Semantic Web

— to Artificial Intelligence

Yue Ma

Laboratoire de Recherche en Informatique (LRI)
Université Paris Sud
ma@lri.fr

Outline

Description Logics

A subfield of knowledge representation

which is a subfield of artificial intelligence

Description

comes from concept description :

a formal expression that determines a set of individuals

Logics

comes from the semantics being defined using logic :

most Description Logics are fragments of First Order Logic

Knowledge Representation

Goal [Brachman & Nardi, 2003]

develop formalisms for providing high-level description of the world that can be effectively used to build intelligent applications

- formalism : well-defined syntax ; formal, unambiguous semantics
- high-level description : only relevant aspects represented
- intelligent applications : reason about the knowledge ; infer implicit knowledge
- effectively used : practical reasoning tools and efficient implementations

Helpful Materials

on brief introductions to DLs :

- Krötzsch, M., Simanǒík, F., Horrocks, I. : A description logic primer. CoRR abs/1201.4089 (2012)
- F. Baader and C. Lutz : Description Logic. In Patrick Blackburn, Johan van Benthem, and Frank Wolter, editors, The Handbook of Modal Logic, pages 757820. Elsevier, 2006.

Read them by yourselves, highly recommended !

Syntax

- explicit symbolic representation of the knowledge
- not implicit (as e.g. in neural networks)

$$
\begin{array}{ll}
\mathit{Woman} \equiv \mathit{Person} \sqcap \mathit{Female} & \mathit{Male}(\mathit{john}) \\
\mathit{Man} \equiv \mathit{Person} \sqcap \neg \mathit{Female} & \mathit{Male}(\mathit{marc}) \\
\mathit{Mother} \equiv \mathit{Woman} \sqcap \exists \mathit{hasChild}.\top & \mathit{Male}(\mathit{stephen}) \\
 & \mathit{Male}(\mathit{jason}) \\
\mathit{hasChild}(\mathit{stephen}, \mathit{marc}) & \mathit{Female}(\mathit{jill}) \\
\mathit{hasChild}(\mathit{marc}, \mathit{anna}) & \mathit{Female}(\mathit{anna}) \\
\mathit{hasChild}(\mathit{john}, \mathit{maria}) & \mathit{Female}(\mathit{maria}) \\
\mathit{hasChild}(\mathit{anna}, \mathit{jason}) &
\end{array}
$$

Semantics

connection between the symbolic representation and the "real world" entities it represents

Declarative semantics

- map symbols to an abstraction of the "world" (interpretation)
- notion of when a symbolic expression is true in the world (model)

NO procedural semantics :

should not just express how specific programs should behave

Expressive Power

(what can be expressed) depends on syntax and semantics

Equilibrium

- not too low : can all the relevant knowledge be represented ?
- not too high : are all the elements really necessary for the application ?

Reasoning

deduce implicit knowledge from the explicit representation

$$\forall x.\forall y.(male(y) \wedge \exists z.(child(x,z) \wedge child(z,y))) \rightarrow grandson(x,y)$$

$child(john, mary)$
$child(mary, paul)$
$male(paul)$

grandson(john,paul)

Knowledge Representation Systems

should provide inference tools to deduce implicit consequences

answers should depend on semantics ; not on the syntactic representation (same semantics must yield the same answer)

Reasoning Procedures

Decision Procedure

- sound : all positive answers are correct
- complete : all negative answers are correct
- terminating : always provides an answer in finite time

Efficient

- ideally, optimal w.r.t. worst-case complexity of the problem

Practical

- easy to implement and optimize
- behave well in applications

Examples of Formalisms

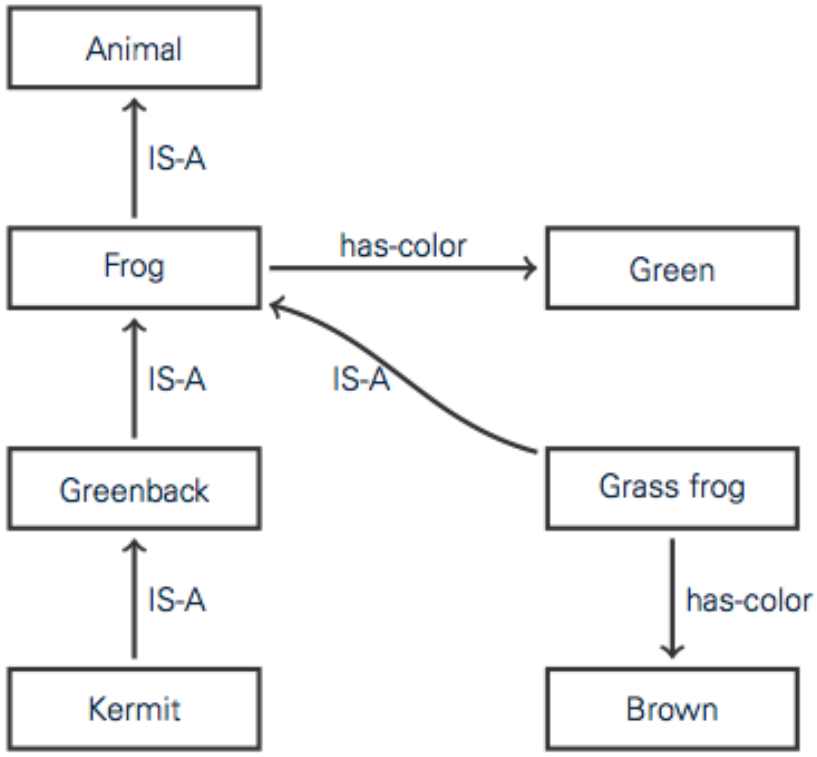
First-order logic

- satisfiability in FOL does not have a decision procedure
- is thus not an appropriate knowledge representation formalism

Propositional logic

- satisfiability in propositional logic has a decision procedure the problem is NP-complete
  - highly optimized SAT solver behave well in practice
  - however, expressive power is often insufficient

Terminological Knowledge



Terminological Knowledge

formalize the terminology (names) of the application domain :

- define important notions of the domain (classes, relations, objects)
- constrain the interpretations of these notions
- deduce consequences : subclass, instance relationships

Example (university domain)

- classes (concepts) : Person, Teacher, Course, Student, . . .
- relations (roles) : gives, attends, likes, . . .
- objects (individuals) : DL WS13, Marcel, Daniel, . . .
- constraints :

  every course is given by a teacher,

  every student must attend at least one course

Ontologies

the modern name for knowledge bases

Applications

- semantic web enable a common understanding of notions for semantic labeling of Web content
- information retrieval support automatic extraction of information from text
- medicine formal definitions that can be used by doctors, patients, insurance companies, etc, to communicate with each other

Description Logics

a class of logic-based knowledge representation formalisms for representing terminological knowledge

Prehistory

early approaches for representing terminological knowledge

- semantic networks (Quillian, 1968)
- frames (Minsky, 1975)

problems with missing semantics led to

- structured inheritance networks
- the first DL system KL-ONE

History of Description Logics

- Phase 1

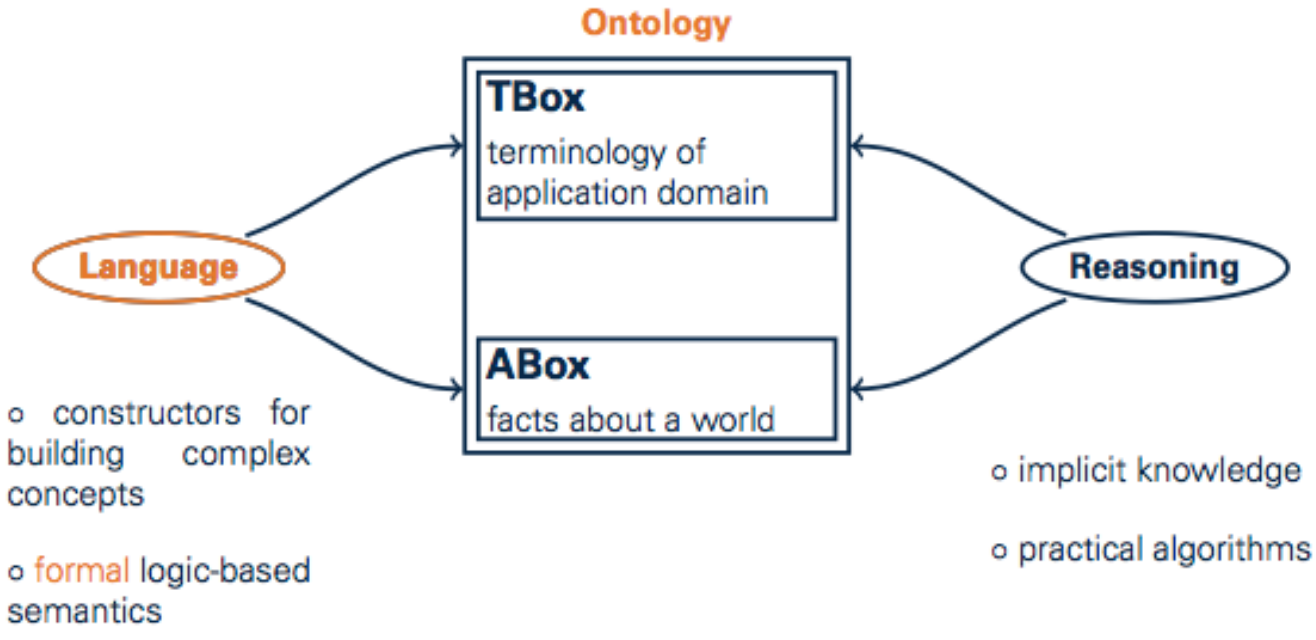
  implementation of systems based on incomplete structural subsumption algorithms

- Phase 2
  - tableau-based algorithms and complexity results
  - first tableau-based systems (Kris, Crack)
  - first formal study of optimization methods
- Phase 3
  - tableau-based algorithms for very expressive DLs
  - highly-optimized tableau-based systems (FaCT, Racer)
  - relationship to modal logic and FOL

History of Description Logics

- Phase 4
  - Web Ontology Language (OWL) based on DL
  - industrial-strength reasoners and ontology editors
  - light-weight (tractable) DLs
- Phase 5
  - non-standard reasoning
  - ontology management
  - semantic extensions

Structure of Description Logic Systems



Outline

Syntax of $\mathcal{ALC}$: Notations

- concept names are also called atomic concepts
- all other concepts are called complex
- instead of $\mathcal{ALC}$ concept, we often say concept

- A, B stand for concept names
- C, D for (complex) concepts
- r, s for role names

Example.
We can have *Female*, *Student*, *Pizza*, *Win*, *Country* as atomic concepts, *hasChild*, *hasTeacherhasTopping*, *locatedIn* as role names, and $Female \sqcap \exists hasChild.Student$ a complex concept.

How is the complex concept constructed ?
And what does it mean (intuitively) ?

Syntax of $\mathcal{ALC}$: constructing complex concepts

Let N_C and N_R two disjoint sets of concept names and role names, respectively. $\mathcal{ALC}$ concepts are defined by induction :

- if $A \in N_C$,then A is an $\mathcal{ALC}$ concept
- if C, D are $\mathcal{ALC}$ concepts and $r \in N_R$, 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)
- Abbreviations :
  - $\top = A \sqcup \neg A$ (top)
  - $\bot = A \sqcap \neg A$ (bottom)

Example. $Female \sqcap \exists hasChild.Student$

For you : try to write down your (atom, complex) $\mathcal{ALC}$ concepts.

$Female \sqcap \forall hasChild.(Student \sqcap \exists hasScore.GoodScore)$

Syntax of $\mathcal{ALC}$: constructing complex concepts

Let N_C and N_R two disjoint sets of concept names and role names, respectively. $\mathcal{ALC}$ concepts are defined by induction :

- if $A \in N_C$,then A is an $\mathcal{ALC}$ concept
- if C, D are $\mathcal{ALC}$ concepts and $r \in N_R$, 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)
- Abbreviations :
  - $\top = A \sqcup \neg A$ (top)
  - $\bot = A \sqcap \neg A$ (bottom)

Example. $Female \sqcap \exists hasChild.Student$

For you : try to write down your (atom, complex) $\mathcal{ALC}$ concepts.

$Female \sqcap \forall hasChild.(Student \sqcap \exists hasScore.GoodScore)$

Syntax of $\mathcal{ALC}$: constructing complex concepts

Let N_C and N_R two disjoint sets of concept names and role names, respectively. $\mathcal{ALC}$ concepts are defined by induction :

- if $A \in N_C$,then A is an $\mathcal{ALC}$ concept
- if C, D are $\mathcal{ALC}$ concepts and $r \in N_R$, 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)
- Abbreviations :
  - $\top = A \sqcup \neg A$ (top)
  - $\bot = A \sqcap \neg A$ (bottom)

Example. $Female \sqcap \exists hasChild.Student$

For you : try to write down your (atom, complex) $\mathcal{ALC}$ concepts.

$Female \sqcap \forall hasChild.(Student \sqcap \exists hasScore.GoodScore)$

Syntax of $\mathcal{ALC}$: constructing complex concepts

Let N_C and N_R two disjoint sets of concept names and role names, respectively. $\mathcal{ALC}$ concepts are defined by induction :

- if $A \in N_C$,then A is an $\mathcal{ALC}$ concept
- if C, D are $\mathcal{ALC}$ concepts and $r \in N_R$, 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)
- Abbreviations :
  - $\top = A \sqcup \neg A$ (top)
  - $\bot = A \sqcap \neg A$ (bottom)

Example. $Female \sqcap \exists hasChild.Student$

For you : try to write down your (atom, complex) $\mathcal{ALC}$ concepts.

$Female \sqcap \forall hasChild.(Student \sqcap \exists hasScore.GoodScore)$

Syntax of $\mathcal{ALC}$: constructing complex concepts

Let N_C and N_R two disjoint sets of concept names and role names, respectively. $\mathcal{ALC}$ concepts are defined by induction :

- if $A \in N_C$,then A is an $\mathcal{ALC}$ concept
- if C, D are $\mathcal{ALC}$ concepts and $r \in N_R$, 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)
- Abbreviations :
  - $\top = A \sqcup \neg A$ (top)
  - $\bot = A \sqcap \neg A$ (bottom)

Example. $\mathit{Female} \sqcap \exists \mathit{hasChild}.\mathit{Student}$

For you : try to write down your (atom, complex) $\mathcal{ALC}$ concepts.

$\mathit{Female} \sqcap \forall \mathit{hasChild}.(\mathit{Student} \sqcap \exists \mathit{hasScore}.\mathit{GoodScore})$

Syntax of $\mathcal{ALC}$: constructing complex concepts

Let N_C and N_R two disjoint sets of concept names and role names, respectively. $\mathcal{ALC}$ concepts are defined by induction :

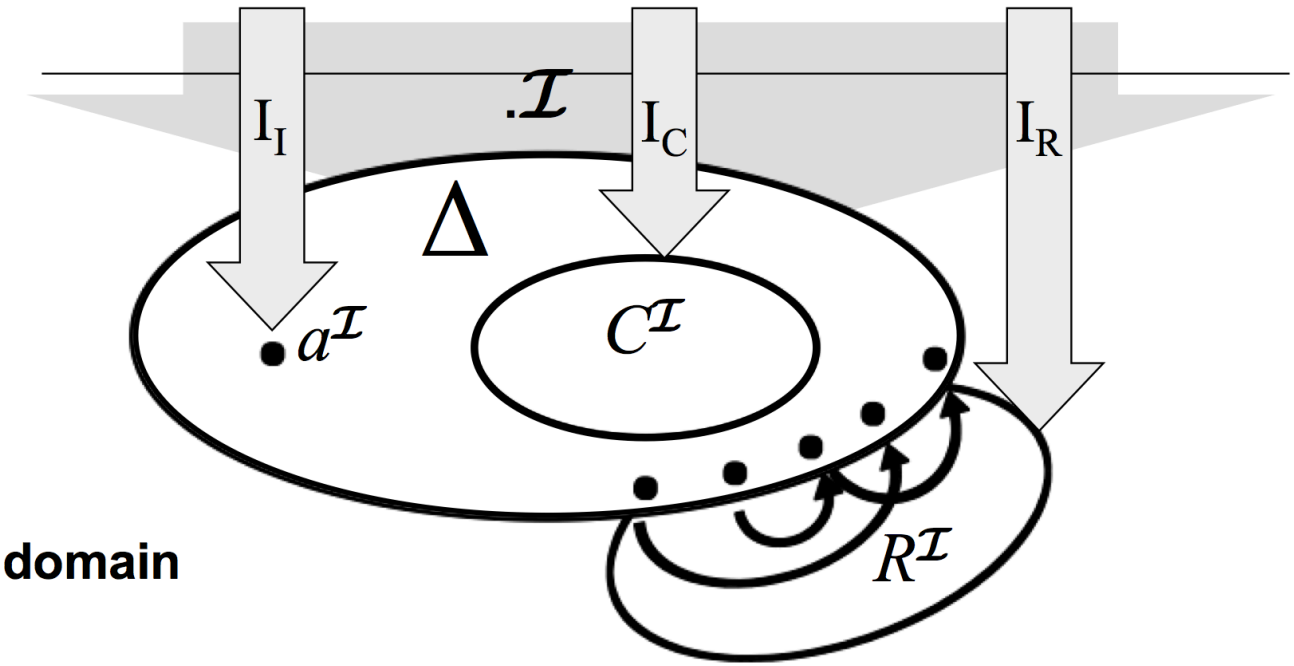
- if $A \in N_C$,then A is an $\mathcal{ALC}$ concept
- if C, D are $\mathcal{ALC}$ concepts and $r \in N_R$, then the following are $\mathcal{ALC}$ concepts :
  - $C \sqcap D$ (conjunction)
  - $C \sqcup D$ (disjunction)
  - $\neg C$ (negation)
  - $\exists r.C$ (existential restriction)
  - $\forall r.C$ (value restriction)
- Abbreviations :
  - $\top = A \sqcup \neg A$ (top)
  - $\bot = A \sqcap \neg A$ (bottom)

Example. $\mathit{Female} \sqcap \exists \mathit{hasChild}.\mathit{Student}$
For you : try to write down your (atom, complex) $\mathcal{ALC}$ concepts.
$\mathit{Female} \sqcap \forall \mathit{hasChild}.(\mathit{Student} \sqcap \exists \mathit{hasScore}.\mathit{GoodScore})$

Semantics of Description Logics

- **model-theoretic semantics**
- **starts with interpretations**
- **an interpretation $\mathcal{I}$ maps**

  individual names, class names and property names...



...into a domain

Semantics of $\mathcal{ALC}$

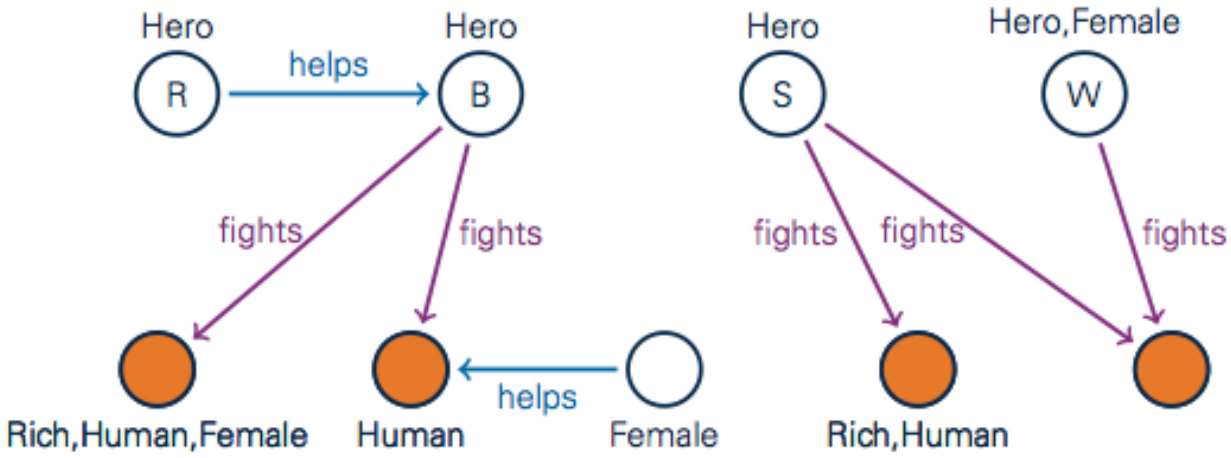
An interpretation $I = (\Delta^I, \cdot^I)$ consists of :

- a non-empty domain Δ^I , and
- an extension mapping $\cdot^I$ (also called interpretation function) :
  - $A^I \subseteq \Delta^I$ for all $A \in N_C$ (concepts interpreted as sets)
  - $r^I \subseteq \Delta^I \times \Delta^I$ for all $r \in N_R$ (roles interpreted as binary relations)

The extension mapping is extended to complex concepts :

$$
\begin{aligned}
(C \sqcap D)^I &:= C^I \cap D^I \\
(C \sqcup D)^I &:= C^I \cup D^I \\
(\neg C)^I &:= \Delta^I \setminus C^I \\
(\exists r.C)^I &:= \{d \in \Delta^I \mid \text{there is } e \in \Delta^I \text{ with } (d,e) \in r^I \text{ and } e \in C^I\} \\
(\forall r.C)^I &:= \{d \in \Delta^I \mid \text{for all } e \in \Delta^I \text{ with } (d,e) \in r^I, \text{ it holds } e \in C^I\}
\end{aligned}
$$

Interpretation Example



$$(Hero \sqcap \exists fights.Human)^I = \{B, S\}$$

$$(Hero \sqcap \forall fights.(Rich \sqcup \neg Human))^I = \{R, S, W\}$$

$$(\forall helps.Human)^I = \Delta^I \setminus \{R\}$$

Interpretation Example

Can you give an interpretation for the following concept ?

$$\mathit{Female} \sqcap \exists \mathit{hasChild}.\mathit{Student}$$

Correspondence between DL and NL

Description Logics	$\longleftrightarrow$	Natural Language
names in N_C, N_R		single words
complex concepts		phrases
axioms		sentences (statements [1])

A step further towards ontologies...

1. like "There is a wash basin in either a kitchen or a bathroom", but not emotional sentences “What a nice day !”

To define what is an ontology, we need the following definitions :

- TBox and ABox axioms
  - TBox : terminology box containing concept definitions and/or GCIs
  - ABox : assertion box, containing individual information.

Defining a Concept (aka. Concept Definitions)

Definitions.

- A concept definition is of the form $A = C$ where
  - A is a concept name.
  - C is a concept description.
- An interpretation I satisfies, also called a model of, the concept definition $A = C$ if $A^I = C^I$.

Examples.

$$\boxed{\mathit{Heroine} = \mathit{Hero} \sqcap \mathit{Female}}$$

$$\boxed{\mathit{Student} = \mathit{People} \sqcap \exists \mathit{registes}.(\mathit{School} \sqcup \mathit{University})}$$

Defining a Concept (aka. Concept Definitions)

Definitions.

- A concept definition is of the form $A = C$ where
  - A is a concept name.
  - C is a concept description.
- An interpretation I satisfies, also called a model of, the concept definition $A = C$ if $A^I = C^I$.

Examples.

$\boxed{\mathit{Heroine} = \mathit{Hero} \sqcap \mathit{Female}}$

Does the following interpretation I satisfy this concept definition ?

$$\Delta^I = \{a, b\}$$
$$\mathit{Heroine}^I = \{a\}$$
$$\mathit{Hero}^I = \{a, b\}$$
$$\mathit{Female}^I = \{b\}$$

If yes, why ?

If no, modify I to make it satisfy the concept definition.

Defining a Concept (aka. Concept Definitions)

Definitions.

- A concept definition is of the form $A = C$ where
  - A is a concept name.
  - C is a concept description.
- An interpretation I satisfies, also called a model of, the concept definition $A = C$ if $A^I = C^I$.

Examples.

$Heroine = Hero \sqcap Female$

$\boxed{Student = People \sqcap \exists registes.(School \sqcup University)}$

For you : give a model of this concept definition

GCIs (General Concept Inclusions) and TBoxes

Definitions.

- A general concept inclusion (GCI) is of the form $C \sqsubseteq D$, where C, D are concepts.
- A TBox $\mathcal{T}$ is a finite set of GCIs and/or concept definitions.
- An interpretation I satisfies the GCI $C \sqsubseteq D$ iff $C^I \subseteq D^I$. I is a model of a TBox T iff it satisfies all GCIs in T.
- Two TBoxes are equivalent if they have the same models.

Examples.
$\mathit{Hero} \sqcap \mathit{Villain} \sqsubseteq \bot$: two concepts are disjoint
$\mathit{Heroine} = \mathit{Hero} \sqcap \mathit{Female}$
$\mathit{Kitchen} \sqcup \mathit{Bathroom} \sqsubseteq \exists \mathit{hasWashbasin}.\top$
$\mathit{Cold} \sqcap \exists \mathit{causedBy}.\mathit{Virus} \sqsubseteq \mathit{Disease}$

GCIs (General Concept Inclusions) and TBoxes

Definitions.

- A general concept inclusion (GCI) is of the form $C \sqsubseteq D$, where C, D are concepts.
- A TBox $\mathcal{T}$ is a finite set of GCIs and/or concept definitions.
- An interpretation I satisfies the GCI $C \sqsubseteq D$ iff $C^I \subseteq D^I$. I is a model of a TBox T iff it satisfies all GCIs in T.
- Two TBoxes are equivalent if they have the same models.

Examples.
$\mathit{Hero} \sqcap \mathit{Villain} \sqsubseteq \bot$: two concepts are disjoint
$\mathit{Heroine} = \mathit{Hero} \sqcap \mathit{Female}$
$\mathit{Kitchen} \sqcup \mathit{Bathroom} \sqsubseteq \exists \mathit{hasWashbasin}.\top$
$\mathit{Cold} \sqcap \exists \mathit{causedBy}.\mathit{Virus} \sqsubseteq \mathit{Disease}$

GCIs (General Concept Inclusions) and TBoxes

Definitions.

- A general concept inclusion (GCI) is of the form $C \sqsubseteq D$, where C, D are concepts.
- A TBox $\mathcal{T}$ is a finite set of GCIs and/or concept definitions.
- An interpretation I satisfies the GCI $C \sqsubseteq D$ iff $C^I \subseteq D^I$. I is a model of a TBox T iff it satisfies all GCIs in T.
- Two TBoxes are equivalent if they have the same models.

Examples.
$\mathit{Hero} \sqcap \mathit{Villain} \sqsubseteq \bot$: two concepts are disjoint
$\mathit{Heroine} = \mathit{Hero} \sqcap \mathit{Female}$
$\mathit{Kitchen} \sqcup \mathit{Bathroom} \sqsubseteq \exists \mathit{hasWashbasin}.\top$
$\mathit{Cold} \sqcap \exists \mathit{causedBy}.\mathit{Virus} \sqsubseteq \mathit{Disease}$

GCIs (General Concept Inclusions) and TBoxes

Definitions.

- A general concept inclusion (GCI) is of the form $C \sqsubseteq D$, where C, D are concepts.
- A TBox $\mathcal{T}$ is a finite set of GCIs and/or concept definitions.
- An interpretation I satisfies the GCI $C \sqsubseteq D$ iff $C^I \subseteq D^I$. I is a model of a TBox T iff it satisfies all GCIs in T.
- Two TBoxes are equivalent if they have the same models.

Examples.
$\mathit{Hero} \sqcap \mathit{Villain} \sqsubseteq \bot$: two concepts are disjoint
$\mathit{Heroine} = \mathit{Hero} \sqcap \mathit{Female}$
$\mathit{Kitchen} \sqcup \mathit{Bathroom} \sqsubseteq \exists \mathit{hasWashbasin}.\top$
$\mathit{Cold} \sqcap \exists \mathit{causedBy}.\mathit{Virus} \sqsubseteq \mathit{Disease}$
For you : give a model for these GCIs and definitions. How many models does each of them have ?

GCIs (General Concept Inclusions) and TBoxes

Definitions.

- A general concept inclusion (GCI) is of the form $C \sqsubseteq D$, where C, D are concepts.
- A TBox $\mathcal{T}$ is a finite set of GCIs and/or concept definitions.
- An interpretation I satisfies the GCI $C \sqsubseteq D$ iff $C^I \subseteq D^I$. I is a model of a TBox T iff it satisfies all GCIs in T.
- Two TBoxes are equivalent if they have the same models.

Examples.
$\mathit{Hero} \sqcap \mathit{Villain} \sqsubseteq \bot$: two concepts are disjoint
$\mathit{Heroine} = \mathit{Hero} \sqcap \mathit{Female}$
$\mathit{Kitchen} \sqcup \mathit{Bathroom} \sqsubseteq \exists \mathit{hasWashbasin}.\top$
$\mathit{Cold} \sqcap \exists \mathit{causedBy}.\mathit{Virus} \sqsubseteq \mathit{Disease}$
For you : give a model for these GCIs and definitions. How many models does each of them have ?

Assertions and ABoxes

Definitions.

- An assertion is of the form $C(a)$ (concept assertion) or $r(a, b)$ (role assertion) where C is a concept, r a role, and a, b are individual names from a set N_I (disjoint with N_C, N_R)
- An ABox $\mathcal{A}$ is a finite set of assertions
- An interpretation I is a model of the ABox $\mathcal{A}$ if it satisfies all its assertions :
  - $a^I \in C^I$ for all $C(a) \in \mathcal{A}$.
  - $(a^I, b^I) \in R^I$ for all $R(a, b) \in \mathcal{A}$.

Examples.
$Kitchen(room1)$: the room1 is a kitchen
$locatedIn(whitechair1, room1)$: the whitechair1 is in the room1
$\neg Student(Jean)$: Jean is not a student

Assertions and ABoxes

Definitions.

- An assertion is of the form $C(a)$ (concept assertion) or $r(a, b)$ (role assertion) where C is a concept, r a role, and a, b are individual names from a set N_I (disjoint with N_C, N_R)
- An ABox $\mathcal{A}$ is a finite set of assertions
- An interpretation I is a model of the ABox $\mathcal{A}$ if it satisfies all its assertions :
  - $a^I \in C^I$ for all $C(a) \in \mathcal{A}$.
  - $(a^I, b^I) \in R^I$ for all $R(a, b) \in \mathcal{A}$.

Examples.
$Kitchen(room1)$: the room1 is a kitchen
$locatedIn(whitechair1, room1)$: the whitechair1 is in the room1
$\neg Student(Jean)$: Jean is not a student
For you : name some models for these ABox assertions. How many models does each of them have ?

Ontology

Definitions.

- An ontology $O = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.
- The interpretation I is a model of the ontology $O = (\mathcal{T}, \mathcal{A})$ iff it is a model of $\mathcal{T}$ and a model of $\mathcal{A}$.

Examples.
Many ontologies from http ://swoogle.umbc.edu and
http ://bioportal.bioontology.org

Let us now define the interesting reasoning services that we can benefit from ontologies

Terminological Reasoning of an Ontology

Let $\mathcal{T}$ be a TBox. Terminological reasoning refers to deciding the following problems :

- Satisfiability : a concept C is satisfiable w.r.t. $\mathcal{T}$ if and only if there is a model I of $\mathcal{T}$ such that $C^I \neq \emptyset$.
- Subsumption : C is subsumed by D w.r.t. $\mathcal{T}$ if and only if $C^I \subseteq D^I$ for all models I of $\mathcal{T}$.
- Equivalence : C is equivalent to D w.r.t. $\mathcal{T}$ if and only if $C^I = D^I$ for all models I of $\mathcal{T}$.
- Entailment : An ontology O entails $C \sqsubseteq D$, written $O \models C \sqsubseteq D$ ($O \models C(a)$, or $O \models R(a, b)$) if and only if $C^I \subseteq D^I$ (resp. $a^I \in C^I$, $(a^I, b^I) \in R^I$) for all models I of O.

Examples (next page)

Ontology : Reasoning Problems and Services

Examples.

- $A \sqcap \neg A$ is unsatisfiable
- $\forall r.A \sqcap \forall r.\neg A$ is satisfiable
- $\forall r.A \sqcap \exists r.\neg A$ is unsatisfiable
- $\exists r.(A \sqcap B)$ is subsumbed by $\exists r.A$
- $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$ (entailment)
- $\{A \sqsubseteq \exists r.B, B \sqsubseteq C\} \models A \sqsubseteq \exists r.C$ (entailment)

Assertional Reasoning of an Ontology

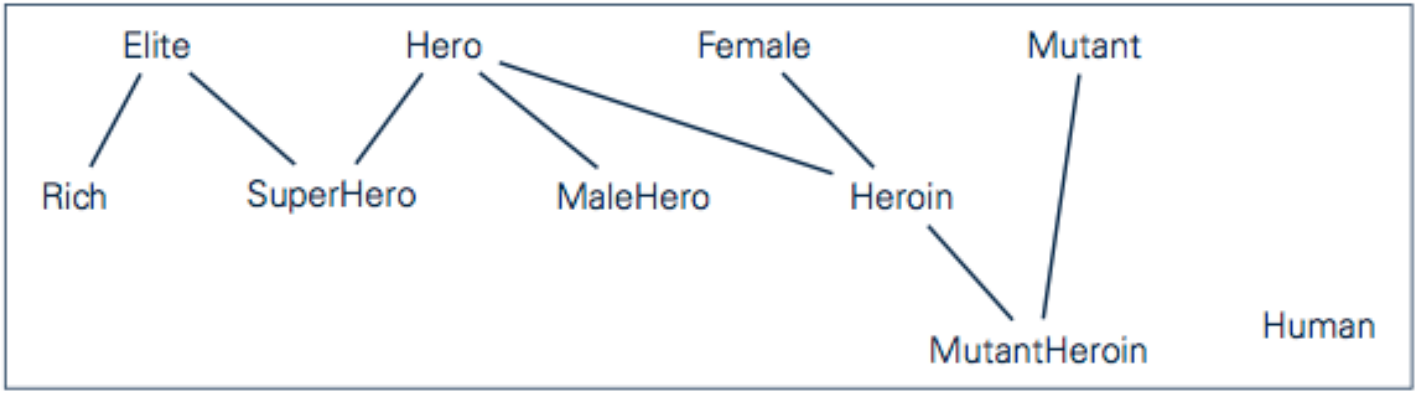
Let $O = (\mathcal{T}, \mathcal{A})$ be an ontology with TBox $\mathcal{T}$ and ABox $\mathcal{A}$.
Assertional reasoning refers to deciding the following problems :

- Consistency : O is consistent iff O has a model.
- Instance : a is an instance of a concept C w.r.t O iff $a^I \in C^I$ for all models I of O

TBox Classification

Computing the subsumption relations between all concept names in $\mathcal{T}$:

$$
\begin{aligned}
\text{Heroine} &\equiv \text{Hero} \sqcap \text{Female} \\
\text{MaleHero} &\equiv \text{Hero} \sqcap \neg\text{Female} \\
\text{MutantHeroine} &\equiv \text{Heroine} \sqcap \text{Mutant} \\
\text{Elite} &\equiv \text{Rich} \sqcup \neg\text{Human} \\
\text{Superhero} &\equiv \text{Hero} \sqcap \text{Elite}
\end{aligned}
$$



Realization

Computing the most specific concept names to which an individual belongs

$$\begin{aligned} \text{Heroine} &\equiv \text{Hero} \sqcap \text{Female} \\ \text{MaleHero} &\equiv \text{Hero} \sqcap \neg\text{Female} \\ \text{MutantHeroine} &\equiv \text{Heroine} \sqcap \text{Mutant} \\ \text{Elite} &\equiv \text{Rich} \sqcup \neg\text{Human} \\ \text{Superhero} &\equiv \text{Hero} \sqcap \text{Elite} \end{aligned}$$

Hero(Superman)

Superman is an instance of

Hero, MaleHero, Elite, Superhero

Reducibility of DL Reasoning Problems

Given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$:

- is $\mathcal{O}$ consistent ? $\mathcal{O} \models \top \sqsubseteq \bot$
- is $\mathcal{O}$ coherent ?
  (for some concept name A)
- classification
  (for some concept names A and B)
- realization
  (for all concept names A, individual names b)

Reducibility of DL Reasoning Problems

Given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$:

- is $\mathcal{O}$ consistent ?
- is $\mathcal{O}$ coherent ?

  (for some concept name A)
- classification

  (for some concept names A and B)
- realization

  (for all concept names A, individual names b)

$\mathcal{O} \models \top \sqsubseteq \bot$

Reducibility of DL Reasoning Problems

Given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$:

- is $\mathcal{O}$ consistent ? $\quad \mathcal{O} \models \top \sqsubseteq \bot$
- is $\mathcal{O}$ coherent ? $\quad \mathcal{O} \models A \sqsubseteq \bot$

  (for some concept name A)
- classification

  (for some concept names A and B)
- realization

  (for all concept names A, individual names b)

Reducibility of DL Reasoning Problems

Given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$:

- is $\mathcal{O}$ consistent ? $\quad \mathcal{O} \models \top \sqsubseteq \bot$
- is $\mathcal{O}$ coherent ? $\quad \mathcal{O} \models A \sqsubseteq \bot$

  (for some concept name A)
- classification

  (for some concept names A and B)
- realization

  (for all concept names A, individual names b)

Reducibility of DL Reasoning Problems

Given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$:

- is $\mathcal{O}$ consistent ? $\quad \mathcal{O} \models \top \sqsubseteq \bot$
- is $\mathcal{O}$ coherent ? $\quad \mathcal{O} \models A \sqsubseteq \bot$

  (for some concept name A)
- classification $\quad \mathcal{O} \models A \sqsubseteq B$

  (for some concept names A and B)
- realization

  (for all concept names A, individual names b)

Reducibility of DL Reasoning Problems

Given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$:

- is $\mathcal{O}$ consistent ? $\quad \mathcal{O} \models \top \sqsubseteq \bot$
- is $\mathcal{O}$ coherent ? $\quad \mathcal{O} \models A \sqsubseteq \bot$

  (for some concept name A)
- classification $\quad \mathcal{O} \models A \sqsubseteq B$

  (for some concept names A and B)
- realization

  (for all concept names A, individual names b)

Reducibility of DL Reasoning Problems

Given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$:

- is $\mathcal{O}$ consistent ? $\mathcal{O} \models \top \sqsubseteq \bot$
- is $\mathcal{O}$ coherent ? $\mathcal{O} \models A \sqsubseteq \bot$

  (for some concept name A)
- classification $\mathcal{O} \models A \sqsubseteq B$

  (for some concept names A and B)
- realization $\mathcal{O} \models A(b)$

  (for all concept names A, individual names b)

Reducibility of DL Reasoning Problems

Given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$:

- is $\mathcal{O}$ consistent ? $\mathcal{O} \models \top \sqsubseteq \bot$
- is $\mathcal{O}$ coherent ? $\mathcal{O} \models A \sqsubseteq \bot$

  (for some concept name A)
- classification $\mathcal{O} \models A \sqsubseteq B$

  (for some concept names A and B)
- realization $\mathcal{O} \models A(b)$

  (for all concept names A, individual names b)

Question : do we need 4 different algorithms for these ?

Consistency Suffices

Theorem : Let O be an ontology and a a fresh individual. Then :

1. C is satisfiable w.r.t. $\mathcal{O}$ iff $\mathcal{O} \cup \{C(a)\}$ is consistent.
2. $\mathcal{O}$ is coherent for a concept name A iff $\mathcal{O} \cup \{A(a)\}$ is consistent
3. $\mathcal{O} \models C \sqsubseteq D$ iff $\mathcal{O} \cup \{(C \sqcap \neg D)(a)\}$ is not consistent
4. $\mathcal{O} \models C(b)$ iff $\mathcal{O} \cup \{\neg C(b)\}$ is not consistent

Consistency Suffices

Theorem : Let O be an ontology and a a fresh individual. Then :

1. C is satisfiable w.r.t. $\mathcal{O}$ iff $\mathcal{O} \cup \{C(a)\}$ is consistent.
2. $\mathcal{O}$ is coherent for a concept name A iff $\mathcal{O} \cup \{A(a)\}$ is consistent
3. $\mathcal{O} \models C \sqsubseteq D$ iff $\mathcal{O} \cup \{(C \sqcap \neg D)(a)\}$ is not consistent
4. $\mathcal{O} \models C(b)$ iff $\mathcal{O} \cup \{\neg C(b)\}$ is not consistent

Consistency Suffices

Theorem : Let O be an ontology and a a fresh individual. Then :

1. C is satisfiable w.r.t. $\mathcal{O}$ iff $\mathcal{O} \cup \{C(a)\}$ is consistent.
2. $\mathcal{O}$ is coherent for a concept name A iff $\mathcal{O} \cup \{A(a)\}$ is consistent
3. $\mathcal{O} \models C \sqsubseteq D$ iff $\mathcal{O} \cup \{(C \sqcap \neg D)(a)\}$ is not consistent
4. $\mathcal{O} \models C(b)$ iff $\mathcal{O} \cup \{\neg C(b)\}$ is not consistent

Consistency Suffices

Theorem : Let O be an ontology and a a fresh individual. Then :

1. C is satisfiable w.r.t. $\mathcal{O}$ iff $\mathcal{O} \cup \{C(a)\}$ is consistent.
2. $\mathcal{O}$ is coherent for a concept name A iff $\mathcal{O} \cup \{A(a)\}$ is consistent
3. $\mathcal{O} \models C \sqsubseteq D$ iff $\mathcal{O} \cup \{(C \sqcap \neg D)(a)\}$ is not consistent
4. $\mathcal{O} \models C(b)$ iff $\mathcal{O} \cup \{\neg C(b)\}$ is not consistent

Consistency Suffices

Theorem : Let O be an ontology and a a fresh individual. Then :

1. C is satisfiable w.r.t. $\mathcal{O}$ iff $\mathcal{O} \cup \{C(a)\}$ is consistent.
2. $\mathcal{O}$ is coherent for a concept name A iff $\mathcal{O} \cup \{A(a)\}$ is consistent
3. $\mathcal{O} \models C \sqsubseteq D$ iff $\mathcal{O} \cup \{(C \sqcap \neg D)(a)\}$ is not consistent
4. $\mathcal{O} \models C(b)$ iff $\mathcal{O} \cup \{\neg C(b)\}$ is not consistent

Answer : a decision procedure to solve consistency decides all standard DL reasoning problems !

This does not mean that the naive reduction is practical !

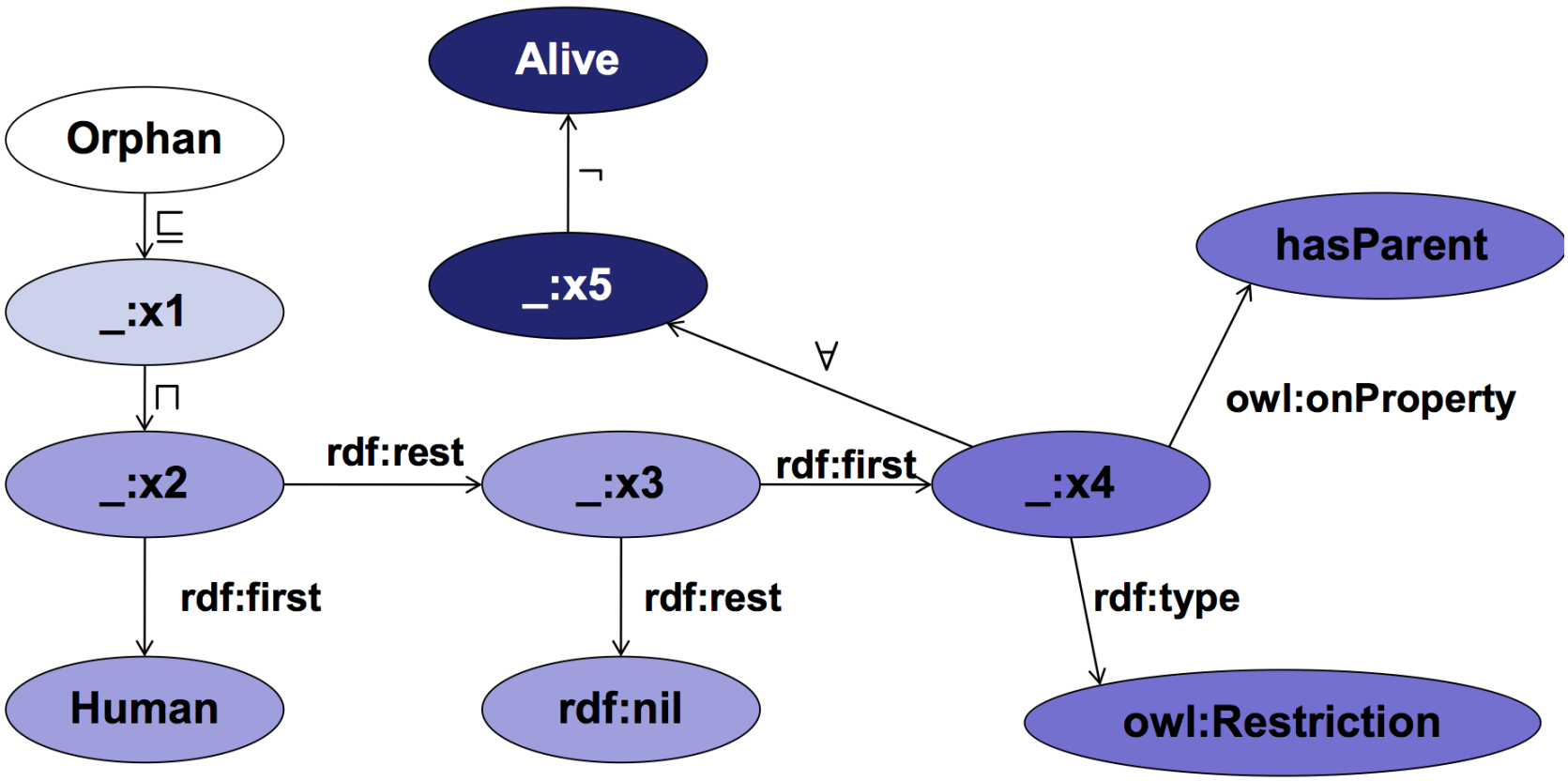
Reasoning algorithms for $\mathcal{ALC}$

A separate slide for details.

Description Logics and Semantic Web

Description Logics and Graphs (RDF triples)

$$\mathtt{Orphan} \sqsubseteq \mathtt{Human} \sqcap \forall \mathtt{hasParent}.\neg \mathtt{Alive}$$



OWL and DLs are axioms centralized (OWL API implementation), not RDF triple oriented (Jena, Sparql) !

OWL and DLs

Ontology := TBox + ABox

OWL is W3C-standardized Web Ontology Language

- If you publish your ontology, it should be in OWL

OWL is basically a DL + a common syntax

- Syntax : OWL/XML, Functional, Manchester, RDF-based
- Semantics : DL model theory

  any reasoning in OWL is reduced to reasoning in DL !!!

OWL includes more stuff :

- Datatypes : strings, integers, dates, etc.

  a.k.a. concrete domains in some DLs
- Non-logical stuff : annotations

OWL Profiles

OWL is a family of languages designed for specific scenarios

- DLs are theoretical foundations of OWL standards
- OWL : OWL-FULL, OWL-DL, OWL-LITE
- OWL 2 : OWL 2 EL, OWL 2 QL, OWL 2 RL, OWL 2 DL

Profile	DL	Scenario
OWL 2 EL	$\mathcal{EL}++$	Maintaining large but simple terminologies (for HCLS, biology, etc., e.g., SNOMED CT) PTime classification
OWL 2 QL	DL-Lite	Scalable ontology-based data access scalable query answering
OWL 2 RL	DLP	Rule-based applications completeness w.r.t. rule systems
OWL 2 DL	$\mathcal{SROIQ}$	Encapsulates all above, remains decidable. Reasoning is tableau-based

$\mathcal{SROIQ}$: the biggest DL for OWL and OWL 2

Concept	Syntax	Semantics
atomic concept	$C \in N_C$	$C^I \subseteq \Delta^I$
individual	$a \in N_I$	$a^I \in \Delta^I$
nominal	$\{a_1, \ldots, a_n\},\ a_i \in N_I$	$\{a_1^I, \ldots, a_n^I\}$
role	$R \in N_R$	$R^I \subseteq \Delta^I \times \Delta^I$
inverse role	$R^-,\ R \in N_R$	$R^{-I} = \{(y,x) \mid (x,y) \in R^I\}$
universal role	U	$U^I = \Delta^I \times \Delta^I$
role composition	$R_1 \circ \ldots \circ R_n$	$\{(x,z) \mid (x,y_1) \in R_1^I \wedge (y_1,y_2) \in R_2^I \wedge \ldots (y_{n-1},y_n) \wedge (y_n,z) \in R_n^I\}$
top	$\top$	$\top^I = \Delta^I$
bottom	$\bot$	$\bot^I = \varnothing$
negation	$\neg C$	$(\neg C)^I = \Delta^I / C^I$
conjunction	$C_1 \sqcap C_2$	$(C_1 \sqcap C_2)^I = C_1^I \cap C_2^I$
disjunction	$C_1 \sqcup C_2$	$(C_1 \sqcup C_2)^I = C_1^I \cup C_2^I$
exists restriction	$\exists R.C$	$\{x \mid (\exists y)[(x,y) \in R^I \wedge y \in C^I]\}$
value restriction	$\forall R.C$	$\{x \mid (\forall y)([(x,y) \in R^I \rightarrow y \in C^I]\}$
self restriction	$\exists R.Self$	$\{x \mid (x,x) \in R^I\}$
atmost restriction	$\leq nR.C$	$\{x \mid \sharp\{y \mid (x,y) \in R^I \wedge y \in C^I\} \leq n\}$
atleast restriction	$\geq nR.C$	$\{x \mid \sharp\{y \mid (x,y) \in R^I \wedge y \in C^I\} \geq n\}$

$\mathcal{SROIQ}(\mathcal{D})$: the biggest DL for OWL and OWL 2

Concept	**Syntax**	$\mathcal{SROIQ}$ **Semantics**
Datatype	D	$D^I \subseteq \Delta_D^I$
Datatype property	U	$U^I \subseteq \Delta^I \times \Delta_D^I$

Axiom	**Syntax**	SROIQ **Semantics**
concept inclusion	$C_1 \sqsubseteq C_2$	$C_1^I \subseteq C_2^I$
role inclusion	$R_1 \circ \ldots \circ R_n \sqsubseteq R_{n+1}$	$(R_1 \circ \ldots \circ R_n)^I \subseteq R_{n+1}^I$
reflexivity	$Ref(R)$	$\{(x,x)\|x \in \Delta^I\} \subseteq R^I$
irreflexivity	$Irr(R)$	$\{(x,x)\|x \in \Delta^I\} \cap R^I = \varnothing$
disjointness	$Dis(R,S)$	$S^I \cap R^I = \varnothing$
class assertion	$C(a)$	$a^I \in C^I$
inequality assertion	$a \neq b$	$a^I \neq b^I$
role (instance) assertion	$R(a,b)$	$(a^I, b^I) \in R^I$
negative role assertion	$\neg R(a,b)$	$(a^I, b^I) \notin R^I$

OWL (Ontology Web Language) and DLs

```
:Mary rdf:type :Woman .
```

```
:John :hasWife :Mary .
```

```
:John owl:differentFrom :Bill .
```

$\{John\} \sqcap \{Bill\} \sqsubseteq \bot$

```
:James owl:sameAs :Jim.
```

$\{John\} \equiv \{Jim\}$

```
:John  :hasAge "51"^^xsd:nonNegativeInteger .
```

```
[]  rdf:type                  owl:NegativePropertyAssertion ;
    owl:sourceIndividual      :Bill ;
    owl:assertionProperty     :hasWife ;
    owl:targetIndividual      :Mary .
```

$\neg hasWife(Bill,Mary)$

```
[]  rdf:type                  owl:NegativePropertyAssertion ;
    owl:sourceIndividual      :Jack ;
    owl:assertionProperty     :hasAge ;
    owl:targetValue           53 .
```

OWL (Ontology Web Language) and DLs

```
:Woman rdfs:subClassOf :Person .
```

```
:Person owl:equivalentClass :Human .
```

```
[]  rdf:type      owl:AllDisjointClasses ;
    owl:members   ( :Woman   :Man ) .
```

$\mathbf{Woman} \sqcap \mathbf{Man} \sqsubseteq \bot$

```
:hasWife rdfs:subPropertyOf :hasSpouse .
```

```
:hasWife rdfs:domain :Man ;
         rdfs:range  :Woman .
```

OWL (Ontology Web Language) and DLs

```
:Mother  owl:equivalentClass  [
  rdf:type              owl:Class ;
  owl:intersectionOf  ( :Woman :Parent )
] .
```

Mother ≡ Woman ⊓ Parent

```
:Parent  owl:equivalentClass  [
  rdf:type      owl:Class ;
  owl:unionOf  ( :Mother :Father )
] .
```

Parent ≡ Mother ⊔ Father

```
:ChildlessPerson  owl:equivalentClass  [
  rdf:type              owl:Class ;
  owl:intersectionOf  ( :Person  [ owl:complementOf  :Parent ] )
] .
```

ChildlessPerson ≡ Person ⊓ ¬Parent

```
:Grandfather  rdfs:subClassOf  [
  rdf:type              owl:Class ;
  owl:intersectionOf  ( :Man  :Parent )
] .
```

OWL (Ontology Web Language) and DLs

Person ⊓ ¬Parent (Jack)

```
:Jack  rdf:type  [
  rdf:type              owl:Class ;
  owl:intersectionOf    ( :Person
                          [ rdf:type           owl:Class ;
                            owl:complementOf   :Parent     ]
                        )
] .
```

OWL (Ontology Web Language) and DLs

```
:Parent  owl:equivalentClass  [
  rdf:type              owl:Restriction ;
  owl:onProperty        :hasChild ;
  owl:someValuesFrom    :Person
] .
```

Parent ≡ ∃hasChild.Person

```
:Orphan  owl:equivalentClass  [
  rdf:type              owl:Restriction ;
  owl:onProperty        [ owl:inverseOf  :hasChild ] ;
  owl:allValuesFrom     :Dead
] .
```

Orphan ≡ ∀ hasChild⁻.Dead

OWL (Ontology Web Language) and DLs

```
:JohnsChildren  owl:equivalentClass  [
  rdf:type          owl:Restriction ;
  owl:onProperty    :hasParent ;
  owl:hasValue      :John
] .
```

$\mathbf{JohnsChildren \equiv \exists hasParent.\{John\}}$

```
:NarcisticPerson owl:equivalentClass  [
  rdf:type          owl:Restriction ;
  owl:onProperty    :loves ;
  owl:hasSelf       "true"^^xsd:boolean .
] .
```

$\mathbf{NarcisticPerson \equiv \exists loves.Self}$

OWL (Ontology Web Language) and DLs

≤4 hasChild.Parent (John)

```
:John  rdf:type  [
  rdf:type                          owl:Restriction ;
  owl:maxQualifiedCardinality       "4"^^xsd:nonNegativeInteger ;
  owl:onProperty                    :hasChild ;
  owl:onClass                       :Parent
] .
```

```
:John  rdf:type  [
  rdf:type                          owl:Restriction ;
  owl:minQualifiedCardinality       "2"^^xsd:nonNegativeInteger ;
  owl:onProperty                    :hasChild ;
  owl:onClass                       :Parent
] .
```

≥2 hasChild.Parent (John)

```
:John  rdf:type  [
  rdf:type                          owl:Restriction ;
  owl:qualifiedCardinality          "3"^^xsd:nonNegativeInteger ;
  owl:onProperty                    :hasChild ;
  owl:onClass                       :Parent
] .
```

=3 hasChild.Parent (John)

OWL (Ontology Web Language) and DLs

```
:John  rdf:type  [
  rdf:type          owl:Restriction ;
  owl:cardinality   "5"^^xsd:nonNegativeInteger ;
  owl:onProperty    :hasChild
] .
```

$=5\ hasChild.\top$ (John)

```
:MyBirthdayGuests  owl:equivalentClass  [
  rdf:type   owl:Class ;
  owl:oneOf  ( :Bill  :John  :Mary )
] .
```

MyBirthdayGuests $\equiv$ {Bill, John, Mary}

OWL (Ontology Web Language) and DLs

```
:hasParent owl:inverseOf :hasChild .

:Orphan  owl:equivalentClass  [
  rdf:type              owl:Restriction ;
  owl:onProperty        [ owl:complementOf    :hasChild ] ;
  owl:allValuesFrom     :Dead
] .

:hasSpouse  rdf:type  owl:SymmetricProperty .

:hasChild  rdf:type  owl:AsymmetricProperty .

:hasParent  owl:propertyDisjointWith  :hasSpouse .

:hasRelative  rdf:type  owl:ReflexiveProperty .

:parentOf  rdf:type  owl:IrreflexiveProperty .

:hasHusband  rdf:type  owl:FunctionalProperty .

:hasHusband  rdf:type  owl:InverseFunctionalProperty .

:hasAncestor  rdf:type  owl:TransitiveProperty .
```

$\mathbf{Orphan} \equiv \forall\, \mathbf{hasChild}^{-}.\mathbf{Dead}$

OWL (Ontology Web Language) and DLs

```
:hasGrandparent owl:propertyChainAxiom ( :hasParent :hasParent ).
```

hasParent ∘ hasParent ⊑ hasGrandParent

```
:Person owl:hasKey ( :hasSSN ) .
```

In OWL 2 a collection of (data or object) properties can be assigned as a key to a class expression. This means that each named instance of the class expression is uniquely identified by the set of values which these properties attain in relation to the instance.

OWL (Ontology Web Language) and DLs

```
:personAge  owl:equivalentClass
 [ rdf:type  rdfs:Datatype;
   owl:onDatatype  xsd:integer;
   owl:withRestrictions (
      [ xsd:minInclusive  "0"^^xsd:integer ]
      [ xsd:maxInclusive  "150"^^xsd:integer ]
   )
 ] .
```

Datatype *facets*

```
:majorAge  owl:equivalentClass
  [ rdf:type  rdfs:Datatype;
    owl:intersectionOf (
       :personAge
       [ rdf:type  rdfs:Datatype;
         owl:datatypeComplementOf  :minorAge ]
    )
```

OWL (Ontology Web Language) and DLs

Feature	Related OWL vocabulary	FOL	DL
top/bottom class	`owl:Thing/owl:Nothing`	(axiomatise)	$\top/\bot$
Class intersection	`owl:intersectionOf`	$\wedge$	$\sqcap$
Class union	`owl:unionOf`	$\vee$	$\sqcup$
Class complement	`owl:complementOf`	$\neg$	$\neg$
Enumerated class	`owl:oneOf`	(ax. with $\approx$)	{a}
Property restrictions	`owl:onProperty`		
Existential	`owl:someValueFrom`	$\exists y \ldots$	$\exists R.C$
Universal	`owl:allValuesFrom`	$\forall y \ldots$	$\forall R.C$
Min. cardinality	`owl:minQualifiedCardinality` `owl:onClass`	$\exists y1 \ldots yn. \ldots$	$\geq n\ S.C$
Max. cardinality	`owl:maxQualifiedCardinality` `owl:onClass`	$\forall y1 \ldots yn+1. \ldots \rightarrow \ldots$	$\leq n\ S.C$
Local reflexivity	`owl:hasSelf`	R(x,x)	$\exists R.Self$

References and extra material

Ontology editors and reasoners

- List of DL reasoners :

  `www.cs.man.ac.uk/sattler/reasoners.html?`
- List of OWL implementations :

  `http://www.w3.org/2001/sw/wiki/OWL/Implementations`
- OWL Reasoner competition :

  `http://ore2013.cs.manchester.ac.uk/`
- Protege ontology editor :

  `http://protege.stanford.edu/`

References on DLs (1)

Comprehensive textbook, but a bit dated.

The Description Logic Handbook : Theory, Implementation and Applications. Edited by Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. Cambridge University Press (2003). Revised edition 2010.

The following ESSLLI summer school course provides a good introduction to the area :
`http://www.informatik.uni-bremen.de/?ts/teaching/2012_dl/`

References on DLs (2)

The lecture notes of the ReasoningWeb summer school (published by Springer) give more detailed introductions to different topics in DLs. Some chapters of particular interest :

- Sebastian Rudolph : Foundations of Description Logics (RW 2011)
- Magdalena Ortiz, Mantas Simkus : Reasoning and Query Answering in Description Logics (RW 2012)
- Markus Krötzsch : OWL 2 Profiles : An Introduction to Lightweight Ontology Languages (RW 2012)
- Roman Kontchakov, Mariano Rodriguez-Muro, Michael Zakharyaschev : Ontology-Based Data Access with Databases : A Short Course (RW 2013)
- Meghyn Bienvenu, Magdalena Ortiz : Ontology-Mediated Query Answering with Data-Tractable Description Logics (RW 2015)

References on DLs (3)

The DL complexity navigator is helpful for finding complexity results for expressive DLs :

`http://www.cs.man.ac.uk/?ezolin/dl/`

For general overview of current DL research, check out the proceedings of the annual DL workshop :

`http://dl.kr.org/workshops/`

DL vs FOL

- ALC can be seen as a fragment of First-order Logic
  - concept names are unary predicates
  - role names are binary predicates
- Interpretations can obviously be seen as first-order interpretations for this signature
- Concepts correspond to FOL formulae with one free variable

Translation to FOL

Syntactic translation $C \mapsto \pi_x(C)$

$$
\begin{aligned}
\pi_x(A) &:= A(x) \text{ for } A \in N_C \\
\pi_x(C \sqcap D) &:= \pi_x(C) \wedge \pi_x(D) \\
\pi_x(C \sqcup D) &:= \pi_x(C) \vee \pi_x(D) \\
\pi_x(\neg C) &:= \neg \pi_x(C) \\
\pi_x(\exists r.C) &:= \exists y.(r(x,y) \wedge \pi_y C), y \text{ new variable different from} x \\
\pi_x(\forall r.C) &:= \forall y.(r(x,y) \rightarrow \pi_y C)
\end{aligned}
$$

Example.

$\pi_x(\forall r.(A \sqcap \exists s.B)) = ?$

Translation to FOL

Syntactic translation $C \mapsto \pi_x(C)$

$$
\begin{aligned}
\pi_x(A) &:= A(x) \text{ for } A \in N_C \\
\pi_x(C \sqcap D) &:= \pi_x(C) \wedge \pi_x(D) \\
\pi_x(C \sqcup D) &:= \pi_x(C) \vee \pi_x(D) \\
\pi_x(\neg C) &:= \neg \pi_x(C) \\
\pi_x(\exists r.C) &:= \exists y.(r(x,y) \wedge \pi_y C), y \text{ new variable different from} x \\
\pi_x(\forall r.C) &:= \forall y.(r(x,y) \rightarrow \pi_y C)
\end{aligned}
$$

Example.
$\pi_x(\forall r.(A \sqcap \exists s.B)) = ?$

Lemma (optional).
C and $\pi_x(C)$ have the same extension ; that is,

$$C^I = \{d \in \Delta^I \mid I \models \pi_x(C)(d)\}.$$