Semantic Web

— to Artificial Intelligence

Yue Ma

Laboratoire Interdisciplinaire des Sciences du Numérique
(LISN)
Université Paris-Saclay
yue.ma@universite-paris-saclay.fr

Outline

OWL ontology from a formal perspective : Description Logics (DLs)

- Components of an ontology
- A basic DL : synatx
- A basic DL : semantics
- Ontology reasoning tasks
- History of Description Logics

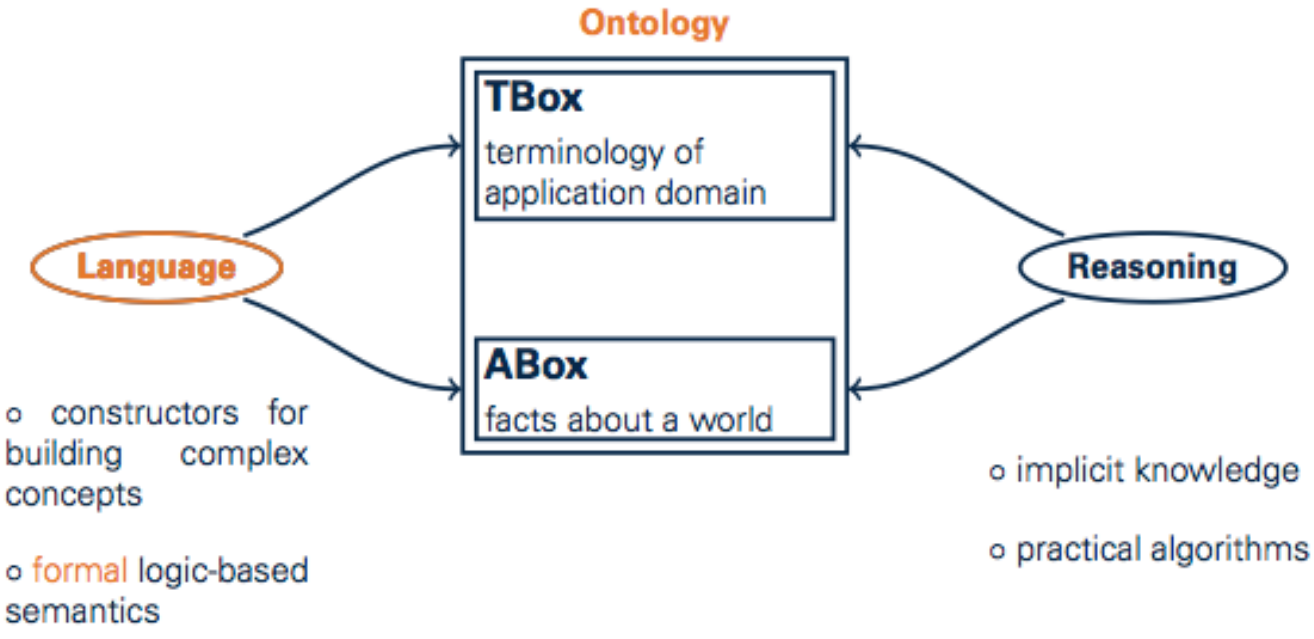
OWL ontology from a formal perspective : Description Logics (DLs)

Expressing knowledge

- Heart disease is a kind of mediastinum disorder.
- Heart disease happens in heart structure.
- Smoking or alcoholism is bad for health.
- Parents are people who have at least one child.
- Each student can register at most two study programs each semester.
- Man and woman are disjoint.

How we can express these pieces of knowledge in a way computers can understand ?

Structure of an ontology



Correspondence between Ontology and NL

Semantic Web Ontology	$\longleftrightarrow$	Natural Language
names in N_C, N_R		single words
complex concepts		phrases
axioms		sentences (statements) expressing knowledge [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 !”

DL ontology

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 :

- classes (concepts) : Person, Teacher, Course, Student, HeartDisease, HeartStructure, ...
- relations (roles) : gives, attends, likes, findingSite, ...
- objects (individuals) : CourseSemanticweb, John, Jim, ...
- constraints/knowledge :

  every course is given by a teacher,

  every student must attend at least one course

  heart disease is a kind of mediastinum disorder

  heart disease happens in heart structure

  happy parents are those whose children are happy

  John is a happy boy

  John's father is Jim

A basic Description Logic $\mathcal{ALC}$: syntax

- concept names are also called atomic concepts
- all other concepts are called complex
- N_C : a set of concept names
- N_R a set of role name

Example.
Let N_C=$\{Female, Parent, Student\}$ and $N_R = \{hasChild\}$.

A basic Description Logic $\mathcal{ALC}$: syntax

- concept names are also called atomic concepts
- all other concepts are called complex
- N_C : a set of concept names
- N_R a set of role name

Example.
Let N_C=$\{Female, Parent, Student\}$ and $N_R = \{hasChild\}$.

We may express the concept “female who has a student child’’ :

A basic Description Logic $\mathcal{ALC}$: syntax

- concept names are also called atomic concepts
- all other concepts are called complex
- N_C : a set of concept names
- N_R a set of role name

Example.
Let N_C=$\{Female, Parent, Student\}$ and $N_R = \{hasChild\}$.

We may express the concept “female who has a student child’’ :

A basic Description Logic $\mathcal{ALC}$: syntax

- concept names are also called atomic concepts
- all other concepts are called complex
- N_C : a set of concept names
- N_R a set of role name

Example.
Let N_C=$\{Female, Parent, Student\}$ and $N_R = \{hasChild\}$.

We may express the concept "female who has a student child'' :

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

How about "female whose children are students'' :

A basic Description Logic $\mathcal{ALC}$: syntax

- concept names are also called atomic concepts
- all other concepts are called complex
- N_C : a set of concept names
- N_R a set of role name

Example.
Let N_C=$\{Female, Parent, Student\}$ and $N_R = \{hasChild\}$.

We may express the concept "female who has a student child'' :

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

How about "female whose children are students'' :

$$Female \sqcap \forall hasChild.Student$$

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

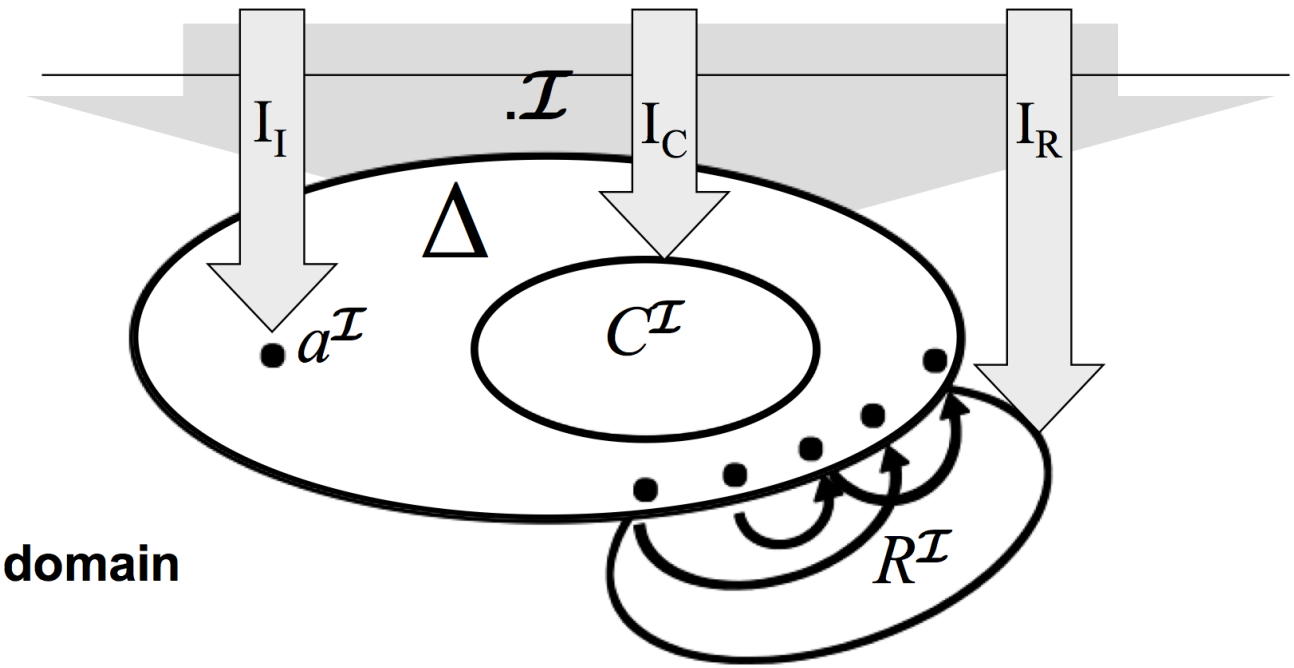
For you : try to write down your (atom, complex) $\mathcal{ALC}$ concepts.

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

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 $\mathcal{ALC}$

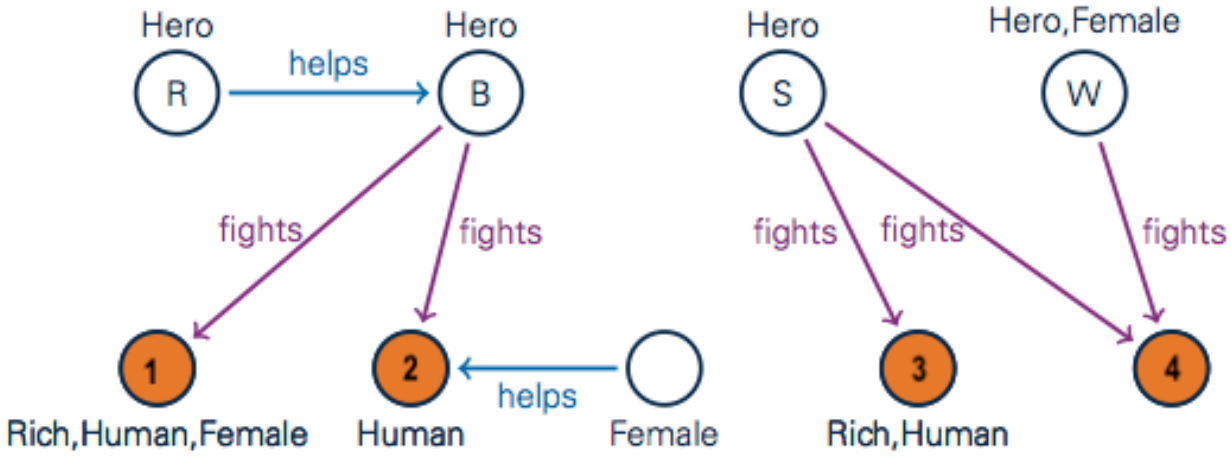
An interpretation $I = (\Delta^I, \cdot^I)$ consists of :

- 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)
  - $a^I \in \Delta^I$ (individuals interpreted as an element of Δ^I)

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 ?

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

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.

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

Does the following interpretation I satisfy this concept definition ?

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

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 $C \sqsubseteq D$ if $C^I \subseteq D^I$. I is a model of a TBox T if 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 $C \sqsubseteq D$ if $C^I \subseteq D^I$. I is a model of a TBox T if 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 $C \sqsubseteq D$ if $C^I \subseteq D^I$. I is a model of a TBox T if 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 $C \sqsubseteq D$ if $C^I \subseteq D^I$. I is a model of a TBox T if 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 $C \sqsubseteq D$ if $C^I \subseteq D^I$. I is a model of a TBox T if 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 ?

Equivalent or not ?

(1) $T_1 : \{A = A_1 \sqcap A_2\}$ and $T_2 : \{A = A_1, A = A_2\}$?

(2) $T_1' : \{A \sqsubseteq A_1 \sqcap A_2\}$ and $T_2' : \{A \sqsubseteq A_1, A \sqsubseteq A_2\}$?

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 $O = (\mathcal{T}, \mathcal{A})$ if 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

TBox or ABox axioms ?

```
: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 .
```

¬hasWife(Bill,Mary)

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

TBox or ABox axioms ?

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

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

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

Woman $\sqcap$ **Man** $\sqsubseteq \bot$

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

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

TBox or ABox axioms ?

```
: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 )
] .
```

TBox or ABox axioms ?

Person ⊓ ¬Parent (Jack)

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

TBox or ABox axioms ?

```
: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

TBox or ABox axioms ?

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

JohnsChildren ≡ ∃hasParent.{John}

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

NarcisticPerson ≡ ∃loves.Self

TBox or ABox axioms ?

≤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)

TBox or ABox axioms ?

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

=5 hasChild.⊤ (John)

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

MyBirthdayGuests ≡ {Bill, John, Mary}

TBox or ABox axioms ?

```
: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 .
```

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

TBox or ABox axioms ?

```
: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.

TBox or ABox axioms ?

```
: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 ]
    )
```

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

Inconsistency and incoherency

- An interpretation I satisfies $C \sqsubseteq D$ (resp. $A = C$) if $C^I \subseteq D^I$ (resp. $A^I = C^I$).
- I is a model of a TBox $\mathcal{T}$ if it satisfies all GCIs in $\mathcal{T}$.
- 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 $R(a, b) \in \mathcal{A}$.

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

Inconsistency : O is inconsistent if O has no model.

Given an ontology $O = (\mathcal{T}, \mathcal{A})$ and a concept C, if $C^I = \emptyset$ for any model I of O, we call C an unsatisfiable concept w.r.t. O.
e.g., $C := A \sqcap \neg A$ is an unsatisfiable concept w.r.t. any ontology.

Incoherency : O is incoherent if O has an unsatisfiable concept.

Inconsistency and incoherency

- An interpretation I satisfies $C \sqsubseteq D$ (resp. $A = C$) if $C^I \subseteq D^I$ (resp. $A^I = C^I$).
- I is a model of a TBox T if it satisfies all GCIs in T.
- 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 $R(a, b) \in \mathcal{A}$.

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

Inconsistency : O is inconsistent if O has no model.

Given an ontology $O = (\mathcal{T}, \mathcal{A})$ and a concept C, if $C^I = \emptyset$ for any model I of O, we call C an unsatisfiable concept w.r.t. O.
e.g., $C := A \sqcap \neg A$ is an unsatisfiable concept w.r.t. any ontology.

Incoherency : O is incoherent if O has an unsatisfiable concept.

Inconsistency and incoherency

- An interpretation I satisfies $C \sqsubseteq D$ (resp. $A = C$) if $C^I \subseteq D^I$ (resp. $A^I = C^I$).
- I is a model of a TBox T if it satisfies all GCIs in T.
- 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 $R(a, b) \in \mathcal{A}$.

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

Inconsistency : O is inconsistent if O has no model.

Given an ontology $O = (\mathcal{T}, \mathcal{A})$ and a concept C, if $C^I = \emptyset$ for any model I of O, we call C an unsatisfiable concept w.r.t. O.
e.g., $C := A \sqcap \neg A$ is an unsatisfiable concept w.r.t. any ontology.

Incoherency : O is incoherent if O has an unsatisfiable concept.

Inconsistency and incoherency

- An interpretation I satisfies $C \sqsubseteq D$ (resp. $A = C$) if $C^I \subseteq D^I$ (resp. $A^I = C^I$).
- I is a model of a TBox T if it satisfies all GCIs in T.
- 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 $R(a, b) \in \mathcal{A}$.

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

Inconsistency : O is inconsistent if O has no model.

Given an ontology $O = (\mathcal{T}, \mathcal{A})$ and a concept C, if $C^I = \emptyset$ for any model I of O, we call C an unsatisfiable concept w.r.t. O.
e.g., $C := A \sqcap \neg A$ is an unsatisfiable concept w.r.t. any ontology.

Incoherency : O is incoherent if O has an unsatisfiable concept.

Inconsistency and incoherency

- An interpretation I satisfies $C \sqsubseteq D$ (resp. $A = C$) if $C^I \subseteq D^I$ (resp. $A^I = C^I$).
- I is a model of a TBox T if it satisfies all GCIs in T.
- 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 $R(a, b) \in \mathcal{A}$.

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

Inconsistency : O is inconsistent if O has no model.

Given an ontology $O = (\mathcal{T}, \mathcal{A})$ and a concept C, if $C^I = \emptyset$ for any model I of O, we call C an unsatisfiable concept w.r.t. O.
e.g., $C := A \sqcap \neg A$ is an unsatisfiable concept w.r.t. any ontology.

Incoherency : O is incoherent if O has an unsatisfiable concept.

Inconsistency and incoherency

- An interpretation I satisfies $C \sqsubseteq D$ (resp. $A = C$) if $C^I \subseteq D^I$ (resp. $A^I = C^I$).
- I is a model of a TBox T if it satisfies all GCIs in T.
- 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 $R(a, b) \in \mathcal{A}$.

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

Inconsistency : O is inconsistent if O has no model.

Given an ontology $O = (\mathcal{T}, \mathcal{A})$ and a concept C, if $C^I = \emptyset$ for any model I of O, we call C an unsatisfiable concept w.r.t. O.
e.g., $C := A \sqcap \neg A$ is an unsatisfiable concept w.r.t. any ontology.

Incoherency : O is incoherent if O has an unsatisfiable concept.

Inconsistency and incoherency

- An interpretation I satisfies $C \sqsubseteq D$ (resp. $A = C$) if $C^I \subseteq D^I$ (resp. $A^I = C^I$).
- I is a model of a TBox T if it satisfies all GCIs in T.
- 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 $R(a, b) \in \mathcal{A}$.

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

Inconsistency : O is inconsistent if O has no model.

Given an ontology $O = (\mathcal{T}, \mathcal{A})$ and a concept C, if $C^I = \emptyset$ for any model I of O, we call C an unsatisfiable concept w.r.t. O.
e.g., $C := A \sqcap \neg A$ is an unsatisfiable concept w.r.t. any ontology.

Incoherency : O is incoherent if O has an unsatisfiable concept.

Inconsistency and incoherency

- An interpretation I satisfies $C \sqsubseteq D$ (resp. $A = C$) if $C^I \subseteq D^I$ (resp. $A^I = C^I$).
- I is a model of a TBox T if it satisfies all GCIs in T.
- 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 $R(a, b) \in \mathcal{A}$.

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

Inconsistency : O is inconsistent if O has no model.

Given an ontology $O = (\mathcal{T}, \mathcal{A})$ and a concept C, if $C^I = \emptyset$ for any model I of O, we call C an unsatisfiable concept w.r.t. O.
e.g., $C := A \sqcap \neg A$ is an unsatisfiable concept w.r.t. any ontology.

Incoherency : O is incoherent if O has an unsatisfiable concept.

Incoherency Example
Lab2 : reasoning with the ontology Pizza via protégé

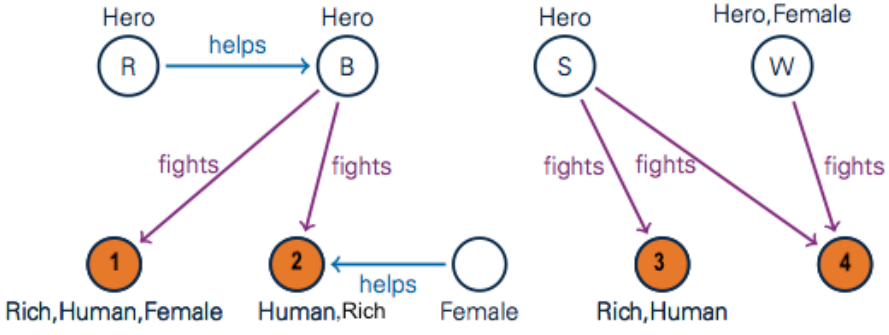
That ontology is consistent but incoherent !

Next : more reasoning tasks...

Terminological Reasoning of an Ontology (Definitions)

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

- 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.

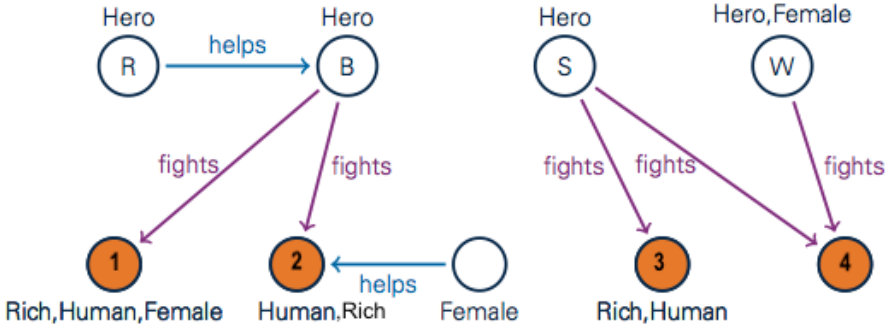


$\text{Rich}^I = \text{Huam}^I$ under I, but may not be always true under another model !!!

Terminological Reasoning of an Ontology (Definitions)

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

- 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.

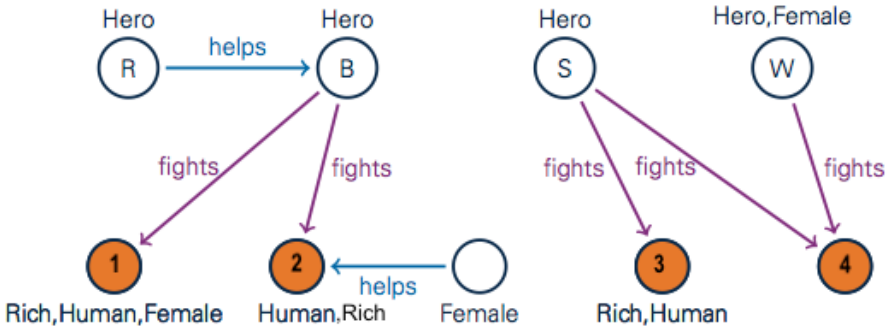


$\text{Rich}^I = \text{Huam}^I$ under I, but may not be always true under another model !!!

Terminological Reasoning of an Ontology (Definitions)

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

- 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.

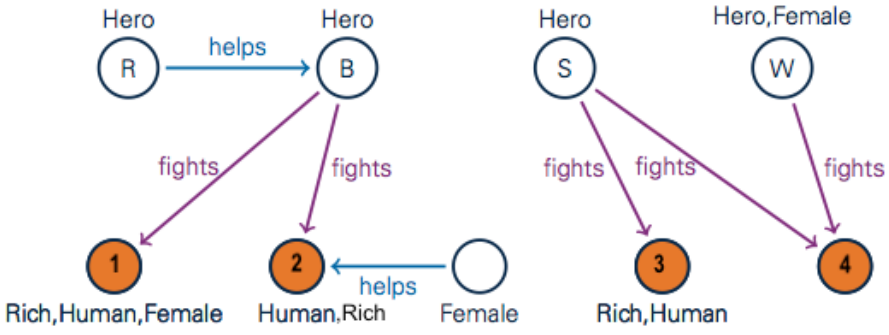


$\text{Rich}^I = \text{Huam}^I$ under I, but may not be always true under another model !!!

Terminological Reasoning of an Ontology (Definitions)

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

- 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.

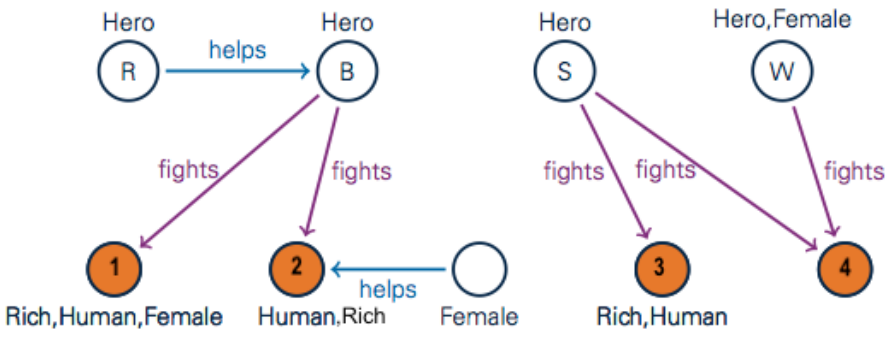


$\text{Rich}^I = \text{Huam}^I$ under I, but may not be always true under another model !!!

Terminological Reasoning of an Ontology (Definitions)

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

- 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.



$\text{Rich}^I = \text{Huam}^I$ under I, but may not be always true under another model !!!

Terminological Reasoning of an Ontology

Examples and Lemmas. Prove by definition...

- $A \sqcap \neg A$ is unsatisfiable (w.r.t. any ontology)
- **Lemma 1. If C is unsatisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O} \subseteq \mathcal{O}'$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is satisfiable (w.r.t. ontology=$\emptyset$ or ontology=$\{A \sqsubseteq \exists r.A\}$, etc.)
- **Lemma 2. If C is satisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O}' \subseteq \mathcal{O}$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is unsatisfiable w.r.t. ontology=$\{\top \sqsubseteq \exists r.\top\}$ for example : $r = hasParent$)
- $\forall r.A \sqcap \exists r.\neg A$ is unsatisfiable (w.r.t. any ontology)
- $\exists r.(A \sqcap B)$ is subsumbed by $\exists r.A$ (w.r.t. any ontology)
- $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$ (entailment)
  In other words, A is subsumbed by C w.r.t. $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq C\}$.
- $\{A \sqsubseteq \exists r.B, B \sqsubseteq C\} \models A \sqsubseteq \exists r.C$ (entailment)
  i.e., $A \sqsubseteq \exists r.C$ w.r.t. TBox= $\{A \sqsubseteq \exists r.B, B \sqsubseteq C\}$

Terminological Reasoning of an Ontology

Examples and Lemmas. Prove by definition...

- $A \sqcap \neg A$ is unsatisfiable (w.r.t. any ontology)
- **Lemma 1. If C is unsatisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O} \subseteq \mathcal{O}'$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is satisfiable (w.r.t. ontology=$\emptyset$ or ontology=$\{A \sqsubseteq \exists r.A\}$, etc.)
- **Lemma 2. If C is satisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O}' \subseteq \mathcal{O}$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is unsatisfiable w.r.t. ontology=$\{\top \sqsubseteq \exists r.\top\}$ for example : $r = hasParent$)
- $\forall r.A \sqcap \exists r.\neg A$ is unsatisfiable (w.r.t. any ontology)
- $\exists r.(A \sqcap B)$ is subsumbed by $\exists r.A$ (w.r.t. any ontology)
- $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$ (entailment)
  In other words, A is subsumbed by C w.r.t. $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq C\}$.
- $\{A \sqsubseteq \exists r.B, B \sqsubseteq C\} \models A \sqsubseteq \exists r.C$ (entailment)
  i.e., $A \sqsubseteq \exists r.C$ w.r.t. TBox= $\{A \sqsubseteq \exists r.B, B \sqsubseteq C\}$

Terminological Reasoning of an Ontology

Examples and Lemmas. Prove by definition...

- $A \sqcap \neg A$ is unsatisfiable (w.r.t. any ontology)
- **Lemma 1. If C is unsatisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O} \subseteq \mathcal{O}'$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is satisfiable (w.r.t. ontology$=\emptyset$ or ontology$=\{A \sqsubseteq \exists r.A\}$, etc.)
- **Lemma 2. If C is satisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O}' \subseteq \mathcal{O}$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is unsatisfiable w.r.t. ontology$=\{\top \sqsubseteq \exists r.\top\}$ for example : $r = hasParent$)
- $\forall r.A \sqcap \exists r.\neg A$ is unsatisfiable (w.r.t. any ontology)
- $\exists r.(A \sqcap B)$ is subsumbed by $\exists r.A$ (w.r.t. any ontology)
- $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$ (entailment)
  In other words, A is subsumbed by C w.r.t. $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq C\}$.
- $\{A \sqsubseteq \exists r.B, B \sqsubseteq C\} \models A \sqsubseteq \exists r.C$ (entailment)
  i.e., $A \sqsubseteq \exists r.C$ w.r.t. TBox$= \{A \sqsubseteq \exists r.B, B \sqsubseteq C\}$

Terminological Reasoning of an Ontology

Examples and Lemmas. Prove by definition...

- $A \sqcap \neg A$ is unsatisfiable (w.r.t. any ontology)
- **Lemma 1. If C is unsatisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O} \subseteq \mathcal{O}'$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is satisfiable (w.r.t. ontology$=\emptyset$ or ontology$=\{A \sqsubseteq \exists r.A\}$, etc.)
- **Lemma 2. If C is satisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O}' \subseteq \mathcal{O}$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is unsatisfiable w.r.t. ontology$=\{\top \sqsubseteq \exists r.\top\}$ for example : $r = hasParent$)
- $\forall r.A \sqcap \exists r.\neg A$ is unsatisfiable (w.r.t. any ontology)
- $\exists r.(A \sqcap B)$ is subsumbed by $\exists r.A$ (w.r.t. any ontology)
- $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$ (entailment)
  In other words, A is subsumbed by C w.r.t. $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq C\}$.
- $\{A \sqsubseteq \exists r.B, B \sqsubseteq C\} \models A \sqsubseteq \exists r.C$ (entailment)
  i.e., $A \sqsubseteq \exists r.C$ w.r.t. TBox$= \{A \sqsubseteq \exists r.B, B \sqsubseteq C\}$

Terminological Reasoning of an Ontology

Examples and Lemmas. Prove by definition...

- $A \sqcap \neg A$ is unsatisfiable (w.r.t. any ontology)
- **Lemma 1. If C is unsatisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O} \subseteq \mathcal{O}'$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is satisfiable (w.r.t. ontology$=\emptyset$ or ontology$=\{A \sqsubseteq \exists r.A\}$, etc.)
- **Lemma 2. If C is satisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O}' \subseteq \mathcal{O}$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is unsatisfiable w.r.t. ontology$=\{\top \sqsubseteq \exists r.\top\}$

for example : $r = hasParent$)

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

  In other words, A is subsumbed by C w.r.t. $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq C\}$.
- $\{A \sqsubseteq \exists r.B, B \sqsubseteq C\} \models A \sqsubseteq \exists r.C$ (entailment)

  i.e., $A \sqsubseteq \exists r.C$ w.r.t. TBox$= \{A \sqsubseteq \exists r.B, B \sqsubseteq C\}$

Terminological Reasoning of an Ontology

Examples and Lemmas. Prove by definition...

- $A \sqcap \neg A$ is unsatisfiable (w.r.t. any ontology)
- **Lemma 1. If C is unsatisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O} \subseteq \mathcal{O}'$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is satisfiable (w.r.t. ontology$=\emptyset$ or ontology$=\{A \sqsubseteq \exists r.A\}$, etc.)
- **Lemma 2. If C is satisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O}' \subseteq \mathcal{O}$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is unsatisfiable w.r.t. ontology$=\{\top \sqsubseteq \exists r.\top\}$ for example : $r = \mathit{hasParent}$)
- $\forall r.A \sqcap \exists r.\neg A$ is unsatisfiable (w.r.t. any ontology)
- $\exists r.(A \sqcap B)$ is subsumbed by $\exists r.A$ (w.r.t. any ontology)
- $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$ (entailment)
  In other words, A is subsumbed by C w.r.t. $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq C\}$.
- $\{A \sqsubseteq \exists r.B, B \sqsubseteq C\} \models A \sqsubseteq \exists r.C$ (entailment)
  i.e., $A \sqsubseteq \exists r.C$ w.r.t. TBox$= \{A \sqsubseteq \exists r.B, B \sqsubseteq C\}$

Terminological Reasoning of an Ontology

Examples and Lemmas. Prove by definition...

- $A \sqcap \neg A$ is unsatisfiable (w.r.t. any ontology)
- **Lemma 1. If C is unsatisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O} \subseteq \mathcal{O}'$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is satisfiable (w.r.t. ontology$=\emptyset$ or ontology$=\{A \sqsubseteq \exists r.A\}$, etc.)
- **Lemma 2. If C is satisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O}' \subseteq \mathcal{O}$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is unsatisfiable w.r.t. ontology$=\{\top \sqsubseteq \exists r.\top\}$ for example : $r = \mathit{hasParent}$)
- $\forall r.A \sqcap \exists r.\neg A$ is unsatisfiable (w.r.t. any ontology)
- $\exists r.(A \sqcap B)$ is subsumbed by $\exists r.A$ (w.r.t. any ontology)
- $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$ (entailment)
  In other words, A is subsumbed by C w.r.t. $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq C\}$.
- $\{A \sqsubseteq \exists r.B, B \sqsubseteq C\} \models A \sqsubseteq \exists r.C$ (entailment)
  i.e., $A \sqsubseteq \exists r.C$ w.r.t. TBox$= \{A \sqsubseteq \exists r.B, B \sqsubseteq C\}$

Terminological Reasoning of an Ontology

Examples and Lemmas. Prove by definition...

- $A \sqcap \neg A$ is unsatisfiable (w.r.t. any ontology)
- **Lemma 1. If C is unsatisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O} \subseteq \mathcal{O}'$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is satisfiable (w.r.t. ontology$=\emptyset$ or ontology$=\{A \sqsubseteq \exists r.A\}$, etc.)
- **Lemma 2. If C is satisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O}' \subseteq \mathcal{O}$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is unsatisfiable w.r.t. ontology$=\{\top \sqsubseteq \exists r.\top\}$ for example : $r = \mathit{hasParent}$)
- $\forall r.A \sqcap \exists r.\neg A$ is unsatisfiable (w.r.t. any ontology)
- $\exists r.(A \sqcap B)$ is subsumbed by $\exists r.A$ (w.r.t. any ontology)
- $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$ (entailment)
  In other words, A is subsumbed by C w.r.t. $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq C\}$.
- $\{A \sqsubseteq \exists r.B, B \sqsubseteq C\} \models A \sqsubseteq \exists r.C$ (entailment)
  i.e., $A \sqsubseteq \exists r.C$ w.r.t. TBox$= \{A \sqsubseteq \exists r.B, B \sqsubseteq C\}$

Terminological Reasoning of an Ontology

Examples and Lemmas. Prove by definition...

- $A \sqcap \neg A$ is unsatisfiable (w.r.t. any ontology)
- **Lemma 1. If C is unsatisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O} \subseteq \mathcal{O}'$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is satisfiable (w.r.t. ontology$=\emptyset$ or ontology$=\{A \sqsubseteq \exists r.A\}$, etc.)
- **Lemma 2. If C is satisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O}' \subseteq \mathcal{O}$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is unsatisfiable w.r.t. ontology$=\{\top \sqsubseteq \exists r.\top\}$ for example : $r = \mathit{hasParent}$)
- $\forall r.A \sqcap \exists r.\neg A$ is unsatisfiable (w.r.t. any ontology)
- $\exists r.(A \sqcap B)$ is subsumbed by $\exists r.A$ (w.r.t. any ontology)
- $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$ (entailment)

  In other words, A is subsumbed by C w.r.t. $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq C\}$.
- $\{A \sqsubseteq \exists r.B, B \sqsubseteq C\} \models A \sqsubseteq \exists r.C$ (entailment)

  i.e., $A \sqsubseteq \exists r.C$ w.r.t. TBox$= \{A \sqsubseteq \exists r.B, B \sqsubseteq C\}$

Terminological Reasoning of an Ontology

Examples and Lemmas. Prove by definition...

- $A \sqcap \neg A$ is unsatisfiable (w.r.t. any ontology)
- **Lemma 1. If C is unsatisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O} \subseteq \mathcal{O}'$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is satisfiable (w.r.t. ontology$=\emptyset$ or ontology$=\{A \sqsubseteq \exists r.A\}$, etc.)
- **Lemma 2. If C is satisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O}' \subseteq \mathcal{O}$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is unsatisfiable w.r.t. ontology$=\{\top \sqsubseteq \exists r.\top\}$ for example : $r = \mathit{hasParent}$)
- $\forall r.A \sqcap \exists r.\neg A$ is unsatisfiable (w.r.t. any ontology)
- $\exists r.(A \sqcap B)$ is subsumbed by $\exists r.A$ (w.r.t. any ontology)
- $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$ (entailment)
  In other words, A is subsumbed by C w.r.t. $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq C\}$.
- $\{A \sqsubseteq \exists r.B, B \sqsubseteq C\} \models A \sqsubseteq \exists r.C$ (entailment)
  i.e., $A \sqsubseteq \exists r.C$ w.r.t. TBox$= \{A \sqsubseteq \exists r.B, B \sqsubseteq C\}$

Terminological Reasoning of an Ontology

Examples and Lemmas. Prove by definition...

- $A \sqcap \neg A$ is unsatisfiable (w.r.t. any ontology)
- **Lemma 1. If C is unsatisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O} \subseteq \mathcal{O}'$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is satisfiable (w.r.t. ontology$=\emptyset$ or ontology$=\{A \sqsubseteq \exists r.A\}$, etc.)
- **Lemma 2. If C is satisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O}' \subseteq \mathcal{O}$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is unsatisfiable w.r.t. ontology$=\{\top \sqsubseteq \exists r.\top\}$ for example : $r = \mathit{hasParent}$)
- $\forall r.A \sqcap \exists r.\neg A$ is unsatisfiable (w.r.t. any ontology)
- $\exists r.(A \sqcap B)$ is subsumbed by $\exists r.A$ (w.r.t. any ontology)
- $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$ (entailment)
  In other words, A is subsumbed by C w.r.t. $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq C\}$.
- $\{A \sqsubseteq \exists r.B, B \sqsubseteq C\} \models A \sqsubseteq \exists r.C$ (entailment)
  i.e., $A \sqsubseteq \exists r.C$ w.r.t. TBox$= \{A \sqsubseteq \exists r.B, B \sqsubseteq C\}$

Terminological Reasoning of an Ontology

Examples and Lemmas. Prove by definition...

- $A \sqcap \neg A$ is unsatisfiable (w.r.t. any ontology)
- **Lemma 1. If C is unsatisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O} \subseteq \mathcal{O}'$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is satisfiable (w.r.t. ontology$=\emptyset$ or ontology$=\{A \sqsubseteq \exists r.A\}$, etc.)
- **Lemma 2. If C is satisfiable w.r.t. an ontology $\mathcal{O}$, and $\mathcal{O}' \subseteq \mathcal{O}$, then C is _____ w.r.t. $\mathcal{O}'$.**
- $\forall r.A \sqcap \forall r.\neg A$ is unsatisfiable w.r.t. ontology$=\{\top \sqsubseteq \exists r.\top\}$ for example : $r = \mathit{hasParent}$)
- $\forall r.A \sqcap \exists r.\neg A$ is unsatisfiable (w.r.t. any ontology)
- $\exists r.(A \sqcap B)$ is subsumbed by $\exists r.A$ (w.r.t. any ontology)
- $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$ (entailment)
  In other words, A is subsumbed by C w.r.t. $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq C\}$.
- $\{A \sqsubseteq \exists r.B, B \sqsubseteq C\} \models A \sqsubseteq \exists r.C$ (entailment)
  i.e., $A \sqsubseteq \exists r.C$ w.r.t. TBox$= \{A \sqsubseteq \exists r.B, B \sqsubseteq C\}$

Terminological Reasoning of an Ontology

How about

$$\exists r.A \sqcap \exists r.B \sqsubseteq \exists r.(A \sqcap B)???$$

and

$$\forall r.A \sqcap \forall r.B \sqsubseteq \forall r.(A \sqcap B)???$$

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 :

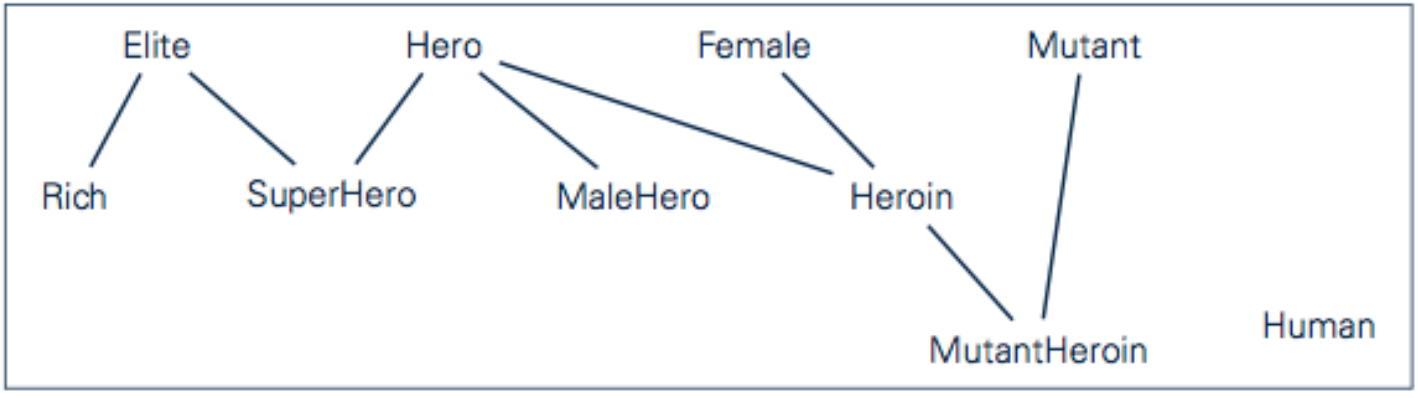
- a is an instance of a concept C w.r.t O if $a^I \in C^I$ for all models I of O
- $r(a, b)$ holds w.r.t O if $(a^I, b^I) \in r^I$ for all models I of O.

Let $O = \{A \sqsubseteq B, A(a)\}$. The we have $O \models B(a)$.

TBox Classification

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

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



Description Logics for Concept Modeling

DLs have been used to formalize different variants of conceptual modeling formalisms, and to provide automated reasoning in such formalisms (see also [Borgida & Brachman 2003]).

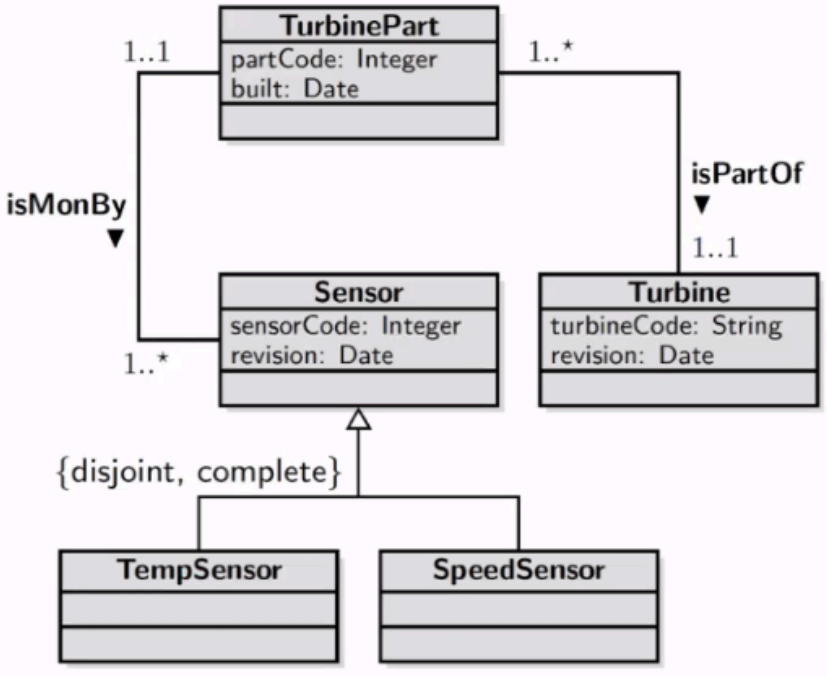
- ER schemas with cardinality constraints (but no hierarchies) [Lenzerini & Nobili 1990]
- Acyclic ER schemas with hierarchies [Bergamaschi & Sartori 1992]
- Arbitrary (possibly cyclic) ER schemas with hierarchies [C., Lenzerini, et al. 1994, 1999]
- UML Class Diagrams [Berardi et al. 2005], with OCL constraints [Queralt et al. 2012]
- ORM [Fillottrani, Keet, et al. 2015; Franconi, Mosca, et al. 2012; Sportelli & Franconi 2016]

In this way, DL reasoning can be used to provide support for various conceptual modeling activities.

While most of the work has been theoretical, also some prototype systems that provide reasoning support for conceptual modeling have been developed:

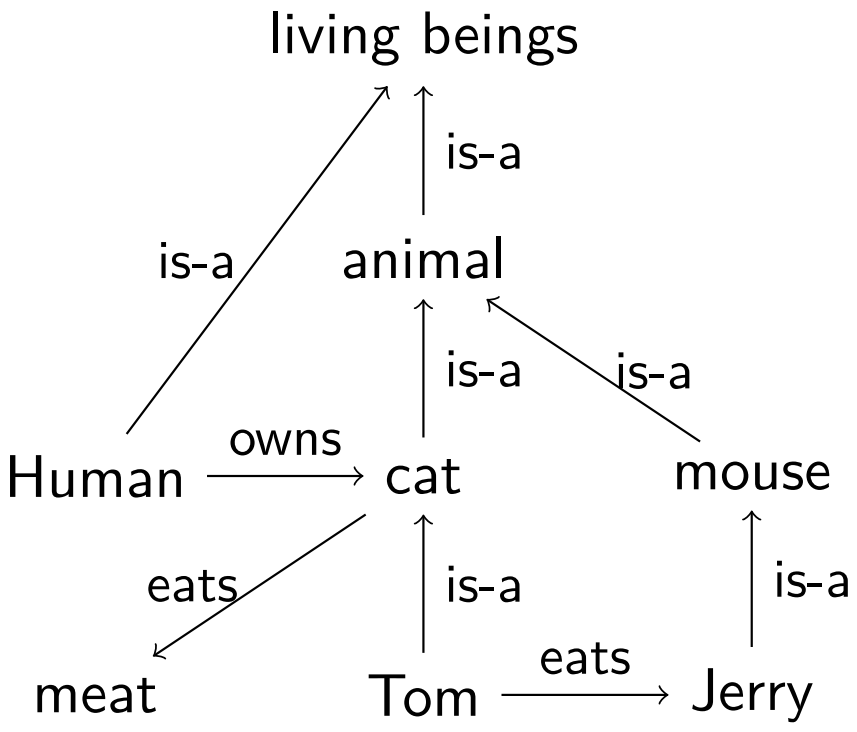
- ICOM conceptual modeling tool [Fillottrani, Franconi, et al. 2012, 2006; Franconi & Ng 2000]

Description Logics vs UML



$$
\begin{aligned}
\text{TempSensor} &\sqsubseteq \text{Sensor}\\
\text{SpeedSensor} &\sqsubseteq \text{Sensor}\\
\text{TempSensor} &\sqsubseteq \neg\text{SpeedSensor}\\
\text{Sensor} &\sqsubseteq \text{TempSensor} \sqcup \text{SpeedSensor}\\
\text{Turbine} &\sqsubseteq \forall \text{turbineCode}.\text{String} \sqcap\\
&\quad\ \exists \text{turbineCode} \sqcap \leq 1\, \text{turbineCode}\\
\exists \text{isMonBy} &\sqsubseteq \text{TurbinePart}\\
\exists \text{isMonBy}^{-} &\sqsubseteq \text{Sensor}\\
\text{TurbinePart} &\sqsubseteq \exists \text{isMonBy}\\
\text{Sensor} &\sqsubseteq \exists \text{isMonBy}^{-}\\
\exists \text{isPartOf} &\sqsubseteq \text{TurbinePart}\\
\exists \text{isPartOf}^{-} &\sqsubseteq \text{Turbine}\\
&(\textbf{funct}\ \text{isMonBy}^{-})\\
&(\textbf{funct}\ \text{isPartOf})\\
&\ldots
\end{aligned}
$$

Example : Semantic Network



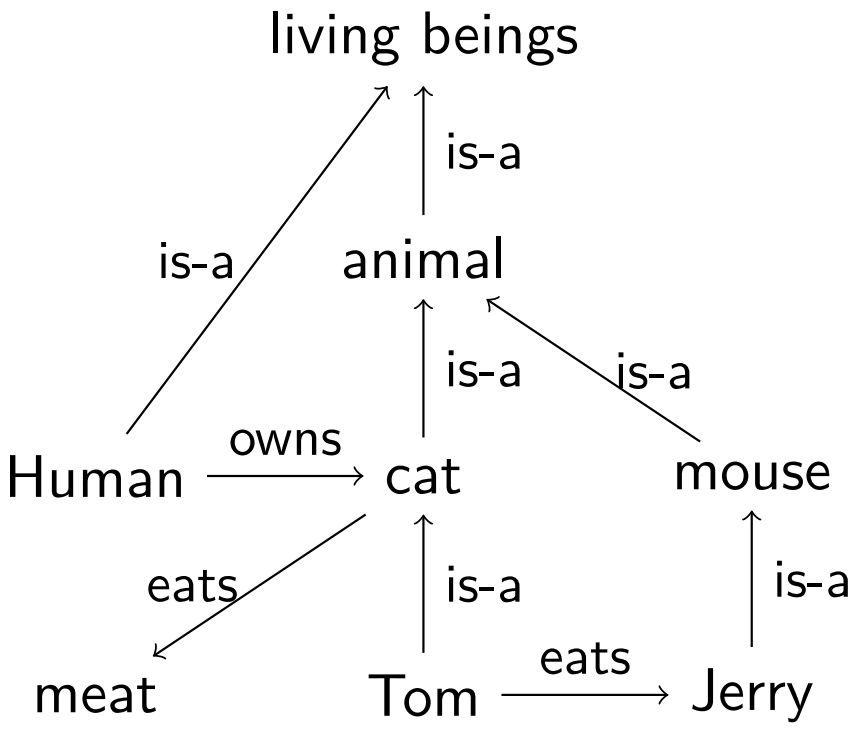
Conclusions

- Tom is an animal
- Tom eats a mouse
- similarities
  - Cats and mice are animals
  - Humans and mice are living being

Difficulties

- Does a herbivore count a cat ?
- Does a carnivore behave as a human ?
- Is Jerry meat ?
- Do cats like only meat, like humans do meat, or also mice ?
- What does *eat* mean ?

Example : Semantic Network



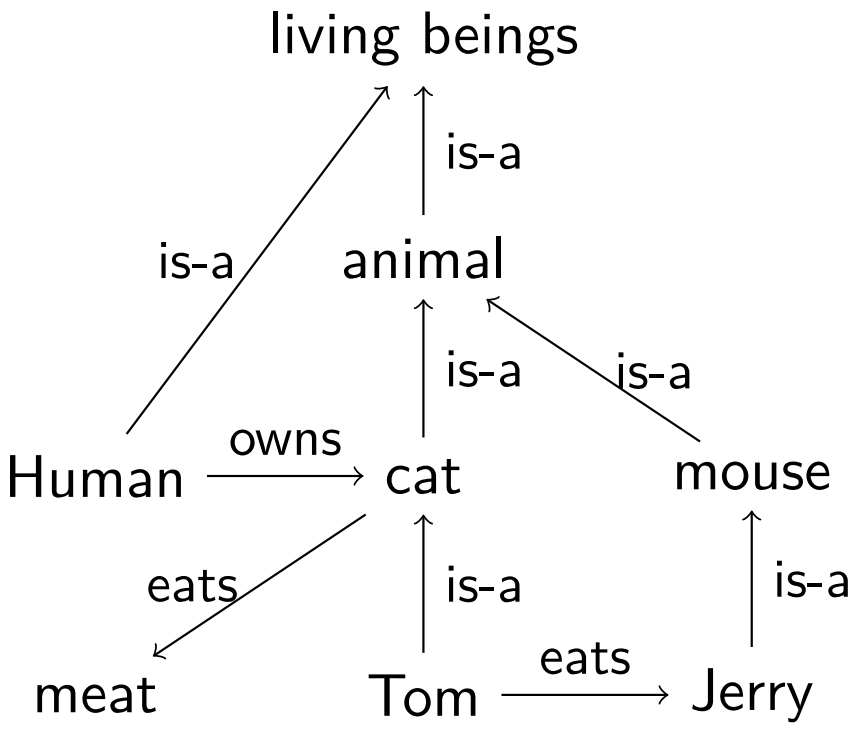
Conclusions

- Tom is an animal
- Tom eats a mouse
- similarities
  - Cats and mice are animals
  - Humans and mice are living being

Difficulties

- Does everyone own a cat ?
- Does everyone belong to a human ?
- Is Jerry meat ?
- Do cats eat only meat, animals, meat, or usually mice ?
- What does 'eat' mean ?

Example : Semantic Network



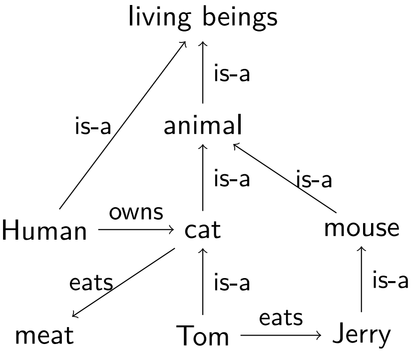
Conclusions

- Tom is an animal
- Tom eats a mouse
- similarities
  - Cats and mice are animals
  - Humans and mice are living being

Difficulties

- Does everyone own a cat ?
- Does every cat belong to a human ?
- Is Jerry meat ?
- Do cats eat only meat, or also fruits, or also only mice ?
- What does *eat* mean ?

Example : Semantic Network



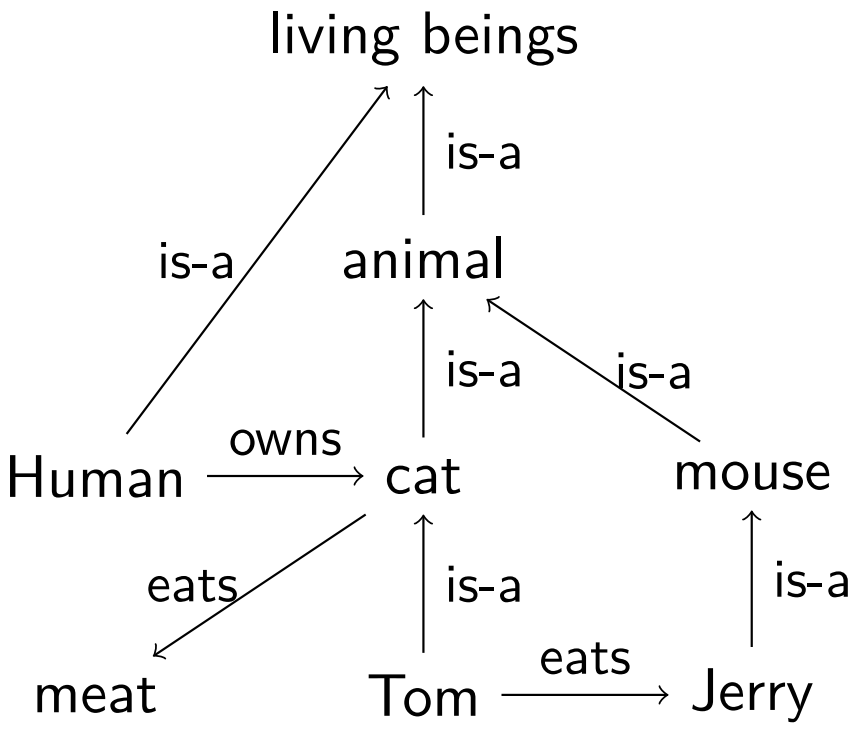
Conclusions

- Tom is an animal
- Tom eats a mouse
- similarities
  - Cats and mice are animals
  - Humans and mice are living being

Difficulties

- Does everyone own a cat?
- Does every cat belong to a human?
- Is Jerry meat?
- Do cats eat only meat, including mice, or usually meat?
- What does "is-a" mean?

Example : Semantic Network



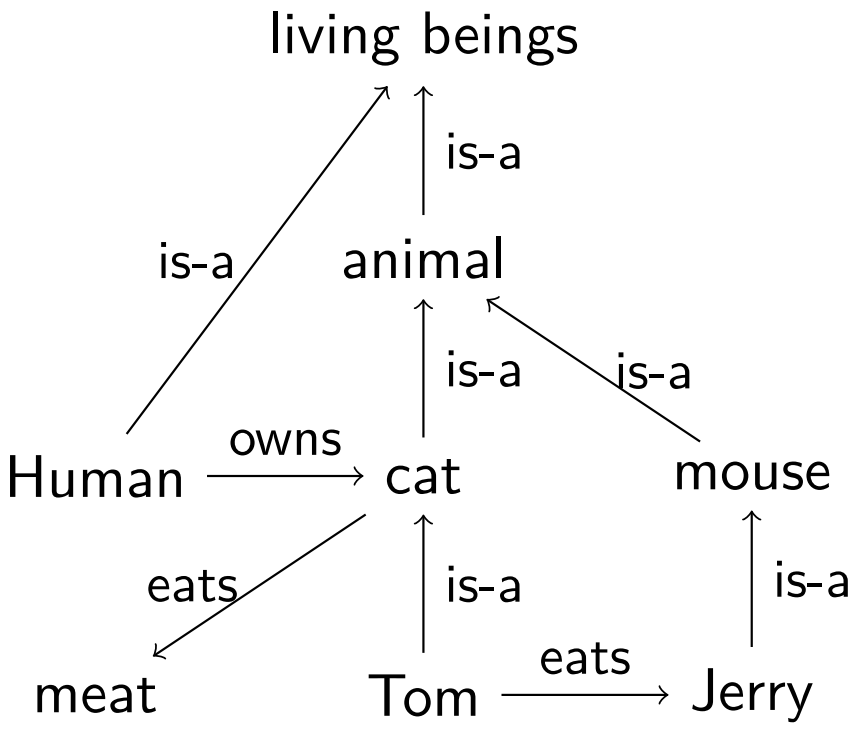
Conclusions

- Tom is an animal
- Tom eats a mouse
- similarities
  - Cats and mice are animals
  - Humans and mice are living being

Difficulties

- Does everyone own a cat ?
- Does every cat belong to a human ?
- Is Jerry meat ?
- Do cats eat only meat, including meat, or usually meat ?
- What does "is-a" mean ?

Example : Semantic Network



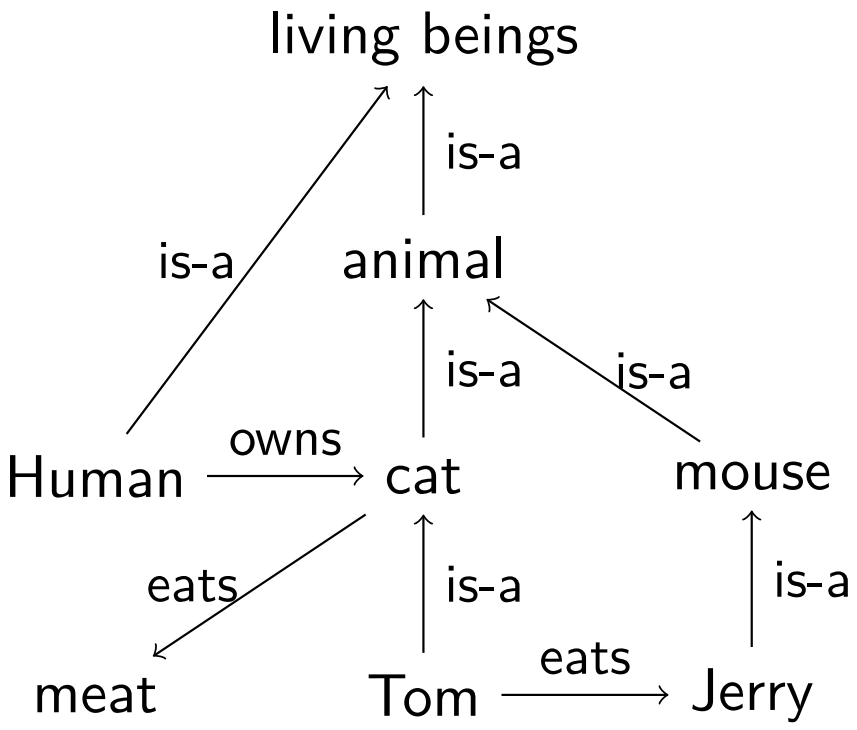
Conclusions

- Tom is an animal
- Tom eats a mouse
- similarities
  - Cats and mice are animals
  - Humans and mice are living being

Difficulties

- Does everyone own a cat ?
- Does every cat belong to a human ?
- Is Jerry meat ?
- Do cats eat only meat, including meat, or usually meat ?
- What does "is-a" mean ?

Example : Semantic Network



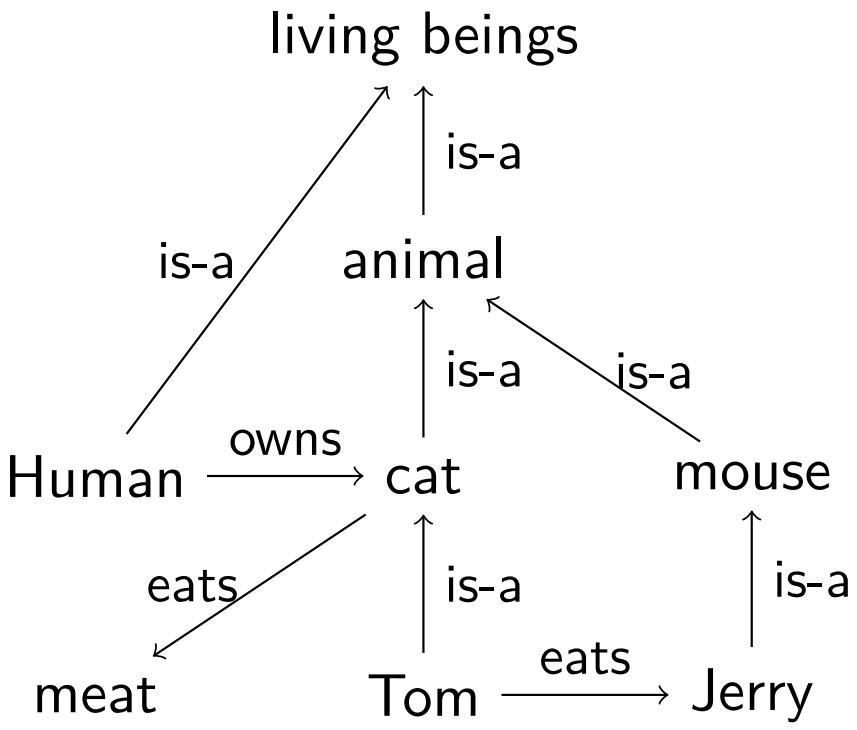
Conclusions

- Tom is an animal
- Tom eats a mouse
- similarities
  - Cats and mice are animals
  - Humans and mice are living being

Difficulties

- Does everyone own a cat ?
- Does every cat belong to a human ?
- Is Jerry meat ?
- Do cats eat only meat, including meat, or usually meat ?
- What does "is-a" mean ?

Example : Semantic Network



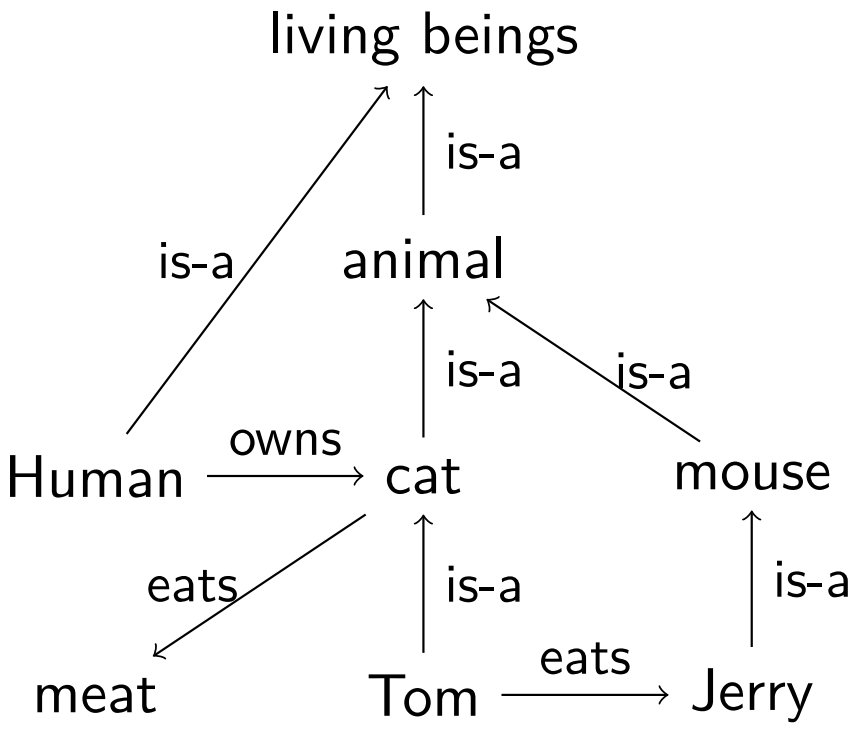
Conclusions

- Tom is an animal
- Tom eats a mouse
- similarities
  - Cats and mice are animals
  - Humans and mice are living being

Difficulties

- Does everyone own a cat ?
- Does every cat belong to a human ?
- Is Jerry meat ?
- Do cats eat only meat, including meat, or usually meat ?
- What does "is-a" mean ?

Example : Semantic Network



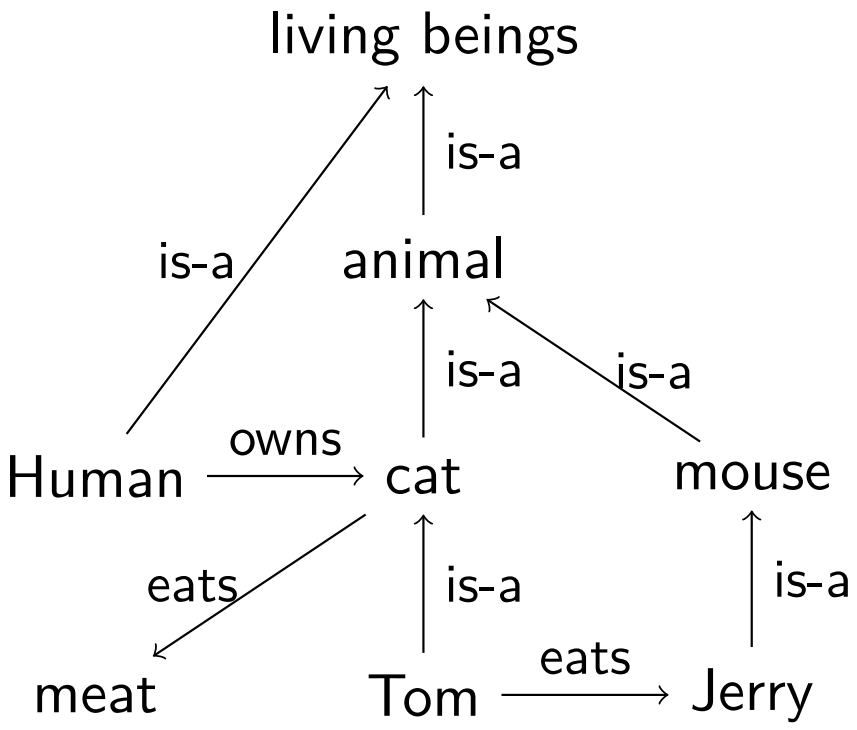
Conclusions

- Tom is an animal
- Tom eats a mouse
- similarities
  - Cats and mice are animals
  - Humans and mice are living being

Difficulties

- Does everyone own a cat ?
- Does every cat belong to a human ?
- Is Jerry meat ?
- Do cats eat only meat, including meat, or usually meat ?
- What does "is-a" mean ?

Example : Semantic Network



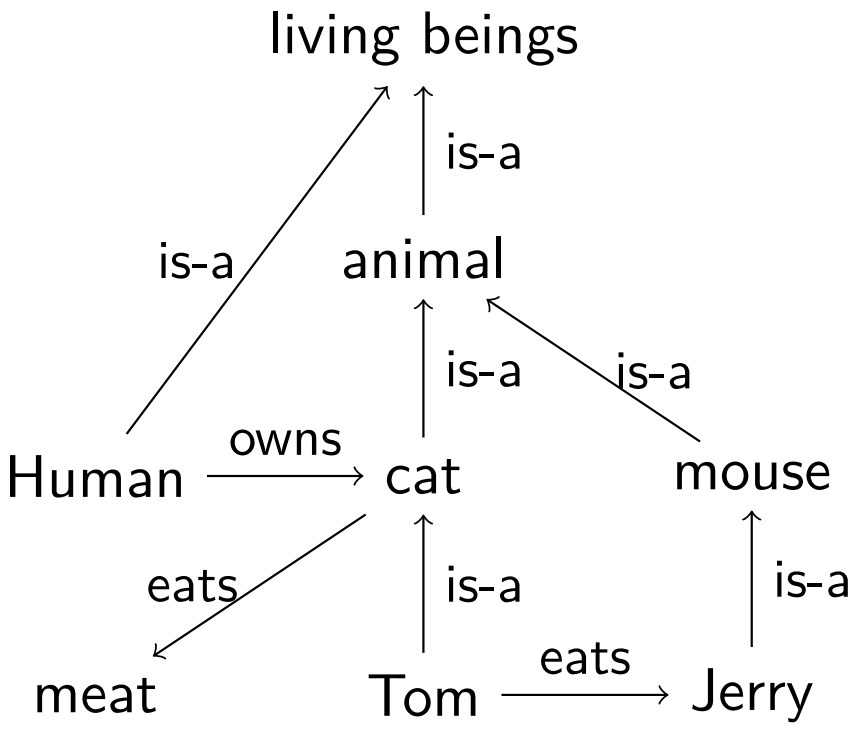
Conclusions

- Tom is an animal
- Tom eats a mouse
- similarities
  - Cats and mice are animals
  - Humans and mice are living being

Difficulties

- Does everyone own a cat ?
- Does every cat belong to a human ?
- Is Jerry meat ?
- Do cats eat only meat, including meat, or usually meat ?
- What does "is-a" mean ?

Example : Semantic Network



Conclusions

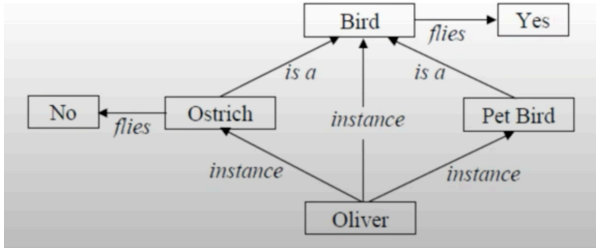
- Tom is an animal
- Tom eats a mouse
- similarities
  - Cats and mice are animals
  - Humans and mice are living being

Difficulties

- Does everyone own a cat ?
- Does every cat belong to a human ?
- Is Jerry meat ?
- Do cats eat only meat, including meat, or usually meat ?
- What does “is-a” mean ?

Problems of Semantic Networks

- ambiguity of is-a
  - individual → class : element
  - class → class : subset
  - "What is-a is and isn't" (Ron Brachman, 1983)
- inheritance as default
  - Any property can be overwritten
  - "Oliver flies or not ?"
  - no terminological information

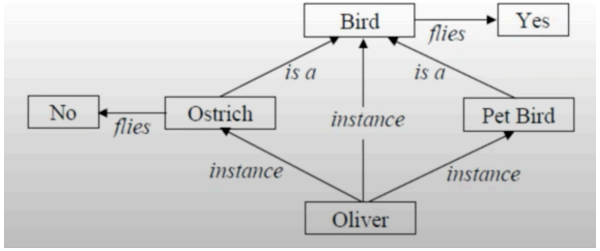


- "Semantic distance" depends on syntactic details
  - Tom → Cat → Animal
    ⇝ Tom is closely related to animal
  - Tom → Persian cat → Cat → Felines → Predator → Vertebrate → Animal
    ⇝ Tom is distantly related to Animal
- no formal semantics !
- Conflicts between user and programmer

Problems of Semantic Networks

- ambiguity of is-a
  - individual → class : element
  - class → class : subset
  - "What is-a is and isn't" (Ron Brachman, 1983)

- inheritance as default
  - Any property can be overwritten
  - "Oliver flies or not ?"
  - no terminological information

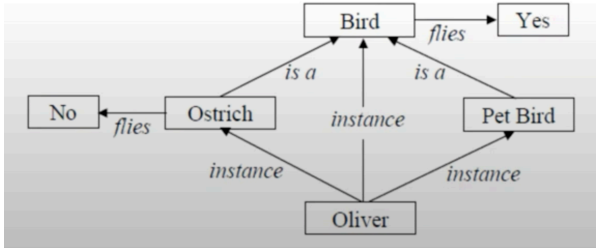


- "Semantic distance" depends on syntactic details
  - Tom → Cat → Animal
    ⇝ Tom is closely related to animal
  - Tom → Persian cat → Cat → Felines → Predator → Vertebrate → Animal
    ⇝ Tom is distantly related to Animal
- no formal semantics !
- Conflicts between user and programmer

Problems of Semantic Networks

- ambiguity of is-a
  - individual → class : element
  - class → class : subset
  - "What is-a is and isn't" (Ron Brachman, 1983)

- inheritance as default
  - Any property can be overwritten
  - "Oliver flies or not ?"
  - no terminological information

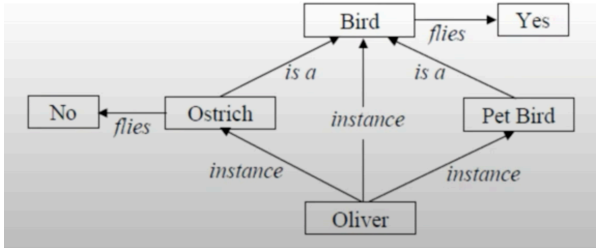


- "Semantic distance" depends on syntactic details
  - Tom → Cat → Animal
    ⇝ Tom is closely related to animal
  - Tom → Persian cat → Cat → Felines → Predator → Vertebrate → Animal
    ⇝ Tom is distantly related to Animal
- no formal semantics !
- Conflicts between user and programmer

Problems of Semantic Networks

- ambiguity of is-a
  - individual $\rightarrow$ class : element
  - class $\rightarrow$ class : subset
  - "What is-a is and isn't" (Ron Brachman, 1983)

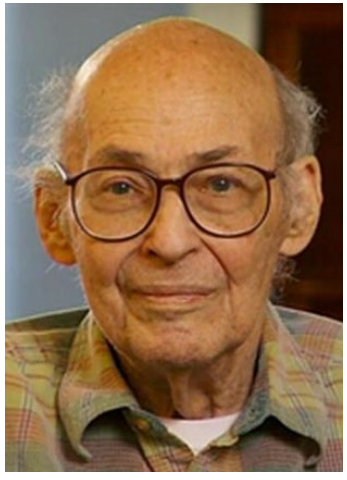
- inheritance as default
  - Any property can be overwritten
  - "Oliver flies or not ?"
  - no terminological information



- "Semantic distance" depends on syntactic details
  - Tom $\rightarrow$ Cat $\rightarrow$ Animal
    $\rightsquigarrow$ Tom is closely related to animal
  - Tom $\rightarrow$ Persian cat $\rightarrow$ Cat $\rightarrow$ Felines $\rightarrow$ Predator $\rightarrow$ Vertebrate $\rightarrow$ Animal
    $\rightsquigarrow$ Tom is distantly related to Animal
- no formal semantics !
- Conflicts between user and programmer

Frames : Origin and Intention

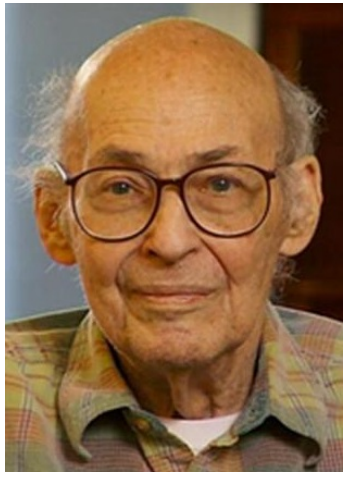
- KR formalism for describing entities
- developed in 1975 by Marvin Minsky
- Goal : Imitation of human categorization



Marvin Minsky (1927-2016)

Frames : Origin and Intention

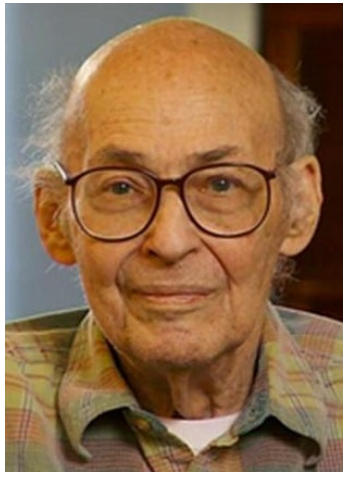
- KR formalism for describing entities
- developed in 1975 by Marvin Minsky
- Goal : Imitation of human categorization



Marvin Minsky (1927-2016)

Frames : Origin and Intention

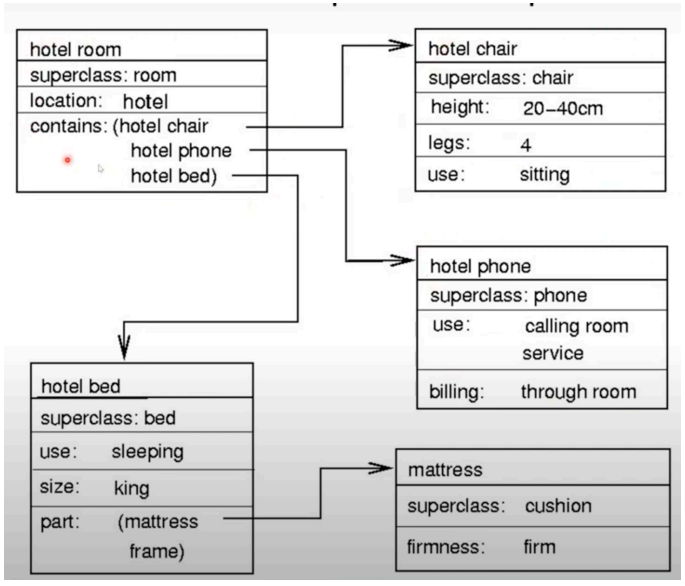
- KR formalism for describing entities
- developed in 1975 by Marvin Minsky
- Goal : Imitation of human categorization



Marvin Minsky (1927-2016)

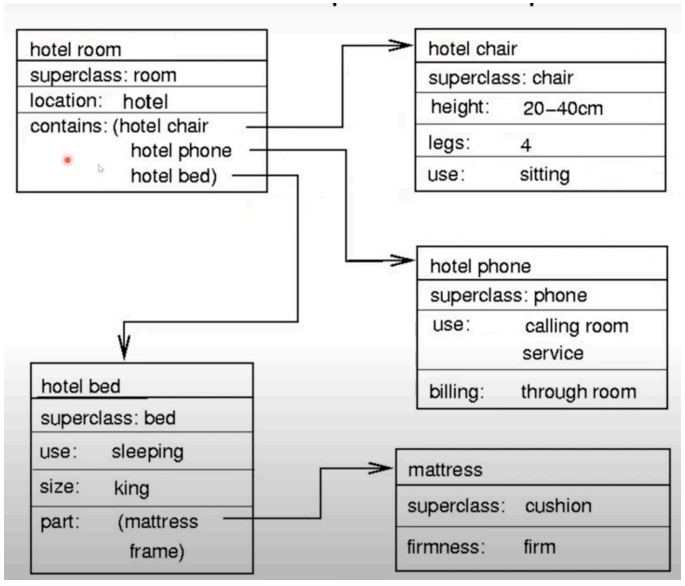
Frames : syntax and conclusions

- Syntax :
  - Class Frame describes class of entities
  - Instance Frame describes specific entities
  - Frame contains slots for certain attributes
  - value of a slot is called filler
  - Defaults for filler are possible
  - Sub-frames for specialization ($\rightsquigarrow$ is-a)
- Reasoning procedure
  - Inheritance : Sub-frames inherit default fillers
  - Criteria : If an instance frame I has suitable fillers for all slots of a class frame C, I is an instance C
  - Matching : Test whether an instance of a frame C_1 is also an instance of another frame C_2



Frames : syntax and conclusions

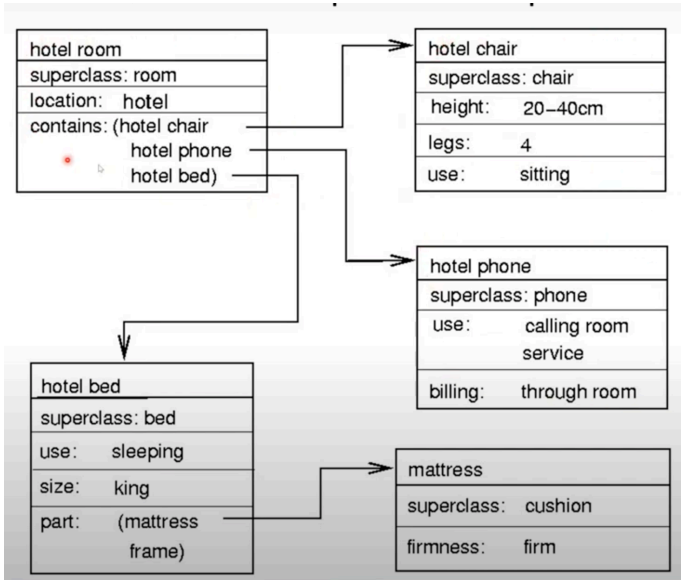
- Syntax :
  - Class Frame describes class of entities
  - Instance Frame describes specific entities
  - Frame contains slots for certain attributes
  - value of a slot is called filler
  - Defaults for filler are possible
  - Sub-frames for specialization ($\rightsquigarrow$ is-a)



- Reasoning procedure
  - Inheritance : Sub-frames inherit default fillers
  - Criteria : If an instance frame I has suitable fillers for all slots of a class frame C, I is an instance C
  - Matching : Test whether an instance of a frame C_1 is also an instance of another frame C_2

Frames : syntax and conclusions

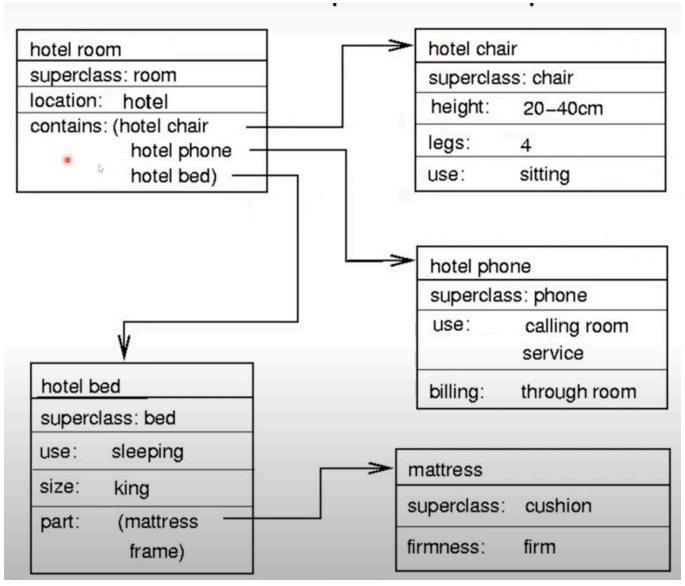
- Syntax :
  - Class Frame describes class of entities
  - Instance Frame describes specific entities
  - Frame contains slots for certain attributes
  - value of a slot is called filler
  - Defaults for filler are possible
  - Sub-frames for specialization ($\rightsquigarrow$ is-a)



- Reasoning procedure
  - Inheritance : Sub-frames inherit default fillers
  - Criteria : If an instance frame I has suitable fillers for all slots of a class frame C, I is an instance C
  - Matching : Test whether an instance of a frame C_1 is also an instance of another frame C_2

Frames : syntax and conclusions

- Syntax :
  - Class Frame describes class of entities
  - Instance Frame describes specific entities
  - Frame contains slots for certain attributes
  - value of a slot is called filler
  - Defaults for filler are possible
  - Sub-frames for specialization ($\rightsquigarrow$ is-a)



- Reasoning procedure
  - Inheritance : Sub-frames inherit default fillers
  - Criteria : If an instance frame I has suitable fillers for all slots of a class frame C, I is an instance C
  - Matching : Test whether an instance of a frame C_1 is also an instance of another frame C_2

Problems of frames

- Frame inferences can be expressed in predicate logic (Pat Hayes, 1979)
- Same as semantic networks
  - No taxonomy if all properties are just defaults
  - Without formal semantics, there is no "right" and "wrong"
- AI as a collection of heuristics
  - programming technology
  - more important for object-oriented programming than for KR

See also a lecture on Semantic Networks and Frames :
`https://www.youtube.com/watch?v=dU9WsMglBSM`

Problems of frames

- Frame inferences can be expressed in predicate logic (Pat Hayes, 1979)
- Same as semantic networks
  - No taxonomy if all properties are just defaults
  - Without formal semantics, there is no "right" and "wrong"
- AI as a collection of heuristics
  - programming technology
  - more important for object-oriented programming than for KR

See also a lecture on Semantic Networks and Frames :
`https://www.youtube.com/watch?v=dU9WsMglBSM`

Problems of frames

- Frame inferences can be expressed in predicate logic (Pat Hayes, 1979)
- Same as semantic networks
  - No taxonomy if all properties are just defaults
  - Without formal semantics, there is no "right" and "wrong"
- AI as a collection of heuristics
  - programming technology
  - more important for object-oriented programming than for KR

See also a lecture on Semantic Networks and Frames :
`https://www.youtube.com/watch?v=dU9WsMglBSM`

Problems of frames

- Frame inferences can be expressed in predicate logic (Pat Hayes, 1979)
- Same as semantic networks
  - No taxonomy if all properties are just defaults
  - Without formal semantics, there is no "right" and "wrong"
- AI as a collection of heuristics
  - programming technology
  - more important for object-oriented programming than for KR

See also a lecture on Semantic Networks and Frames :
`https://www.youtube.com/watch?v=dU9WsMglBSM`

Problems of frames

- Frame inferences can be expressed in predicate logic (Pat Hayes, 1979)
- Same as semantic networks
  - No taxonomy if all properties are just defaults
  - Without formal semantics, there is no “right” and “wrong”
- AI as a collection of heuristics
  - programming technology
  - more important for object-oriented programming than for KR

See also a lecture on Semantic Networks and Frames :
`https://www.youtube.com/watch?v=dU9WsMglBSM`

Problems of frames

- Frame inferences can be expressed in predicate logic (Pat Hayes, 1979)
- Same as semantic networks
  - No taxonomy if all properties are just defaults
  - Without formal semantics, there is no “right” and “wrong”
- AI as a collection of heuristics
  - programming technology
  - more important for object-oriented programming than for KR

See also a lecture on Semantic Networks and Frames :
`https://www.youtube.com/watch?v=dU9WsMglBSM`

Problems of frames

- Frame inferences can be expressed in predicate logic (Pat Hayes, 1979)
- Same as semantic networks
  - No taxonomy if all properties are just defaults
  - Without formal semantics, there is no “right” and “wrong”
- AI as a collection of heuristics
  - programming technology
  - more important for object-oriented programming than for KR

See also a lecture on Semantic Networks and Frames :
`https://www.youtube.com/watch?v=dU9WsMglBSM`

Problems with early KR systems

lack of formal semantics unclear meaning of the network
unreliable reasoning procedures
incompatibility

Graph-based algorithms Result depends on irrelevant information

Consequence : logic-based KR systems

- formal semantics based on mathematical logic
- "Semantic Technologies"
- often : decidable fragments of predicate logic
- often : simpler, intuitive syntax, e.g. no variables

Problems with early KR systems

lack of formal semantics unclear meaning of the network
unreliable reasoning procedures
incompatibility

Graph-based algorithms Result depends on irrelevant information

Consequence : logic-based KR systems

- formal semantics based on mathematical logic
- "Semantic Technologies"
- often : decidable fragments of predicate logic
- often : simpler, intuitive syntax, e.g. no variables

Problems with early KR systems

lack of formal semantics unclear meaning of the network
unreliable reasoning procedures
incompatibility

Graph-based algorithms Result depends on irrelevant information

Consequence : logic-based KR systems

- formal semantics based on mathematical logic
- "Semantic Technologies"
- often : decidable fragments of predicate logic
- often : simpler, intuitive syntax, e.g. no variables

Problems with early KR systems

lack of formal semantics	unclear meaning of the network unreliable reasoning procedures incompatibility
Graph-based algorithms	Result depends on irrelevant information

Consequence : logic-based KR systems

- formal semantics based on mathematical logic
- "Semantic Technologies'
- often : decidable fragments of predicate logic
- often : simpler, intuitive syntax, e.g. no variables

Problems with early KR systems

lack of formal semantics	unclear meaning of the network unreliable reasoning procedures incompatibility
Graph-based algorithms	Result depends on irrelevant information

Consequence : logic-based KR systems

- formal semantics based on mathematical logic
- "Semantic Technologies'
- often : decidable fragments of predicate logic
- often : simpler, intuitive syntax, e.g. no variables

Problems with early KR systems

lack of formal semantics	unclear meaning of the network unreliable reasoning procedures incompatibility
Graph-based algorithms	Result depends on irrelevant information

Consequence : logic-based KR systems

- formal semantics based on mathematical logic
- "Semantic Technologies'
- often : decidable fragments of predicate logic
- often : simpler, intuitive syntax, e.g. no variables

Problems with early KR systems

lack of formal semantics	unclear meaning of the network unreliable reasoning procedures incompatibility
Graph-based algorithms	Result depends on irrelevant information

Consequence : logic-based KR systems

- formal semantics based on mathematical logic
- "Semantic Technologies'
- often : decidable fragments of predicate logic
- often : simpler, intuitive syntax, e.g. no variables

Prehistory of Description Logics

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

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

We will study some of these algorithms...

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

We will study some of these algorithms...

Summary

- Introduction to DLs
- Decidability, complexity, reasoning algorithms (following lectures)