description logics : not one language but a family of logical languages.
  - Driving concern: still the search for a trade-off between expressive power and reasoning efficiency.
- Each DL differs in this trade-off by the constructs of the concept description language.
- We will see the basic description logic $\mathcal{ALC}$: Attributive Language with Complement.
  - Attributive : using attributes (roles).
  - Complement : (set) negation.

$\mathcal{ALC}$–concept description language : syntax

- $\mathcal{ALC}$-concept descriptions can be defined recursively by the following rules :

If Nc is a set of concept names and Nr a set of role names :

- A concept name A is an atomic concept description, ∀A ∈ Nc;
- ⊤ (universal concept) and ⊥ (bottom concept) are concept descriptions;
- If C and D are concept descriptions and r ∈ Nr a role, the following formulas are also concept descriptions:

$\neg C$	(negation);
$C \sqcap D$	(intersection);
$C \sqcup D$	(union);
$\exists r.C$	(existential restriction);
$\forall r.C$	(value restriction).

$\mathcal{ALC}$–concepts semantics : intuitive understanding

C : denotes the individuals who belong to concept C.

¬C : denotes those individuals who do not belong to concept C.

C ⊓ D : denotes those individuals belonging both to C and D.

Person ⊓ ¬Female — non female persons (males).

C ⊔ D : denotes those individuals belonging either to C or to D.

Father ⊔ Mother — fathers or mothers (parents).

∃r.C : at least one individual belonging to concept C is in relation r with the concept described.

∃hasChild.Female — individuals having at least one female child.

∀r.C : all individuals in relation r with the concept described must belong to concept C.

∀hasChild.Female — individuals whose all children are female.

⊤ and ⊥

- ⊤ is the universal concept, also named top, or Thing.
  - Every individual of the interpretation domain belongs to ⊤.

  Person ⊓ ∃hasChild.⊤ persons with at least one (uncategorized) child.

- ⊥ is the inconsistent concept, also named bottom, or Nothing
  - No individual of the interpretation domain belongs to ⊥.

  Person ⊓ ∀hasChild.⊥ persons without children.

Role restrictions, domain and range

Role domain and range restrictions are expressed using existential and value restrictions with $\top$ and $\top$:

- **Domain restriction :**

  $\exists \text{sonOf}.\top \subseteq \text{Male}$ restricts the domain of sonOf to male individuals.

- **Range description :**

  $\top \subseteq \forall \text{sonOf}.\text{Parent}$ restricts the range of sonOf to parents.

$\mathcal{ALC}$ semantics: interpretation

- The formal definition of $\mathcal{ALC}$ semantics follows again the model-theoretic approach :

  We describe how an interpretation or model maps expressions of the language to the real world.

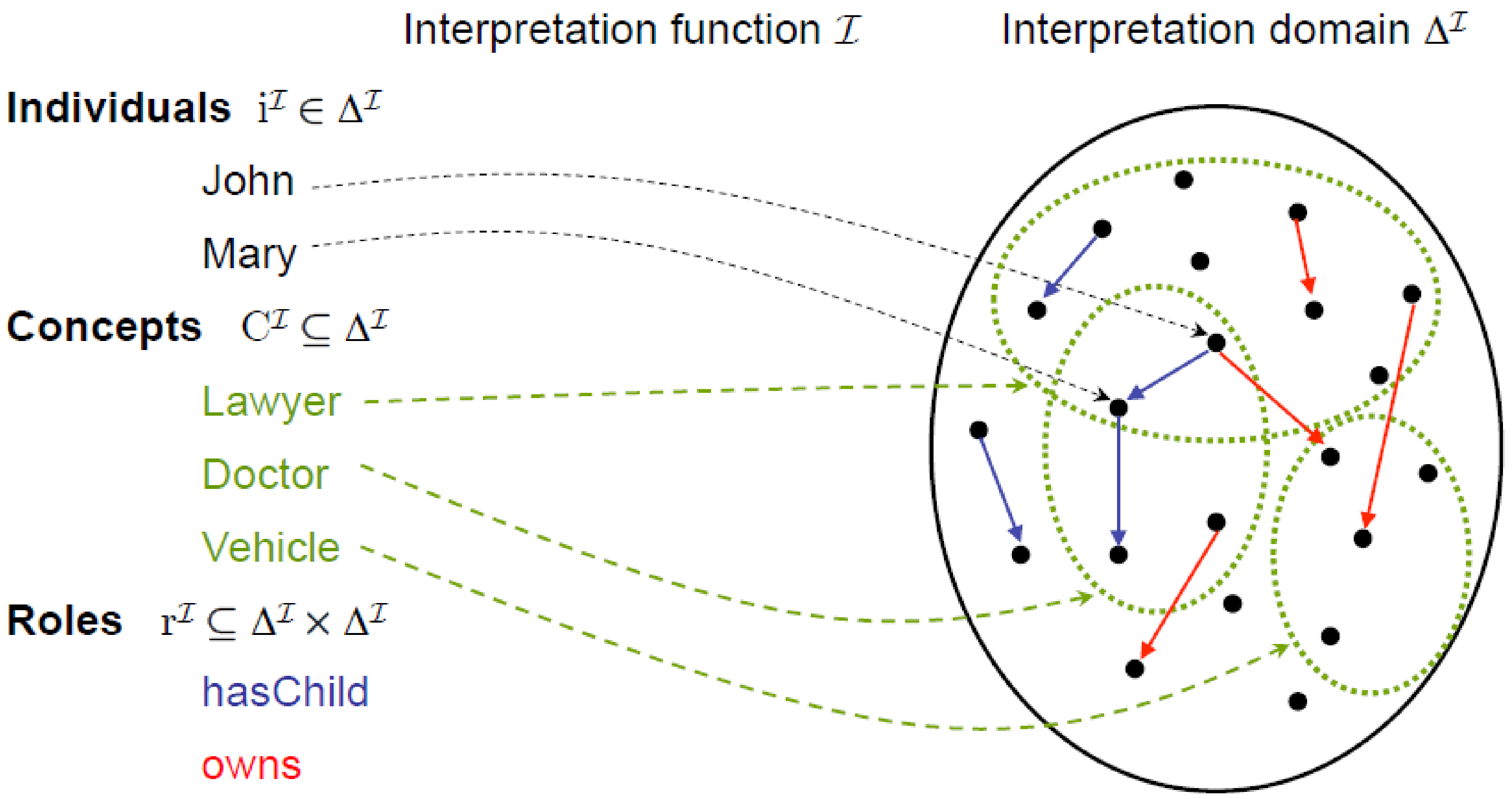
As in FOL we will use an interpretation domain and an interpretation function :

- The interpretation domain $\Delta^{\mathcal{I}}$ is the nonempty set of objects referred by an interpretation.

- The interpretation function $.^{\mathcal{I}}$ assigns an interpretation to every construct of the language :
  - Every atomic $\mathcal{ALC}$-concept A to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$,
  - Every role name r to a subset $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$,
  - Every individual name a to an element $a^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ (*).

* : Interpretations of individuals are not needed for concept descriptions, but for the ABox axioms .

DL Semantics: example



(From Description Logic, a formal foundation for ontology languages and tools, I. Horrocks, Oxford University)

$\mathcal{ALC}$-concepts semantics

- The interpretation function $\cdot^{\mathcal{I}}$ is defined recursively.

  For all $\mathcal{ALC}$-concepts C, D and all role names r :

  - $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$,
  - $\bot^{\mathcal{I}} = \emptyset$,
  - $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$,
  - $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$,
  - $\neg C^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$,
  - $(\exists r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \text{there is some } y \in \Delta^{\mathcal{I}} \text{ with } \langle x, y\rangle \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$,
  - $(\forall r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \text{for all } y \in \Delta^{\mathcal{I}}, \text{ if } \langle x, y\rangle \in r^{\mathcal{I}}, \text{ then } y \in C^{\mathcal{I}}\}$.

- $C^{\mathcal{I}}$ (resp. $r^{\mathcal{I}}$) is the extension of the concept C (resp. role name r) in the interpretation $\mathcal{I}$.
- If $x \in C^{\mathcal{I}}$, then we say that x is an instance of C in $\mathcal{I}$.

Agenda

1. Origins and motivation
2. Architecture and terminology
3. Basic description logic $\mathcal{ALC}$
4. Description logic knowledge bases
5. Extending description logics

Semantics of terminological axioms

The following definitions apply to any description logic, not only to $\mathcal{ALC}$.

If C and D are concept expressions, and A an atomic concept name :

- An interpretation $\mathcal{I}$ satisfies a general concept inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$,
  - We will write $\mathcal{I} \vDash C \sqsubseteq D$ (same as for first order logic).
- An interpretation $\mathcal{I}$ satisfies a concept definition $A \equiv C$ if $A^{\mathcal{I}} = C^{\mathcal{I}}$.
  - $C \equiv D$ can be seen as a shorthand notation for the two GCIs $C \sqsubseteq D$ and $D \sqsubseteq C$.
- A TBox is a finite set of GCIs.

  An interpretation $\mathcal{I}$ satisfies a TBox $\mathcal{T}$, noted $\mathcal{I} \vDash \mathcal{T}$, if $\mathcal{I}$ is a model of every GCI in $\mathcal{T}$.

TBoxes and terminologies

- A TBox $\mathcal{T}$ is called a terminology if :
  - It contains only concept definitions of the form $A \equiv C$;
  - Every atomic concept A has only one definition.
- Concepts appearing on the left hand side of a definition are called defined concepts.
- Concepts which are not defined are called primitive concepts.
- A terminology is acyclic if no definition refers to the atomic concept defined.

  Example of a cyclic definition : $\text{Human} \equiv \text{Animal} \sqcap \forall \text{hasParent.Human}$.
- Acyclic terminologies support simplified reasoning techniques, of lower complexity.

Terminology quiz for $\mathcal{ALC}$

Is this an acyclic $\mathcal{ALC}$ terminology ?

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

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

$\text{Mother} \equiv \text{Woman} \sqcap \exists\text{hasChild.Person}$

$\text{Father} \equiv \text{Man} \sqcap \exists\text{hasChild.Person}$

$\text{Parent} \equiv \text{Father} \sqcup \text{Mother}$

$\text{Grandmother} \equiv \text{Mother} \sqcap \exists\text{hasChild.Parent}$

$\text{MotherWithoutDaughter} \equiv \text{Mother} \sqcap \forall\text{hasChild.}\neg\text{Woman}$

$\text{Wife} \equiv \text{Woman} \sqcap \exists\text{hasHusband.Man}$

Answer: yes.

(source Baader and Nutt 2003)

Semantics of assertional axioms

The ABox contains assertion axioms about individuals.

- An interpretation $\mathcal{I}$ satisfies a concept assertion $C(a)$ or $a : C$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$.
- An interpretation $\mathcal{I}$ satisfies a role assertion $r(a, b)$ or $(a, b) : r$ if $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$ (*) .
- An ABox is a finite set of assertional axioms.

  An interpretation $\mathcal{I}$ satisfies an ABox $\mathcal{A}$, noted $\mathcal{I} \models \mathcal{A}$, if $\mathcal{I}$ is a model of every axiom in $\mathcal{A}$.

Description logics knowledge bases

A knowledge base $\mathcal{KB}$ is a pair $<\mathcal{T}, \mathcal{A}>$ where is $\mathcal{T}$ a TBox and $\mathcal{A}$ is an ABox.

- An interpretation $\mathcal{I}$ satisfies a knowledge base $\mathcal{KB}$, written $\mathcal{I} \vDash \mathcal{KB}$, iff

  $\mathcal{I}$ is a model of $\mathcal{T}$ and $\mathcal{I}$ is also a model of $\mathcal{A}$.

As in first order logic, we will say that :

- A knowledge base $\mathcal{KB}$ is called satisfiable or consistent if it has at least one model.
- It is called unsatisfiable or inconsistent otherwise.
- A knowledge base $\mathcal{KB}$ entails an axiom ax, written $\mathcal{KB} \vDash$ ax, iff every model of $\mathcal{KB}$ satisfies also ax.

Quiz

$\mathcal{T}$ = {*Doctor* ⊑ *Person*, *Parent* ≡ *Person* ⊓ ∃*hasChild*.*Person*,
HappyParent ≡ *Parent* ⊓ ∀*hasChild*.(*Doctor* ⊔ ∃*hasChild*.*Doctor*)}

$\mathcal{A}$ = {*John* : *HappyParent*, (*John*, *Mary*) : *hasChild*, (*John*, *Sally*) : *hasChild*,
Mary : *Person* ⊓ ¬*Doctor*, (*Mary*, *Peter*) : *hasChild*, *Mary* : (≤ 1 *hasChild*)}

$\mathcal{K}$ ⊨ John:Person ? ✓

$\mathcal{K}$ ⊨ Peter:Doctor ? ✓

$\mathcal{K}$ ⊨ Mary:HappyParent ? ✓

What if we add (*Mary*, *Jane*) : *hasChild* ? (≤ *1 hasChild* means that there is at most 1 value for the role)

$\mathcal{K}$ ⊨ Peter = Jane

What if we add *HappyPerson* ≡ *Person* ⊓ ∃ *hasChild*.*Doctor* ?

$\mathcal{K}$ ⊨ HappyPerson ⊆ Parent

(From Description Logic, a formal foundation for ontology languages and tools, Horrocks, Oxford University)

Mapping description logics to FOL

Description languages are fragments of FOL and can be mapped to it.

Defining that mapping is another way to precise the semantics of DLs.
Basic principles :

- As already noted, the DL notation is variable free.
- Any concept C can be transformed into a FOL formula with one free variable x :

  Person ⊓ Female is equivalent to Person(x) ∧ Female(x).

- Using the concept description in an axiom has the effect of binding the free variable :

  Woman ≡ Person ⊓ Female is equivalent to ∀x (Woman(x) ↔ Person(x) ∧ Female(x))

- ABox axioms are equivalent to FOL formulas without variables :

  (Mary, Paul) : hasChild is equivalent to hasChild(Mary, Paul)

$\mathcal{ALC}$–concepts : mapping to FOL

The function π_x maps concepts to FOL formulae with one free variable x.

π_x is defined recursively as follows :

If A is an atomic concept name, r a role name and C and D concept descriptions, then :

- $\pi_x(A) = A(x)$
- $\pi_x(\neg C) = \neg\pi_x(C)$
- $\pi_x(C \sqcup D) = \pi_x(C) \vee \pi_x(D)$
- $\pi_x(C \sqcap D) = \pi_x(C) \wedge \pi_x(D)$
- $\pi_x(\exists r.C) = \exists y\ (r(x, y) \wedge \pi_y(C))$
- $\pi_x(\forall r.C) = \forall y\ (r(x, y) \rightarrow \pi_y(C))$

(*after Baader et al. 2003*)

DL knowledge base : mapping into FOL

The function π maps axioms to FOL formulae.

The TBox is a conjunction of axioms; each can be mapped using the following rules :

- $\pi(\{C \sqsubseteq D\}) = \forall x \ (\pi_x(C) \rightarrow \pi_x(D))$ (*)
- $\pi(\{C \equiv D\}) = \forall x \ (\pi_x(C) \leftrightarrow \pi_x(D))$

The ABox is a conjunction of axioms; each can be mapped using the following rules :

- $\pi(\{a : C\}) = \pi_x(C)_{[x/a]}$

  (where $F_{[x/a]}$ denotes the formula obtained from F by replacing all free occurrences of x with a).
- $\pi(\{(a, b) : r\}) = r(a, b)$

* : axioms are noted between {}

Consequences of translation rules into FOL

- The translation from DL to FOL preserves the model-theoretic semantics :

$$C^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \mathcal{I} \models \pi_x(C)_{[x/d]}\}$$

We can view DL interpretations as first order interpretations and vice versa.

- As a consequence, reasoning in $\mathcal{ALC}$ corresponds to first-order inference.
- These rules allow us to rely on known decidability results concerning FOL.

Decidability of description logics

- FOL is undecidable, but fortunately, some fragments of FOL are decidable !

- One of them is the two variables fragment $\mathcal{L}^2$ (sentences with only two variables).
  - Decidability of $\mathcal{L}^2$ has been proven.
  - Any $\mathcal{ALC}$-concept can be rewritten in a FOL formula with only two variables (using variable renaming).

  Example : $\forall r.(\exists r.A)$ translates first into $\forall y\ (r(x, y) \rightarrow \exists z\ (r(y, z) \wedge A(z)))$

  As $\exists z\ (r(y, z) \wedge A(z))$ does not contain x, the bound variable z is renamed into x : $\forall y\ (r(x, y) \rightarrow \exists x\ (r(y, x) \wedge A(x)))$

- $\mathcal{ALC}$ is decidable, as is any extension of $\mathcal{ALC}$ which can be mapped into FOL $\mathcal{L}^2$ fragment.
  - Note: number restrictions (seen next) cannot be mapped into FOL $\mathcal{L}^2$ fragment.

    However, the logic with two variables and counting quantifiers $\mathcal{C}^2$ has also be proven decidable !

Open World and non unique names

- Description logics make the open world assumption (OWA) :
  - Missing information is treated as unknown.
- Description logics usually do not make the unique name assumption (No UNA) :
  - Individuals may have several names.
- Example

HarryPotter : Wizard
DracoMalfoy : Wizard
hasFriend(HarryPotter, RonWeasley)
hasFriend(HarryPotter, HermioneGranger)
hasPet(HarryPotter, Hedwig)

Is DracoMalfoy a friend of HarryPotter ?

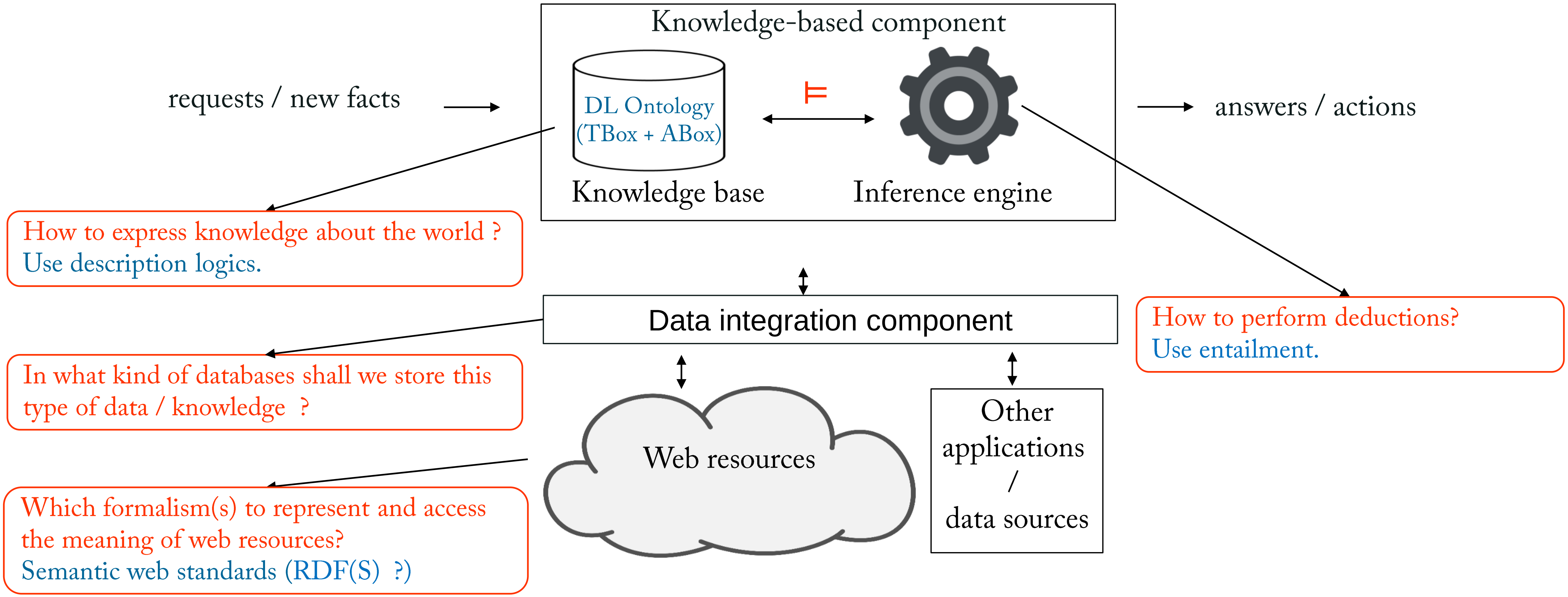
CWA : No. OWA : don't know.

How many friends does HarryPotter have ?

UNA : 2. No UNA : at least 1.

(From Description Logic, a formal foundation for ontology languages and tools, I. Horrocks, Oxford University)

Updated framework of reference for semantic applications



Agenda

1 Origins and motivation

2 Architecture and terminology

3 Basic description logic $\mathcal{ALC}$

4 Description logic knowledge bases

5 Family of description logics

Reminder of some useful logical equivalences

$C1 \sqcup C2 \equiv \neg$?	$C1 \sqcup C2 \equiv \neg(\neg C1 \sqcap \neg C2)$
$\exists r.C \equiv \neg$?	$\exists r.C \equiv \neg(\forall r.\neg C)$
$A \sqcup \neg A \equiv$?	$A \sqcup \neg A \equiv \top$
$A \sqcap \neg A \equiv$?	$A \sqcap \neg A \equiv \bot$

- Negation and intersection give us union.
- Negation and value restriction give us (full) existential restriction.

Basic description logic $\mathcal{AL}$ and naming scheme

- $\mathcal{ALC}$ is an extension of the basic description logic $\mathcal{AL}$.
- $\mathcal{AL}$ includes:
  - Intersection: $\sqcap$
  - Value restriction: $\forall r.C$
  - Limited existential restriction: $\exists r.\top$ (no type restriction associated to the role filler)
- To each construct is associated a letter in the naming scheme :
  - $\mathcal{C}$ stands for Negation (Complement): $\neg$
  - $\mathcal{U}$ stands for Union: $\sqcup$
  - $\mathcal{E}$ stands for full existential restriction: $\exists r.C$
- As a result $\mathcal{ALC}$ is equivalent to $\mathcal{ALUEC}$.

Extensions of the basic description logic $\mathcal{ALC}$

Extensions of the basic description logic are built by adding various syntactic constructs designed to express interesting knowledge elements.

We describe some of the most important ones.

Each extension may impact the complexity of the reasoning algorithms and even the decidability of the resulting logic.

Their combination must be selected with care by the language designers (more on that in next chapters).

Number restrictions

- Unqualified number restrictions, denoted by $\mathcal{N}$, are written as :

  $\geq n\ r$ (at-least restriction) and as $\leq n\ r$ (at-most restriction), where n ranges over nonnegative integers.

  Persons that have either not more than one child or at least three children, one of which is female.

  Person ⊓ (≤1 hasChild ⊔ (≥3 hasChild ⊓ ∃hasChild.Female))

- Qualified number restrictions, denoted by $\mathcal{Q}$, are written as :

  $\geq n\ r.C$ (at-least restriction) and as $\leq n\ r.C$ (at-most restriction), where n ranges over nonnegative integers.

  Persons that have at least three female children.

  Person ⊓ ≥ 3 hasChild.Female

Nominals

- Nominals, denoted by $\mathcal{O}$ in the naming scheme, allow us to build a concept from a set of individuals $\{a_1, a_2, \ldots, a_n\}$ or from a singleton $\{a\}$.
- Example :

The computer scientists who have met Turing.

$\text{CScientist} \sqcap \exists \text{hasMet}.\{\text{Turing}\}$

Role constructor extensions

- Functional roles, denoted by $\mathcal{F}$
  - The role behaves as a function. Can be expressed by number restrictions : $\top \sqsubseteq (\leq 1 \; \mathit{hasMother})$
- Inverse roles, denoted by $\mathcal{I}$
  - The inverse role constructor $^{-}$ allows us to specify the inverse of a role: $\mathrm{hasParent} \equiv \mathrm{hasChild}^{-}$.
- Role Hierarchies, denoted by $\mathcal{H}$
  - A hierarchy of roles can be defined by role axioms of the form $r \sqsubseteq s$: $\mathrm{hasParent} \sqsubseteq \mathrm{hasAncestor}$.
  - Complex role hierarchies can be defined by role composition: $\mathit{hasParent} \circ \mathit{hasBrother} \sqsubseteq \mathit{hasUncle}$
- Transitive roles, denoted by $\mathcal{R}^{+}$
  - Roles may be declared as being transitive through a role axiom : Trans(is_part_of).
  - Or a complex role hierarchy declaration : is_part_of ○ is_part_of ⊆ is_part_of.

Transitive axioms have been motivated by the need to capture part_of relationships (cf. chapter 5).

Note : in presence of transitive roles, additional conditions must be enforced to ensure decidability.
For example number restrictions cannot be applied on transitive roles.

Standard abbreviations

- $\mathcal{S}$ denotes the usual combination of $\mathcal{ALC}$ extended with transitive roles : $\mathcal{S} = \mathcal{ALCR}^+$
- $\mathcal{SR}$ denotes $\mathcal{ALC}$ extended with various useful role axioms and properties :
  - Role hierarchies;
  - Role axioms : Reflexivity, Transitivity, Symmetry, Antisymmetry.
  - Self concepts, allowing a role to connect an individual to itself : Narcissist ≡ ∃loves.Self.
- Relationships
  - $\mathcal{SR}$ subsumes $\mathcal{ALC}$, $\mathcal{S}$ and $\mathcal{SH}$.
  - $\mathcal{F}$ becomes obsolete once $\mathcal{N}$ is present and both are superseded by $\mathcal{Q}$.
- The most expressive computable DL is $\mathcal{SROIQ}$, corresponding to OWL2.

Basic description logic $\mathcal{EL}$

- There are other basic description logics than $\mathcal{AL}$.
  - Basic description logics support efficient algorithms; they are suitable for large ontologies.
- The basic DL $\mathcal{EL}$ includes the concept constructors $\top$, $\sqcap$, and $\exists r.C$.
  - Large scale biomedical ontologies are written in a minor extension of $\mathcal{EL}$.
  - SNOMED CT, a very large scale comprehensive healthcare terminology with approximately 400000 definitions, is almost an acyclic $\mathcal{EL}$ terminology (except inclusions between role names) .

Examples of axioms

EntireFemur ⊆ StructureOfFemur

FemurPart ⊆ StructureOfFemur ⊓ ∃part_of.EntireFemur

DistalFemurPart ⊆ BoneStructureOfDistalFemur ⊓ ∃part_of.EntireDistalFemur

Pericarditis ⊆ Inflammation ⊓ ∃has_loc.Pericardium

Inflammation ⊆ Disease ⊓ ∃acts_on.Tissue

Extract of hierarchy

EntireDistalEpiphysisOfFemur
⊆ StructureOfDistalEpiphysisOfFemur
⊆ DistalFemurPart
⊆ BoneStructureOfDistalFemur
⊆ FemurPart

(example from: Description Logics $\mathcal{EL}$, F. Wolter, University of Liverpool)

Summary

- A description logic has a precise structure, built around a Terminology box (TBox), capturing the ontology of the domain and an Assertions box (ABox), capturing a specific state of the world.
- Concepts can be defined by complex concept expressions and terminology axioms, taking the most general form of general concept inclusions.
- Their structure and the use of a concept description language make them very suitable for representing ontologies.
- Description logics are a family of logics differing by the constructs used in the concept description language, impacting the trade-off between reasoning performance and expressive power.
- Most of them (including $\mathcal{ALC}$) can be mapped into decidable fragments of FOL with efficient reasoning algorithms (this last topic will be tackled in chapter 8).

Appendix : constructors

Construct	Syntax	Semantics	Symbol
Roles			
atomic role	r	$r^{\mathcal{I}}$	$\mathcal{AL}$
inverse role	r^{-}	$\{<x, y> \mid <y, x> \in r^{\mathcal{I}}\}$	$\mathcal{I}$
Concepts			
atomic concept	A	$A^{\mathcal{I}}$	$\mathcal{AL}$
top concept	$\top$	$\Delta^{\mathcal{I}}$	$\mathcal{AL}$
bottom concept	$\bot$	$\emptyset$	$\mathcal{AL}$
intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	$\mathcal{AL}$
union	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	$\mathcal{U}$
complement	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	$\mathcal{C}$
existential restriction	$\exists r.C$	$\{x \mid$ There is some $y \in \Delta^{\mathcal{I}}$ with $<x, y> \in r^{\mathcal{I}}$ and $y \in C^{\mathcal{I}}\}$	$\mathcal{E}$
universal restriction	$\forall r.C$	$\{x \mid$ For all $y \in \Delta^{\mathcal{I}}$, if $<x, y> \in r^{\mathcal{I}}$, then $y \in C^{\mathcal{I}}\}$	$\mathcal{AL}$
at-least restriction	$\geq n\ r$	$\{x \in \Delta^{\mathcal{I}} \mid \mid\{y \mid <x, y> \in r^{\mathcal{I}\leftarrow}\}\mid \geq n\}$	$\mathcal{N}$
at-most restriction	$\leq n\ r$	$\{x \in \Delta^{\mathcal{I}} \mid \mid\{y \mid <x, y> \in r^{\mathcal{I}\leftarrow}\}\mid \leq n\}$	$\mathcal{N}$
qualified at-least restriction	$\geq n\ r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \mid\{y \mid <x, y> \in r^{\mathcal{I}\leftarrow}$ and $y \in C^{\mathcal{I}}\}\mid \geq n\}$	$\mathcal{Q}$
qualified at-most restriction	$\leq n\ r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \mid\{y \mid <x, y> \in r^{\mathcal{I}\leftarrow}$ and $y \in C^{\mathcal{I}}\}\mid \leq n\}$	$\mathcal{Q}$
nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$	$\mathcal{Q}$

Appendix : axioms

Construct	Syntax	Semantics	Symbol
ABox			
concept assertion	$C(a)$ or a : C	$a^{\mathcal{I}} \in C^{\mathcal{I}}$	$\mathcal{AL}$
role assertion	$r(a, b)$ or $(a, b) : r$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$	$\mathcal{AL}$
TBox			
concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$	$\mathcal{AL}$
concept equivalence	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$	$\mathcal{AL}$
role inclusion	$r \sqsubseteq S$	$r^{\mathcal{I}} \subseteq S^{\mathcal{I}}$	$\mathcal{H}$
complex role inclusion	$r_1 \circ r_2 \sqsubseteq S$	$r_1^{\mathcal{I}} \circ r_2^{\mathcal{I}} \subseteq S^{\mathcal{I}}$	$\mathcal{R}$
Transitive role	Trans(r)	Mapped to binary transitive relations $r^{\mathcal{I}} \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, such as if $(d1; d2) \in r^{\mathcal{I}}$ and $(d2; d3) \in r^{\mathcal{I}}$, then $(d1; d3) \in r^{\mathcal{I}}$.	$\mathcal{R}^{+}$
Functional role	$\top \sqsubseteq \leq 1\ r$	mapped to binary functions $f^{\mathcal{I}} \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, such as $\forall a, b, c : f^{\mathcal{I}}(a; b) \wedge f^{\mathcal{I}}(a; c) \rightarrow b = c$.	$\mathcal{F}$

References

[Antoniou and van Harmelen 2004]: Antoniou G. and Van Harmelen F., A semantic web primer, *MIT Press, 2004.*

[Baader and Nutt 2003]: Baader F. and Nutt W., Basic description logic, *in Baader F., Calvanese D., McGuinness D., Nardi D., and Patel-Schneider P. (eds.):* The Description Logic Handbook: Theory, Implementation, and Applications, chapter 2, *Cambridge University Press, New York, NY, USA, 2003.*

[Baader et al. 2003]: Baader F., Calvanese D., McGuinness D., Nardi D., and Patel-Schneider P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. *Cambridge University Press, New York, NY, USA, 2003.*

[Baader et al. 2017]: Baader, F., Horrocks, I. Lutz C. and Sattler, U., An introduction to Description Logic, *Cambridge University Press, 2017.*

[Guarino et al. 2009]: Guarino N., Oberle D. and Staab S., What is an ontology? *in (Staab, S., Studer R. eds.), Handbook on Ontologies 2nd edition, Springer, 2009.*

[Krötzsch et al. 2013] Krötzsch M., Frantisek Simancik F., Horrocks I., A Description Logic Primer, *Cornell University Library, arXiv:1201.4089v3, 2013.*

[Ortiz & Simkus 2012]: Ortiz M. and Simkus M., Reasoning and Query Answering in Description Logics, *Tutorial, 8th Reasoning Web Summer School, Vienna University of Technology, 2012.*

[Rudolph 2011]: Rudolph S., Foundations of Description Logics, *in Polleres A., d'Amato C., Arenas M., Handschuh S., Kroner P., Ossowski S., Patel-Schneider P. (eds.), Reasoning Web. Semantic Technologies for the Web of Data - 7th International Summer School 2011, volume 6848 of LNCS, 76-136. Springer, 2011.*

THANK YOU