Description Logics

- Basic Principles
- A Simple DL: $\mathcal{ALC}$
- Syntax and Semantics of $\mathcal{ALC}$
- Reasoning in DLs

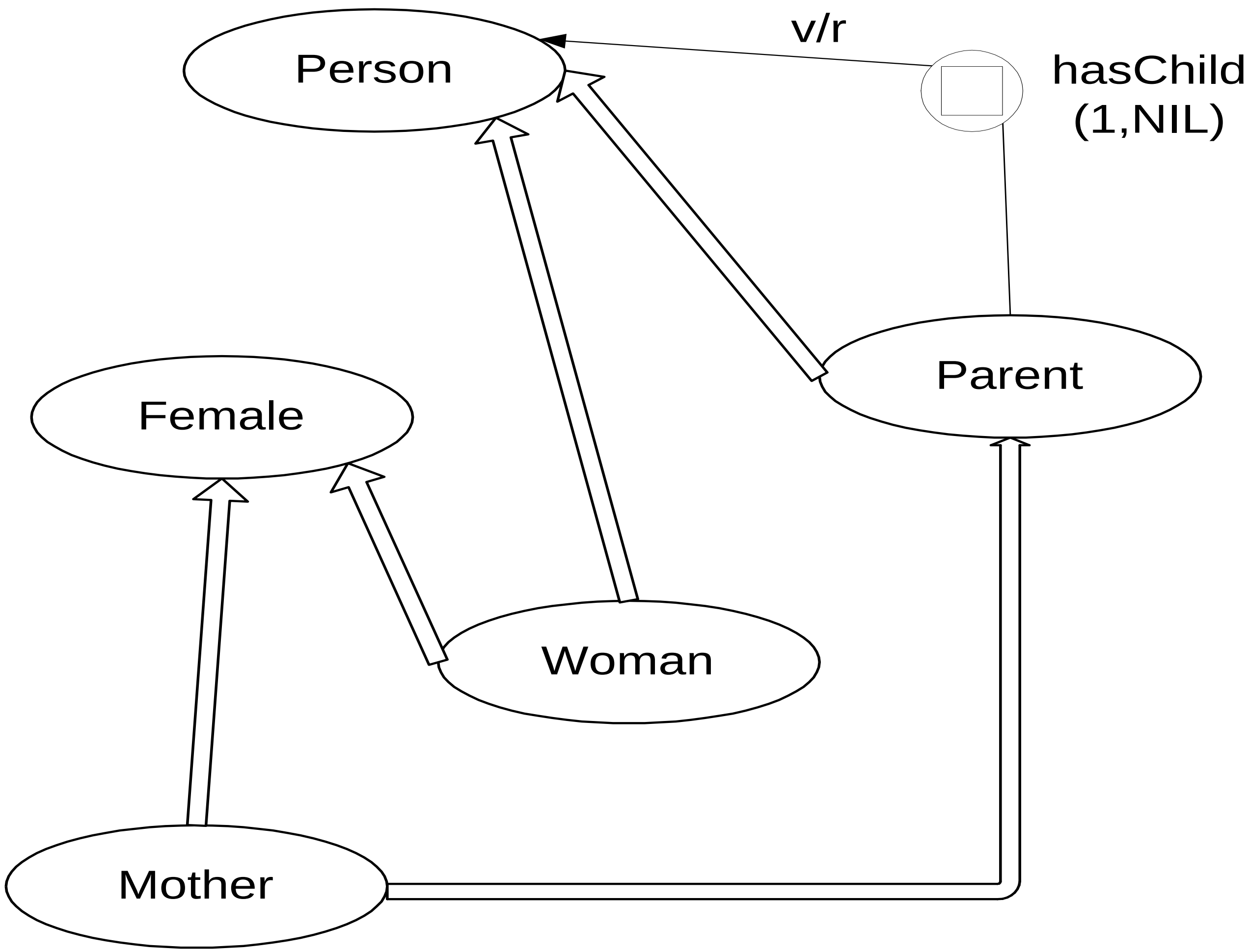
DLs: Some History

- The origins of DLs lie in research on **semantic networks** and **frames**. DLs are languages for describing the **nature and structure of objects.**
- The DL approach to KR was developed in the 80's and 90's in parallel with pure FOL approaches and other languages for structured objects like Telos and F-logic. Recently, DLs have been used to provide the **foundations for ontology languages** for the Web e.g., OWL.
- DLs have been developed as logics of **concepts** or **terms.** They are also known as **terminological languages** or **concept languages.**

DLs: Some History (cont'd)

- It all started with work on KL-ONE by Ron Brachman and colleagues. KL-ONE is the root of the family of DLs.
- There is currently a great body of theoretical work in DLs and many implemented DL systems (see `www.dl.kr.org`).

An Example of a Kl-One Network



DLs are Logics for Knowledge Representation

For each DL of interest, we will define:

- Syntax
- Semantics
- Reasoning (inference, proof-theory)

We will use DLs to represent knowledge about a domain of interest.

Syntax

- Three disjoint alphabets of symbols: **atomic concepts, atomic roles** and **individuals**.

  Concepts and roles should be understood as the equivalent of **classes** and **properties** or **relationships** in other languages.

  Individuals should be understood as the equivalent of **objects** in other languages.

- More complex concepts and roles are built from the basic symbols using **constructors**:
  - **conjunction, disjunction and negation of concepts**
  - **value restrictions**
  - **number restrictions**
  - ...

Examples - Syntax

- Atomic concepts: `Person, Male, Female`
- Atomic role: `child`
- Individual: `ANNA`

Examples - Syntax (cont'd)

Complex concepts:

- $\texttt{Person} \sqcap \neg\texttt{Female}$
- $\texttt{Female} \sqcup \texttt{Male}$
- $\forall\texttt{child}.\texttt{Person}$
- $\exists\texttt{child}.\texttt{Person}$
- $(\geqslant 3\ \texttt{child})$
- $\exists\texttt{child}.\texttt{Person} \sqcap \forall\texttt{child}.\texttt{Person}$
- $(\geqslant 3\ \texttt{child}) \sqcap \texttt{Male}$

Note: The above constructors can be nicely read as: not, and, or, all, some, at least etc.

Examples - Syntax (cont'd)

- Assertions about individuals:
  - $\texttt{Female}(\texttt{ANNA})$
  - $(\texttt{Person} \sqcap \neg\texttt{Male})(\texttt{ANNA})$
  - $((\geqslant \texttt{3 child}) \sqcap \texttt{Male})(\texttt{JOHN})$

Syntactic Conventions:

- Individuals will be written in uppercase.
- Concepts start with an uppercase letter followed by a lowercase letter.
- Roles start with a lowercase letter.

Semantics

DL expressions are given semantics by introducing the notion of **interpretation** (similarly with FOL expressions):

- An interpretation has a **domain**.
- Concepts are interpreted as **subsets** of the domain.
- Roles are interpreted as **binary relations** over the domain.
- Individuals are mapped to **elements of the domain.**
- The semantics of complex DL expressions is defined by appropriate **set expressions** which refer to sets that give the semantics of the parts of these expressions (e.g., the semantics of conjunction is defined by set intersection).

Examples - Informal Semantics

- `Male`

  The set of male persons.

- `child`

  The set of pairs of individuals (x, y) such that y is a child of x.

- $\texttt{Person} \sqcap \neg\texttt{Female}$

  The set of individuals that are persons and are not female.

- $\exists\texttt{child.Person} \sqcap \forall\texttt{child.Person}$

  The set of individuals that have at least one child who is a person, and additionally, all of their children are persons.

- $(\geqslant 3\ \texttt{child}) \sqcap \texttt{Male}$

  The set of individuals that have at least 3 children, and additionally, they are male.

Knowledge Representation with DLs

- In DLs we make a clear distinction between **intensional knowledge** and **extensional knowledge**.
- A KB consists of two components: a **TBox** and an **Abox**.
  - **TBox:** intensional knowledge in the form of concepts (terms), their properties and their relations.
  - **Abox:** extensional (assertional) knowledge.
- TELL and ASK interface

TBox

In the TBox one defines concepts of the application domain, their properties and their relations:

Example:

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

$$\texttt{Male} \sqsubseteq \neg\texttt{Female}$$

ABox

In the ABox one makes assertions about the individuals in the application domain: **membership in classes** and **role filling.**

Example:

$$(\texttt{Person} \sqcap \texttt{Female})(\texttt{ANNA}), \quad \texttt{child}(\texttt{ANNA}, \texttt{JOHN})$$

Reasoning and Proof Theory in DLs

Like other logics, DLs have their specialized inference rules, proof procedures etc. We will see proof procedures based on tableaus.

The following reasoning tasks have also been studied in the literature:

- Subsumption and classification
- Concept satisfiability
- Instance checking
- Knowledge base consistency
- Realization
- Retrieval

$\mathcal{ALC}$: The Smallest Propositionally Closed DL

Syntax	Semantics	Terminology
A	$A^{\mathcal{I}} \subseteq \Delta$	atomic concept
R	$R^{\mathcal{I}} \subseteq \Delta \times \Delta$	atomic role
$\top$	Δ	top (universal) concept
$\bot$	$\emptyset$	bottom concept
$\neg C$	$\Delta \setminus C^{\mathcal{I}}$	concept complement
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	concept conjunction
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	concept disjunction
$\forall R.C$	$\{x \mid (\forall y)((x,y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}})\}$	universal restriction
$\exists R.C$	$\{x \mid (\exists y)((x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\}$	existential restriction

$\mathcal{ALC}$ Syntax

To define the syntax of $\mathcal{ALC}$, we start with the following three disjoint alphabets:

- **Concept names**
- **Role names**
- **Individual names**

Concept names and role names are also called **atomic concepts** and **atomic roles**.

$\mathcal{ALC}$ Syntax: Concepts

The set of **concept expressions** or just **concepts** is defined inductively as follows:

1. $\top$ (**top concept**) and $\bot$ (**bottom concept**) are concepts.
2. Every **concept name** is a concept.
3. If C and D are concepts and R is a role name then the following are concepts:
   - $\neg C$ (**complement** of C)
   - $C \sqcap D$ (**conjunction** of C and D)
   - $C \sqcup D$ (**disjunction** of C and D)
   - $\forall R.C$ (**universal restriction**)
   - $\exists R.C$ (**existential restriction**)
4. Nothing else is a concept.

$\mathcal{ALC}$ Syntax: Concept Definitions

Let C be a concept **name** and D a concept.

Concept definitions are statements of the following forms:

- **Concept equivalences:** $C \equiv D$ which is read "C is defined to be equivalent to D".
- **Concept inclusions:** $C \sqsubseteq D$ which is read "C is subsumed by D".

Note: In the literature, concept equivalences are frequently written as $C \doteq D$.

Examples

- $\texttt{Student} \equiv \texttt{Person} \sqcap \exists \texttt{name.String} \sqcap$
  $\exists \texttt{address.String} \sqcap$
  $\exists \texttt{enrolled.Course}$

- $\texttt{Student} \sqsubseteq \exists \texttt{enrolled.Course}$

Intuitive Meaning of Concept Definitions

Concept definitions are used to introduce **symbolic names** for complex descriptions.

In a set of concept definitions, we distinguish between **name symbols** that occur in the left-hand side of a definition and **base symbols** that occur only on the right-hand side of some axioms.

Name symbols appearing in concept definitions are usually called **defined concepts** and base symbols **primitive concepts**.

Primitive vs. Defined Concepts

An important feature of DLs is their ability to **distinguish** primitive from defined concepts:

- **Defined concepts** have necessary and sufficient conditions for concept membership.

  Examples: student, instructor, driver, white wine etc.

- **Primitive concepts** cannot be defined or need not be defined. However, we might know some necessary (but not sufficient) conditions for membership.

  Examples: dog (or any other natural kind), wine (in a food and wine recommendation application).

Necessary Conditions

A **concept inclusion** of the form $C \sqsubseteq D$ states a **necessary condition** for membership in the concept C: For an individual to be in C, it is necessary that it is also in D (it has the properties expressed by D).

Example: $\texttt{Student} \sqsubseteq \exists\texttt{enrolled}.\texttt{Course}$

Concept inclusions express "if" statements.

Necessary and Sufficient Conditions

A **concept equivalence (definition)** of the form $C \equiv D$ states a **necessary and sufficient condition** for membership in the concept C: For an individual to be in C, it is necessary that it is also in D (it has the properties expressed by D). If an individual is in D, this is a sufficient condition for concluding that it is also in C.

Example: $\texttt{Student} \equiv \texttt{Person} \sqcap \exists \texttt{name}.\texttt{String} \sqcap \exists \texttt{address}.\texttt{String} \sqcap \exists \texttt{enrolled}.\texttt{Course}$

Concept equivalences express "if and only if" statements.

Example: Family Relationships

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

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

$$\texttt{Mother} \equiv \texttt{Woman} \sqcap \exists\texttt{child.Person}$$

$$\texttt{Father} \equiv \texttt{Man} \sqcap \exists\texttt{child.Person}$$

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

$$\texttt{Grandmother} \equiv \texttt{Mother} \sqcap \exists\texttt{child.Parent}$$

$$\texttt{MotherWithoutDaughter} \equiv \texttt{Mother} \sqcap \forall\texttt{child.}\neg\texttt{Woman}$$

$$\texttt{Wife} \equiv \texttt{Woman} \sqcap \exists\texttt{husband.Man}$$

$\mathcal{ALC}$ Syntax: Terminological Axioms

Terminological axioms are formulas of the forms $C \equiv D$ or $C \sqsubseteq D$ where C and D are concepts.

Examples:

- $\texttt{Student} \equiv \texttt{Person} \sqcap \exists \texttt{name.String} \sqcap \exists \texttt{address.String} \sqcap \exists \texttt{enrolled.Course}$

- $\texttt{Student} \sqsubseteq \exists \texttt{enrolled.Course}$

- $\texttt{Male} \sqsubseteq \neg \texttt{Female}$

Note: Concept definitions are terminological axioms in which the left concept is an atomic concept name.

Terminological Axioms (cont'd)

Terminological axioms are useful for expressing properties of concepts and roles. For example:

- **Disjointness of concepts:** $\texttt{Male} \sqsubseteq \neg\texttt{Female}$
- **Coverings:** $\top \sqsubseteq \texttt{Male} \sqcup \texttt{Female}$
- **Domain restrictions:** $\exists\texttt{child}.\top \sqsubseteq \texttt{Parent}$
- **Range restrictions:** $\top \sqsubseteq \forall\texttt{child}.\texttt{Person}$
- ...

$\mathcal{ALC}$ Syntax: Assertions about Individuals

In a DL, one can also describe a specific state of affairs of an application domain in terms of **individuals**, concepts and roles. This is done by:

- **Concept assertions:** Statements of the form $C(a)$ where C is a concept and a is an individual.
- **Role assertions:** Statements of the form $R(a, b)$ where R is a role and a, b are individuals.

Examples of Assertions

- $\texttt{Student}(\texttt{JOHN})$
- $\texttt{enrolled}(\texttt{JOHN}, \texttt{CS415})$
- $(\texttt{Student} \sqcup \texttt{Professor})(\texttt{PAUL})$

TBoxes, ABoxes and Knowledge Bases

A **TBox** is a set of **terminological axioms** (including concept definitions).

An **Abox** is a set of concept and role **assertions**.

A **knowledge base** Σ is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an Abox.

$\mathcal{ALC}$ Semantics

Definition. An **interpretation** $\mathcal{I}$ is a pair $(\Delta, \cdot^{\mathcal{I}})$ which consists of:

- a nonempty set Δ (the **domain**)
- a function $\cdot^{\mathcal{I}}$ (the **interpretation function**) which maps
  - every individual a to $a^{\mathcal{I}} \in \Delta$
  - every concept C to a subset $C^{\mathcal{I}}$ of Δ
  - every role R to a subset $R^{\mathcal{I}}$ of $\Delta \times \Delta$

Unique Names Assumption (UNA): We will assume that if a and b are distinct individuals then $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.

Note that the UNA might not be assumed in other contexts e.g., OWL.

$\mathcal{ALC}$ Semantics (cont'd)

Then $\mathcal{I}$ is extended to arbitrary concepts as follows:

$$\top^{\mathcal{I}} = \Delta$$

$$\bot^{\mathcal{I}} = \emptyset$$

$$(\neg C)^{\mathcal{I}} = \Delta \setminus C^{\mathcal{I}}$$

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$$

$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$

$$(\forall R.C)^{\mathcal{I}} = \{ \ x \in \Delta \mid (\forall y)((x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}) \ \}$$

$$(\exists R.C)^{\mathcal{I}} = \{ \ x \in \Delta \mid (\exists y)((x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}) \ \}$$

$\mathcal{ALC}$ Semantics (cont'd)

Notice that $\mathcal{ALC}$ is a **propositionally closed** language:

- $\neg\top \equiv \bot$
- $\neg\bot \equiv \top$
- $\neg(C \sqcap D) \equiv \neg C \sqcup \neg D$
- $\neg(C \sqcup D) \equiv \neg C \sqcap \neg D$
- $\neg(\forall R.C) \equiv \exists R.\neg C$
- $\neg(\exists R.C) \equiv \forall R.\neg C$

TBox: Semantics

Satisfaction. Let $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ be an interpretation.

- $\mathcal{I}$ **satisfies** the statement $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- $\mathcal{I}$ **satisfies** the statement $C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$.

Model. An interpretation $\mathcal{I}$ is a **model** for a TBox $\mathcal{T}$ if $\mathcal{I}$ satisfies all the statements in $\mathcal{T}$.

Satisfiability. A TBox $\mathcal{T}$ is **satisfiable** if it has a model.

ABox: Semantics

Satisfaction. Let $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ be an interpretation.

- $\mathcal{I}$ **satisfies** $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$.
- $\mathcal{I}$ **satisfies** $R(a,b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

Model. An interpretation $\mathcal{I}$ is a **model** of an ABox $\mathcal{A}$ if it satisfies every assertion of $\mathcal{A}$.

Satisfiability. An ABox $\mathcal{A}$ is **satisfiable** if it has a model.

Knowledge Bases - Semantics

Satisfaction. An interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ **satisfies** a knowledge base $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ if $\mathcal{I}$ satisfies both $\mathcal{T}$ and $\mathcal{A}$.

Model. An interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ is a **model** of a knowledge base $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ if $\mathcal{I}$ is a model of $\mathcal{T}$ and $\mathcal{A}$.

Satisfiability. A knowledge base Σ is said to be **satisfiable** if it has a model.

Entailment (Logical Implication)

Definition. We will say that Σ **entails** ϕ (denoted by $\Sigma \models \varphi$) if every model of Σ is a model of φ.

Example:

TBox:

$\texttt{Female} \sqsubseteq \texttt{Person}$

ABox:

$\texttt{Female}(\texttt{ANNA})$

$\Sigma \models \texttt{Person}(\texttt{ANNA})$

Example

TBox:

$$\exists \texttt{teaches}.\texttt{Course} \sqsubseteq \neg\texttt{Undergrad} \sqcup \texttt{Professor}$$

ABox:

$\texttt{teaches}(\texttt{JOHN}, \texttt{CS415}), \texttt{Course}(\texttt{CS415}),$
$\texttt{Undergrad}(\texttt{JOHN})$

$$\Sigma \models \texttt{Professor}(\texttt{JOHN})$$

Example (cont'd))

There is nothing wrong with the entailment

$$\Sigma \models \texttt{Professor}(\texttt{JOHN})$$

since the TBox has no axiom that precludes somebody from being and undergrad and also a professor.

Example (Revisited)

TBox:

$\exists \texttt{teaches.Course} \sqsubseteq$
$\texttt{Undergrad} \sqcup \texttt{Professor}$

ABox:

$\texttt{teaches}(\texttt{JOHN}, \texttt{CS415}), \texttt{Course}(\texttt{CS415}),$
$\texttt{Undergrad}(\texttt{JOHN})$

$$\Sigma \overset{?}{\models} \texttt{Professor}(\texttt{JOHN})$$

$$\Sigma \overset{?}{\models} \neg\texttt{Professor}(\texttt{JOHN})$$

Example (Revisited)

TBox:

$\exists \texttt{child}.\top \sqsubseteq \texttt{Parent}$

$\top \sqsubseteq \forall \texttt{child}.\texttt{Person}$

ABox:

$\texttt{child}(\texttt{ANNA}, \texttt{JOHN})$

$\Sigma \models \texttt{Parent}(\texttt{ANNA})$

$\Sigma \models \texttt{Person}(\texttt{JOHN})$

Validity

Definition. A terminological axiom ϕ is **valid** if every interpretation is a model of ϕ.

Examples:

$$\mathtt{A} \sqcap \mathtt{B} \sqsubseteq \mathtt{A}, \quad \mathtt{A} \sqcap \mathtt{B} \sqcap \mathtt{C} \sqsubseteq \mathtt{A} \sqcap \mathtt{B}, \quad \forall \mathtt{R}.(\mathtt{A} \sqcap \mathtt{B}) \sqsubseteq \forall \mathtt{R}.\mathtt{A}$$

Definition. We will say that a knowledge base Σ is **valid** if every interpretation is a model of Σ.

Example:

TBox: $\forall \mathtt{R}.(\mathtt{A} \sqcap \mathtt{B}) \sqsubseteq \forall \mathtt{R}.\mathtt{A}$

ABox: empty

Reasoning Services

- **Concept Satisfiability**.

  This is the problem of checking whether a concept C is satisfiable with respect to a knowledge base Σ, i.e., whether there exists a model $\mathcal{I}$ of Σ such that $C^{\mathcal{I}} \neq \emptyset$.

  Equivalently: $\Sigma \not\models C \equiv \bot$

  Example: $\texttt{Student} \sqcap \neg\texttt{Person}$

- **Subsumption.**

  This is the problem of checking whether C is subsumed by D with respect to a knowledge base Σ, i.e., whether $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in every model $\mathcal{I}$ of Σ.

  Equivalently: $\Sigma \models C \sqsubseteq D$

  Example: $\texttt{Student} \sqsubseteq \texttt{Person}$

Reasoning Services (cont'd)

- **Knowledge base satisfiability.**

  This is the problem of checking whether Σ is satisfiable, i.e., whether it has a model.

  Example: $\texttt{Student} \equiv \neg\texttt{Person}$

- **Instance Checking.**

  $\Sigma \models C(a)$

  This is the problem of checking whether the assertion $C(a)$ is satisfied in every model of Σ.

  Example: $\texttt{Professor}(\texttt{JOHN})$

Reasoning Services (cont'd)

- **Retrieval** or **query answering.**

  Find all a such that $\{a \mid \Sigma \models C(a)\}$.

  Example: `Professor` $\Rightarrow$ `JOHN`

- **Realization.**

  Given an individual a, find the **most specific concepts** C such that $\Sigma \models C(a)$

  Example: `JOHN` $\Rightarrow$ `Professor`

Reduction to Satisfiability

- Concept Satisfiability

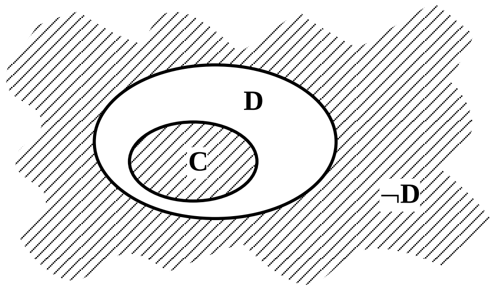
  $\Sigma \not\models C \equiv \bot \quad \leftrightarrow$

  exists x s.t. $\Sigma \cup \{C(x)\}$ has a model

- Subsumption

  $\Sigma \models C \sqsubseteq D \quad \leftrightarrow$

  exists x s.t. $\Sigma \cup \{(C \sqcap \neg D)(x)\}$ has no models

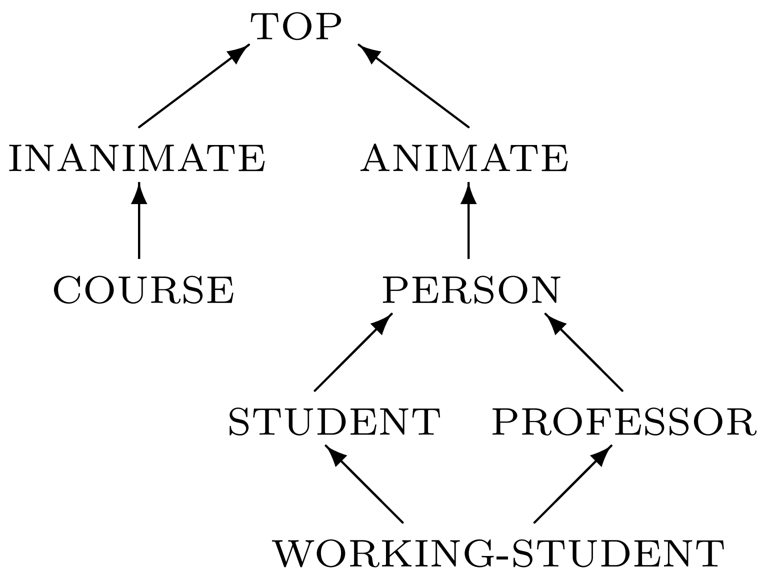


- Instance Checking

  $\Sigma \models C(a) \quad \leftrightarrow$

  $\Sigma \cup \{\neg C(a)\}$ has no models

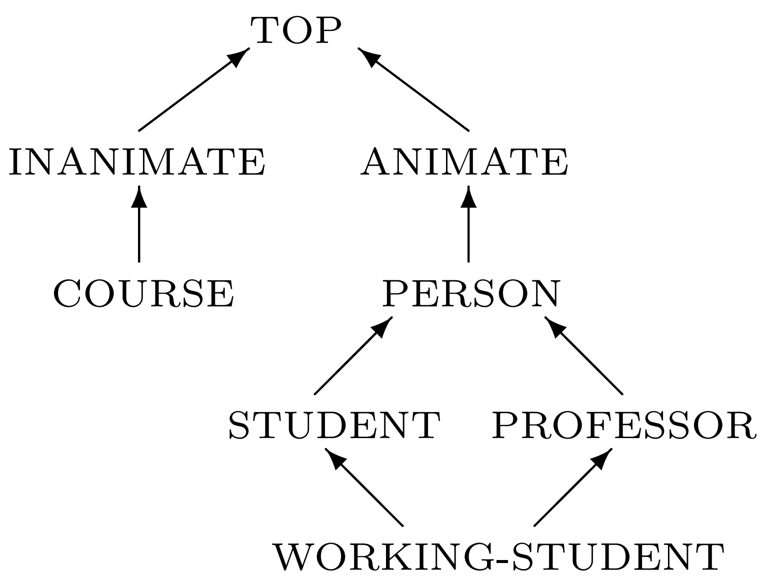
Taxonomies


TOP
INANIMATE
ANIMATE
COURSE
PERSON
STUDENT
PROFESSOR
WORKING-STUDENT

Taxonomies (cont'd)

- Subsumption is a **partial ordering** relation.
- If we consider only **named** concepts, subsumption induces a **taxonomy** where only direct subsumptions are explicitly drawn.
- A **taxonomy** is the minimal relation in the space of named concepts such that its reflexive and transitive closure is the subsumption relation.

Taxonomies (cont'd)



What is the place of the following concept in the above taxonomy?

$$\texttt{N} \equiv \texttt{ANIMATE} \sqcap (\texttt{STUDENT} \sqcup \texttt{PROFESSOR})$$

Classification

- The problem of **classification:** Given a concept C and a TBox $\mathcal{T}$, for all concepts D of $\mathcal{T}$ determine whether D subsumes C, or D is subsumed by C.
- Intuitively, this amounts to finding the "right place" for C in the taxonomy implicitly present in $\mathcal{T}$.
- **Classification** is the task of inserting new concepts in a taxonomy. It is **sorting** in partial orders.
- What is the solution to the classification problem posed in the previous slide?

Reasoning Procedures

- Terminating, complete and efficient algorithms for deciding **satisfiability** – and all the other reasoning services – are available for $\mathcal{ALC}$.
- Algorithms are based on **tableaux-calculi** techniques.
- Completeness is important for the usability of description logics in real applications.
- Such algorithms have been shown to be **efficient** for real knowledge bases, even if the problem in the corresponding logic is in PSPACE or EXPTIME.
- We will talk about tableaux-calculi for DLs in the next lecture.

Some Extensions of $\mathcal{ALC}$

Constructor	**Syntax**	**Semantics**
concept name	A	$A^{\mathcal{I}} \subseteq \Delta$
top concept	$\top$	Δ
bottom concept	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction ($\mathcal{U}$)	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
negation ($\mathcal{C}$)	$\neg C$	$\Delta \setminus C^{\mathcal{I}}$
universal restriction	$\forall R.C$	$\{x \mid (\forall y)(R^{\mathcal{I}}(x,y) \rightarrow C^{\mathcal{I}}(y))\}$
existential restriction ($\mathcal{E}$)	$\exists R.C$	$\{x \mid (\exists y)(R^{\mathcal{I}}(x,y) \wedge C^{\mathcal{I}}(y))\}$
cardinality restrictions ($\mathcal{N}$)	$\geqslant n\ R$	$\{x \mid \sharp\{y \mid R^{\mathcal{I}}(x,y)\} \geq n\}$
	$\leqslant n\ R$	$\{x \mid \sharp\{y \mid R^{\mathcal{I}}(x,y)\} \leq n\}$
enumeration of individuals ($\mathcal{O}$)	$\{a_1, \ldots, a_n\}$	$\{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$

Cardinality Restrictions

Role quantification **cannot** express that a woman has **at least 3** (or **at most 5**) children.

Cardinality restrictions can express conditions on the number of fillers.

Examples:

- $\texttt{BusyWoman} \equiv \texttt{Woman} \sqcap (\geqslant 3\ \texttt{child})$
- $\texttt{ConsciousWoman} \equiv \texttt{Woman} \sqcap (\leqslant 5\ \texttt{child})$

Observation: $(\geqslant 1\ R) \equiv \exists R$

Cardinality Restrictions (cont'd)

Example:

$$\texttt{BusyWoman} \equiv \texttt{Woman} \sqcap (\geqslant 3 \texttt{ child})$$

$$\texttt{ConsciousWoman} \equiv \texttt{Woman} \sqcap (\leqslant 5 \texttt{ child})$$

$$\texttt{BusyWoman}(\texttt{MARY})$$

$$\texttt{child}(\texttt{MARY}, \texttt{JOHN}),\ \ \texttt{child}(\texttt{MARY}, \texttt{JACK}),\ \ \texttt{child}(\texttt{MARY}, \texttt{KARL})$$

Question: $\Sigma \models \texttt{ConsciousWoman}(\texttt{MARY})$?

Example

Let Σ be the following:

$$\texttt{Family}(\texttt{F})$$

$$\texttt{Father}(\texttt{F}, \texttt{JOHN}),\ \texttt{Mother}(\texttt{F}, \texttt{SUE})$$

$$\texttt{Son}(\texttt{F}, \texttt{PAUL}),\ \texttt{Son}(\texttt{F}, \texttt{GEORGE}),\ \texttt{Son}(\texttt{F}, \texttt{ALEX})$$

Question: How many children does family `F` have?

OWA vs. CWA

Contrary to databases, DLs make the Open World Assumption. Absence of information is **not** interpreted as presence of negative information but simply as **lack of knowledge.**

Examples:

- $\Sigma \models (\geqslant 3\ \texttt{Son})(\texttt{F})$ **Yes**
- $\Sigma \models (\leqslant 1\ \texttt{Son})(\texttt{F})$ **No**
- $\Sigma \models (\geqslant 5\ \texttt{Son})(\texttt{F})$ **Unknown**

Enumeration Construct (One-of)

Examples:

$$\texttt{Weekday} \equiv \{\ \texttt{MON}, \texttt{TUE}, \texttt{WED}, \texttt{THU}, \texttt{FRI}, \texttt{SAT}, \texttt{SUN}\ \}$$

$$\texttt{Citizen} \equiv \texttt{Person} \sqcap \forall \texttt{lives}.\texttt{Country}$$

$$\texttt{French} \equiv \texttt{Citizen} \sqcap \forall \texttt{lives}.\{\texttt{FRANCE}\}$$

A Naming Scheme for DLs

Historically, in the family of languages we presented, the first language was $\mathcal{AL}$ (attributive concept description language). Extensions of $\mathcal{AL}$ have been studied and have been identified by strings of the form:

$$\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{C}]$$

The name $\mathcal{ALC}$ originally comes from "attributive concept description language with complements".

Because combinations of constructs can **simulate** others there can be different names for languages that are essentially the same, i.e., have the same expressive power.

Example: $\mathcal{ALC}$ has same expressivity as $\mathcal{ALCUE}$. Why?

Some Constructors for Role Expressions

Constructor	Syntax	Semantics
role name	P	$P^{\mathcal{I}} \subseteq \Delta \times \Delta$
conjunction	$R \sqcap S$	$R^{\mathcal{I}} \cap S^{\mathcal{I}}$
disjunction	$R \sqcup S$	$R^{\mathcal{I}} \cup S^{\mathcal{I}}$
negation	$\neg R$	$\Delta \times \Delta \setminus R^{\mathcal{I}}$
inverse	R^{-}	$\{(x, y) \in \Delta \times \Delta \mid (y, x) \in R^{\mathcal{I}}\}$
composition	$R \circ S$	$\{(x, y) \in \Delta \times \Delta \mid \exists z.\ (x, z) \in R^{\mathcal{I}} \wedge (z, y) \in S^{\mathcal{I}}\}$
range	$R\|_C$	$\{(x, y) \in \Delta \times \Delta \mid (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
product	$C \times D$	$\{(x, y) \in C^{\mathcal{I}} \times D^{\mathcal{I}}\}$

Extending Description Logics

The literature contains various nice extensions of the DLs we studied:

- Defaults and Beliefs
- Probability- and similarity-based reasoning
- Epistemic statements
- Closed world assumption
- Plural entities: records, sets, collections, aggregations
- Concrete domains
- Ontological primitives
  - time and action
  - space
  - parts and wholes

Implemented DL Systems

- The beginning of it all: Kl-One (1977)
- Krypton (1983), Nikl (1983), Kandor (1984), Penni, Kl-Two (1985)
- Second generation DL systems: Loom (1987), Classic (1989)
- Back (1990), Flex (1995), Kris (1991), Crack (1995)
- Optimization techniques take charge: FaCT (1997), DLP (1998), Racer (1999)
- DL reasoners for the ontologies and Semantic Web era: FaCT++, KAON2, Pellet, RacerPro

See `http://www.cs.man.ac.uk/~sattler/reasoners.html` for pointers to Web pages of DL reasoners.

Applications

- Conceptual Modelling
- Data Integration
- Configuration
- Software Engineering
- Medical Informatics
- Bioinformatics
- Natural Language Processing
- Knowledge Representation and Reasoning in the Semantic Web (remaining of this course!)
- ...

Readings

- Chapter 1 (An Introduction to DLs) and Chapter 2 (Reasoning in DLs) and Chapter 10 (Conceptual Modelling with DLs) of the DL Handbook:

  F. Baader, D. Calvanese, D. McGuinness, D. Nardi and P. F. Patel-Schneider (editors). The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2002.

  Available from:
  `http://www.inf.unibz.it/~franconi/dl/course/dlhb/home.html`

  Chapters 1 and 2 are good introductions to DLs.

  Chapter 10 is useful for ontology development using DLs.

Readings (cont'd)

- Franz Baader, Ian Horrocks, and Ulrike Sattler. Description Logics. In Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, editors, Handbook of Knowledge Representation. Elsevier, 2007.

  Available from `http://www.comlab.ox.ac.uk/people/ian.horrocks/Publications/complete.html#2007`

  This is a very recent comprehensive survey of the area of DLs.

- A. Borgida. Description Logics in Data Management. IEEE Transactions on Knowledge and Data Engineering 7(5), pages 671-682, 1995.

  Available from:
  `ftp://ftp.cs.rutgers.edu/pub/borgida/dldb.survey.TKDE.pdf`

  This paper is useful if you want to understand some of the connections of DLs to databases.

Readings (cont'd)

- R. J. Brachman and D. L. McGuinness and P. Patel-Schneider and L. A. Resnick and A. Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In *Principles of Semantic Networks*. John Sowa (editor), Morgan Kaufmann, 1991, pages 401-456.

  Available from:

  `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.9028`

  This paper contains a lot of nice examples so it is great for explaining where to use description logics. The syntax used is that of the DL-based language CLASSIC, but this should not be a problem in appreciating the examples and discussion in the paper.

Acknowledgements

These slides have been prepared by modifying slides by Enrico Franconi, University of Bolzano-Bozen, Italy.

See `http://www.inf.unibz.it/~franconi/dl/course/` for Enrico's course on DLs.

Some other courses on DLs are listed on `http://dl.kr.org/courses.html`.