Outline

1. **FOL Recap**
   - Syntax
   - Semantics
   - First Order Structures

2. **Description logics**
   - Introduction
   - Basic DL: $\mathcal{ALC}$
   - Reasoning services
Outline

1 FOL Recap
   - Syntax
   - Semantics
   - First Order Structures

2 Description logics
   - Introduction
   - Basic DL: ALC
   - Reasoning services
From data to ORM2 or text and then to FOL—or v.v.
Student *is an entity type.*
DegreeProgramme *is an entity type.*
Student attends DegreeProgramme.

**Each** Student attends **exactly one** DegreeProgramme.
**It is possible that more than one** Student attends the same DegreeProgramme.

*OR, in the negative:*
For each Student, it is impossible that that Student attends more than one DegreeProgramme.
**It is impossible that any** Student attends no DegreeProgramme.

<table>
<thead>
<tr>
<th>Student</th>
<th>DegreeProgramme</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Computer Science</td>
</tr>
<tr>
<td>Mary</td>
<td>Design</td>
</tr>
<tr>
<td>Fabio</td>
<td>Design</td>
</tr>
<tr>
<td>Claudio</td>
<td>Computer Science</td>
</tr>
<tr>
<td>Markus</td>
<td>Biology</td>
</tr>
<tr>
<td>Inge</td>
<td>Computer Science</td>
</tr>
</tbody>
</table>
How to formalise it?

- Note: logic is not the study of truth, but of the relationship between the truth of one statement and that of another

- Syntax
  - Alphabet
  - Languages constructs
  - Sentences to assert knowledge

- Semantics
  - Formal meaning
How to formalise it?

- Note: logic is not the study of truth, but of the relationship between the truth of one statement and that of another.

- Syntax
  - Alphabet
  - Languages constructs
  - Sentences to assert knowledge

- Semantics
  - Formal meaning
The lexicon of a first order language contains:

- **Connectives & Parentheses**: ¬, →, ↔, ∧, ∨, ( and );
- **Quantifiers**: ∀ (universal) and ∃ (existential);
- **Variables**: x, y, z, ... ranging over particulars;
- **Constants**: a, b, c, ... representing a specific element;
- **Functions**: f, g, h, ..., with arguments listed as f(x₁,...xₙ);
- **Relations**: R, S, ... with an associated arity.
Example: From Natural Language to First order logic (or \textit{vv.})

- Each animal is an organism
  - All animals are organisms
  - If it is an animal then it is an organism
  \[ \forall x (\text{Animal}(x) \rightarrow \text{Organism}(x)) \]

- Each student must be registered for a degree programme
  \[ \forall x (\text{registered for}(x, y) \rightarrow \text{Student}(x) \land \text{DegreeProgramme}(y)) \]
  \[ \forall x (\text{Student}(x) \rightarrow \exists y \text{ registered for}(x, y)) \]

- Aliens exist
  \[ \exists x \text{ Alien}(x) \]

- There are books that are heavy
  \[ \exists x (\text{Book}(x) \land \text{heavy}(x)) \]
Example: From Natural Language to First order logic (or vv.)

- Each animal is an organism
  All animals are organisms
  If it is an animal then it is an organism
  $\forall x (\text{Animal}(x) \rightarrow \text{Organism}(x))$

- Each student must be registered for a degree programme
  $\forall x (\text{registered for}(x, y) \rightarrow \text{Student}(x) \land \text{DegreeProgramme}(y))$
  $\forall x (\text{Student}(x) \rightarrow \exists y \text{ registered for}(x, y))$

- Aliens exist
  $\exists x \text{ Alien}(x)$

- There are books that are heavy
  $\exists x (\text{Book}(x) \land \text{heavy}(x))$
Example: From Natural Language to First order logic (or vv.)

- Each animal is an organism
  All animals are organisms
  If it is an animal then it is an organism
  \( \forall x (Animal(x) \rightarrow Organism(x)) \)

- Each student must be registered for a degree programme
  \( \forall x (\text{registered for}(x, y) \rightarrow Student(x) \wedge DegreeProgramme(y)) \)
  \( \forall x (Student(x) \rightarrow \exists y \text{ registered for}(x, y)) \)

- Aliens exist
  \( \exists x \text{ Alien}(x) \)

- There are books that are heavy
  \( \exists x (\text{Book}(x) \wedge \text{heavy}(x)) \)
Example: From Natural Language to First order logic (or vv.)

- Each animal is an organism
  All animals are organisms
  If it is an animal then it is an organism
  $$\forall x (\text{Animal}(x) \rightarrow \text{Organism}(x))$$

- Each student must be registered for a degree programme
  $$\forall x (\text{registered_for}(x, y) \rightarrow \text{Student}(x) \land \text{DegreeProgramme}(y))$$
  $$\forall x (\text{Student}(x) \rightarrow \exists y \text{ registered_for}(x, y))$$

- Aliens exist
  $$\exists x \text{ Alien}(x)$$

- There are books that are heavy
  $$\exists x (\text{Book}(x) \land \text{heavy}(x))$$
Example: From Natural Language to First order logic (or \textit{vv.})

- Each animal is an organism
  All animals are organisms
  If it is an animal then it is an organism
  \[ \forall x (\text{Animal}(x) \rightarrow \text{Organism}(x)) \]

- Each student must be registered for a degree programme
  \[ \forall x (\text{registered} \_\text{for}(x, y) \rightarrow \text{Student}(x) \land \text{DegreeProgramme}(y)) \]
  \[ \forall x (\text{Student}(x) \rightarrow \exists y \text{ registered} \_\text{for}(x, y)) \]

- Aliens exist
  \[ \exists x \text{ Alien}(x) \]

- There are books that are heavy
  \[ \exists x (\text{Book}(x) \land \text{heavy}(x)) \]
Example: From Natural Language to First order logic (or vv.)

- Each animal is an organism
  All animals are organisms
  If it is an animal then it is an organism
  $\forall x (\text{Animal}(x) \rightarrow \text{Organism}(x))$

- Each student must be registered for a degree programme
  $\forall x (\text{registered\_for}(x, y) \rightarrow \text{Student}(x) \land \text{DegreeProgramme}(y))$
  $\forall x (\text{Student}(x) \rightarrow \exists y \text{ registered\_for}(x, y))$

- Aliens exist
  $\exists x \text{ Alien}(x)$

- There are books that are heavy
  $\exists x (\text{Book}(x) \land \text{heavy}(x))$
Example: From Natural Language to First order logic (or \textit{vv.})

- Each animal is an organism
  All animals are organisms
  If it is an animal then it is an organism
  \(\forall x(\text{Animal}(x) \rightarrow \text{Organism}(x))\)

- Each student must be registered for a degree programme
  \(\forall x(\text{registered\_for}(x, y) \rightarrow \text{Student}(x) \land \text{DegreeProgramme}(y))\)
  \(\forall x(\text{Student}(x) \rightarrow \exists y \text{ registered\_for}(x, y))\)

- Aliens exist
  \(\exists x \text{ Alien}(x)\)

- There are books that are heavy
  \(\exists x (\text{Book}(x) \land \text{heavy}(x))\)
Example: From Natural Language to First order logic (or \textit{vv.})

- Each animal is an organism
  All animals are organisms
  If it is an animal then it is an organism
  \[ \forall x (\text{Animal}(x) \rightarrow \text{Organism}(x)) \]

- Each student must be registered for a degree programme
  \[ \forall x (\text{registered\_for}(x, y) \rightarrow \text{Student}(x) \land \text{DegreeProgramme}(y)) \]
  \[ \forall x (\text{Student}(x) \rightarrow \exists y \text{ registered\_for}(x, y)) \]

- Aliens exist
  \[ \exists x \text{ Alien}(x) \]

- There are books that are heavy
  \[ \exists x (\text{Book}(x) \land \text{heavy}(x)) \]
(countably infinite) Supply of symbols (signature): Variables, Functions, Constants, and Relations

Terms: A term is inductively defined by two rules, being:
1. Every variable and constant is a term.
2. If \( f \) is a \( m \)-ary function and \( t_1, \ldots, t_m \) are terms, then \( f(t_1, \ldots, t_m) \) is also a term.

Definition (atomic formula)

An atomic formula is a formula that has the form \( t_1 = t_2 \) or \( R(t_1, \ldots, t_n) \) where \( R \) is an \( n \)-ary relation and \( t_1, \ldots, t_n \) are terms.

R1. If \( \phi \) is a formula then so is \( \neg \phi \).
R2. If \( \phi \) and \( \psi \) are formulas then so is \( \phi \land \psi \).
R3. If \( \phi \) is a formula then so is \( \exists x \phi \) for any variable \( x \).
First order logic

- (countably infinite) Supply of **symbols** (signature): Variables, Functions, Constants, and Relations
- **Terms**: A term is inductively defined by two rules, being:
  1. Every variable and constant is a term.
  2. If \( f \) is a \( m \)-ary function and \( t_1, \ldots, t_m \) are terms, then \( f(t_1, \ldots, t_m) \) is also a term.

**Definition (atomic formula)**

An *atomic formula* is a formula that has the form \( t_1 = t_2 \) or \( R(t_1, \ldots, t_n) \) where \( R \) is an \( n \)-ary relation and \( t_1, \ldots, t_n \) are terms.

R1. If \( \phi \) is a formula then so is \( \neg \phi \).
R2. If \( \phi \) and \( \psi \) are formulas then so is \( \phi \land \psi \).
R3. If \( \phi \) is a formula then so is \( \exists x \phi \) for any variable \( x \).
First order logic

- (countably infinite) Supply of symbols (signature): Variables, Functions, Constants, and Relations
- **Terms**: A term is inductively defined by two rules, being:
  1. Every variable and constant is a term.
  2. If $f$ is a $m$-ary function and $t_1, \ldots, t_m$ are terms, then $f(t_1, \ldots, t_m)$ is also a term.

**Definition (atomic formula)**

An atomic formula is a formula that has the form $t_1 = t_2$ or $R(t_1, \ldots, t_n)$ where $R$ is an $n$-ary relation and $t_1, \ldots, t_n$ are terms.

R1. If $\phi$ is a formula then so is $\neg \phi$.
R2. If $\phi$ and $\psi$ are formulas then so is $\phi \land \psi$.
R3. If $\phi$ is a formula then so is $\exists x \phi$ for any variable $x$. 
A string of symbols is a \textit{formula} of FOL if and only if it is constructed from atomic formulas by repeated applications of rules R1, R2, and R3.

A \textit{free variable} of a formula $\phi$ is that variable occurring in $\phi$ that is not quantified. We then can introduce the definition of \textit{sentence}.

A \textit{sentence} of FOL is a formula having no free variables.
### Definition (formula)

A string of symbols is a *formula* of FOL if and only if it is constructed from atomic formulas by repeated applications of rules R1, R2, and R3.

A *free variable* of a formula $\phi$ is that variable occurring in $\phi$ that is not quantified. We then can introduce the definition of *sentence*.

### Definition (sentence)

A *sentence* of FOL is a formula having no free variables.
Whether a sentence is true or not depends on the underlying set and the interpretation of the function, constant, and relation symbols.

A *structure* consists of an *underlying set* together with an *interpretation* of functions, constants, and relations.

Given a sentence $\phi$ and a structure $M$, $M$ *models* $\phi$ means that the sentence $\phi$ is true with respect to $M$. More precisely,

**Definition (vocabulary)**

A *vocabulary* $\mathcal{V}$ is a set of function, relation, and constant symbols.
FOL Recap

Description logics

Summary

Semantics

FOL Cont.: toward semantics

- Whether a sentence is true or not depends on the underlying set and the interpretation of the function, constant, and relation symbols.
- A structure consists of an underlying set together with an interpretation of functions, constants, and relations.
- Given a sentence $\phi$ and a structure $M$, $M$ models $\phi$ means that the sentence $\phi$ is true with respect to $M$. More precisely,

**Definition (vocabulary)**

A *vocabulary* $\mathcal{V}$ is a set of function, relation, and constant symbols.
A \(\mathcal{V}\)-structure consists of a non-empty underlying set \(\Delta\) along with an interpretation of \(\mathcal{V}\). An interpretation of \(\mathcal{V}\) assigns an element of \(\Delta\) to each constant in \(\mathcal{V}\), a function from \(\Delta^n\) to \(\Delta\) to each \(n\)-ary function in \(\mathcal{V}\), and a subset of \(\Delta^n\) to each \(n\)-ary relation in \(\mathcal{V}\). We say \(M\) is a \textit{structure} if it is a \(\mathcal{V}\)-structure of some vocabulary \(\mathcal{V}\).

Let \(\mathcal{V}\) be a vocabulary. A \(\mathcal{V}\)-formula is a formula in which every function, relation, and constant is in \(\mathcal{V}\). A \(\mathcal{V}\)-sentence is a \(\mathcal{V}\)-formula that is a sentence.
Definition (\(V\)-structure)

A \(V\)-structure consists of a non-empty underlying set \(\Delta\) along with an interpretation of \(V\). An interpretation of \(V\) assigns an element of \(\Delta\) to each constant in \(V\), a function from \(\Delta^n\) to \(\Delta\) to each \(n\)-ary function in \(V\), and a subset of \(\Delta^n\) to each \(n\)-ary relation in \(V\). We say \(M\) is a structure if it is a \(V\)-structure of some vocabulary \(V\).

Definition (\(V\)-formula)

Let \(V\) be a vocabulary. A \(V\)-formula is a formula in which every function, relation, and constant is in \(V\). A \(V\)-sentence is a \(V\)-formula that is a sentence.
When we say that $M$ models $\phi$, denoted with $M \models \phi$, this is with respect to $M$ being a $\mathcal{V}$-structure and $\mathcal{V}$-sentence $\phi$ is true in $M$.

Model theory: the interplay between $M$ and a set of first-order sentences $\mathcal{T}(M)$, which is called the theory of $M$, and its ‘inverse’ from a set of sentences $\Gamma$ to a class of structures.

**Definition (theory of $M$)**

For any $\mathcal{V}$-structure $M$, the theory of $M$, denoted with $\mathcal{T}(M)$, is the set of all $\mathcal{V}$-sentences $\phi$ such that $M \models \phi$.

**Definition (model)**

For any set of $\mathcal{V}$-sentences, a model of $\Gamma$ is a $\mathcal{V}$-structure that models each sentence in $\Gamma$. The class of all models of $\Gamma$ is denoted by $\mathcal{M}(\Gamma)$. 
When we say that $M$ models $\phi$, denoted with $M \models \phi$, this is with respect to $M$ being a $\mathcal{V}$-structure and $\mathcal{V}$-sentence $\phi$ is true in $M$.

Model theory: the interplay between $M$ and a set of first-order sentences $\mathcal{T}(M)$, which is called the theory of $M$, and its ‘inverse’ from a set of sentences $\Gamma$ to a class of structures.

**Definition (theory of $M$)**

For any $\mathcal{V}$-structure $M$, the theory of $M$, denoted with $\mathcal{T}(M)$, is the set of all $\mathcal{V}$-sentences $\phi$ such that $M \models \phi$.

**Definition (model)**

For any set of $\mathcal{V}$-sentences, a model of $\Gamma$ is a $\mathcal{V}$-structure that models each sentence in $\Gamma$. The class of all models of $\Gamma$ is denoted by $\mathcal{M}(\Gamma)$.
When we say that $M$ models $\phi$, denoted with $M \models \phi$, this is with respect to $M$ being a $\mathcal{V}$-structure and $\mathcal{V}$-sentence $\phi$ is true in $M$.

Model theory: the interplay between $M$ and a set of first-order sentences $\mathcal{T}(M)$, which is called the theory of $M$, and its ‘inverse’ from a set of sentences $\Gamma$ to a class of structures.

**Definition (theory of $M$)**

For any $\mathcal{V}$-structure $M$, the theory of $M$, denoted with $\mathcal{T}(M)$, is the set of all $\mathcal{V}$-sentences $\phi$ such that $M \models \phi$.

**Definition (model)**

For any set of $\mathcal{V}$-sentences, a model of $\Gamma$ is a $\mathcal{V}$-structure that models each sentence in $\Gamma$. The class of all models of $\Gamma$ is denoted by $\mathcal{M}(\Gamma)$.
Theory in the context of logic

Definition (complete \( \mathcal{V} \)-theory)

Let \( \Gamma \) be a set of \( \mathcal{V} \)-sentences. Then \( \Gamma \) is a complete \( \mathcal{V} \)-theory if, for any \( \mathcal{V} \)-sentence \( \phi \) either \( \phi \) or \( \neg \phi \) is in \( \Gamma \) and it is not the case that both \( \phi \) and \( \neg \phi \) are in \( \Gamma \).

- It can then be shown that for any \( \mathcal{V} \)-structure \( M \), \( T(M) \) is a complete \( \mathcal{V} \)-theory (for proof, see e.g. [Hedman04, p90])

Definition

A set of sentences \( \Gamma \) is said to be consistent if no contradiction can be derived from \( \Gamma \).

Definition (theory)

A theory is a consistent set of sentences.
**Theory in the context of logic**

**Definition (complete $\mathcal{V}$-theory)**

Let $\Gamma$ be a set of $\mathcal{V}$-sentences. Then $\Gamma$ is a *complete $\mathcal{V}$-theory* if, for any $\mathcal{V}$-sentence $\phi$ either $\phi$ or $\neg \phi$ is in $\Gamma$ and it is not the case that both $\phi$ and $\neg \phi$ are in $\Gamma$.

- It can then be shown that for any $\mathcal{V}$-structure $M$, $T(M)$ is a complete $\mathcal{V}$-theory (for proof, see e.g. [Hedman04, p90])

**Definition**

A set of sentences $\Gamma$ is said to be *consistent* if no contradiction can be derived from $\Gamma$.

**Definition (theory)**

A *theory* is a consistent set of sentences.
**Theory in the context of logic**

**Definition (complete $\mathcal{V}$-theory)**

Let $\Gamma$ be a set of $\mathcal{V}$-sentences. Then $\Gamma$ is a complete $\mathcal{V}$-theory if, for any $\mathcal{V}$-sentence $\phi$ either $\phi$ or $\neg\phi$ is in $\Gamma$ and it is not the case that both $\phi$ and $\neg\phi$ are in $\Gamma$.

- It can then be shown that for any $\mathcal{V}$-structure $M$, $T(M)$ is a complete $\mathcal{V}$-theory (for proof, see e.g. [Hedman04, p90])

**Definition**

A set of sentences $\Gamma$ is said to be consistent if no contradiction can be derived from $\Gamma$.

**Definition (theory)**

A theory is a consistent set of sentences.
Some definitions

- A formula is **valid** if it holds under *every* assignment. $\models \phi$ to denote this. A valid formula is called a **tautology**.
- A formula is **satisfiable** if it holds under *some* assignment.
- A formula is **unsatisfiable** if it holds under *no* assignment. An unsatisfiable formula is called a **contradiction**.
Example

- Is this a theory?
  \[ \forall x (\text{Woman}(x) \rightarrow \text{Female}(x)) \]
  \[ \forall x (\text{Mother}(x) \rightarrow \text{Woman}(x)) \]
  \[ \forall x (\text{Man}(x) \leftrightarrow \neg \text{Woman}(x)) \]
  \[ \forall x (\text{Mother}(x) \rightarrow \exists y (\text{partnerOf}(x, y) \land \text{Spouse}(y))) \]
  \[ \forall x (\text{Spouse}(x) \rightarrow \text{Man}(x) \lor \text{Woman}(x)) \]
  \[ \forall x, y (\text{Mother}(x) \land \text{partnerOf}(x, y) \rightarrow \text{Father}(y)) \]

- Is it still a theory if we add:
  \[ \forall x (\text{Hermaphrodite}(x) \rightarrow \text{Man}(x) \land \text{Woman}(x)) \]
Example

- Is this a theory?
  \[ \forall x (\text{Woman}(x) \rightarrow \text{Female}(x)) \]
  \[ \forall x (\text{Mother}(x) \rightarrow \text{Woman}(x)) \]
  \[ \forall x (\text{Man}(x) \leftrightarrow \neg \text{Woman}(x)) \]
  \[ \forall x (\text{Mother}(x) \rightarrow \exists y (\text{partnerOf}(x, y) \land \text{Spouse}(y))) \]
  \[ \forall x (\text{Spouse}(x) \rightarrow \text{Man}(x) \lor \text{Woman}(x)) \]
  \[ \forall x, y (\text{Mother}(x) \land \text{partnerOf}(x, y) \rightarrow \text{Father}(y)) \]

- Is it still a theory if we add:
  \[ \forall x (\text{Hermaphrodite}(x) \rightarrow \text{Man}(x) \land \text{Woman}(x)) \]
Examples of first-order structures (exercise)

- Graphs are mathematical structures.
- A graph is a set of points, called vertices, and lines, called edges between them. For instance:

![Graphs A, B, and C](image)

- Figures A and B are different depictions, but have the same descriptions w.r.t. the vertices and edges. Check this.
- Graph C has a property that A and B do not have. Represent this in a first-order sentence.
- Find a suitable first-order language for A (/B), and formulate at least two properties of the graph using quantifiers.
Graphs are mathematical structures.

A graph is a set of points, called **vertices**, and lines, called **edges** between them. For instance:

- Figures A and B are different depictions, but have the same descriptions w.r.t. the vertices and edges. Check this.
- Graph C has a property that A and B do not have. Represent this in a first-order sentence.
- Find a suitable first-order language for A (/B), and formulate at least two properties of the graph using quantifiers.
Graphs are mathematical structures.

A graph is a set of points, called vertices, and lines, called edges between them. For instance:

Figures A and B are different depictions, but have the same descriptions w.r.t. the vertices and edges. Check this.

Graph C has a property that A and B do not have. Represent this in a first-order sentence.

Find a suitable first-order language for A (/B), and formulate at least two properties of the graph using quantifiers.
More graphs (exercise)

Consider the following graph, and first-order language $\mathcal{L} = \langle R \rangle$, with $R$ being a binary relation symbol (edge).

1. Formalise the following properties of the graph as $\mathcal{L}$-sentences: (i) $(a, a)$ and $(b, b)$ are edges of the graph; (ii) $(a, b)$ is an edge of the graph; (iii) $(b, a)$ is not an edge of the graph. Let $T$ stand for the resulting set of sentences.

2. Prove that $T \cup \{\forall x \forall y R(x, y)\}$ is unsatisfiable using tableaux calculus.
More graphs (exercise)

Consider the following graph, and first-order language $\mathcal{L} = \langle R \rangle$, with $R$ being a binary relation symbol (edge).

1. Formalise the following properties of the graph as $\mathcal{L}$-sentences: (i) $(a, a)$ and $(b, b)$ are edges of the graph; (ii) $(a, b)$ is an edge of the graph; (iii) $(b, a)$ is not an edge of the graph. Let $T$ stand for the resulting set of sentences.

2. Prove that $T \cup \{\forall x \forall y R(x, y)\}$ is unsatisfiable using tableaux calculus.
Representing the knowledge in a suitable logic is one thing, reasoning over it another. e.g.:

- How do we find out whether a formula is valid or not?
- How do we find out whether our knowledge base is satisfiable?

Among others:

- Truth tables
- Tableaux (principal approach for DL reasoners)
- ...
Representing the knowledge in a suitable logic is one thing, reasoning over it another. e.g.:

- How do we find out whether a formula is valid or not?
- How do we find out whether our knowledge base is satisfiable?

Among others:

- Truth tables
- Tableaux (principal approach for DL reasoners)
- ...

Representing the knowledge in a suitable logic is one thing, reasoning over it another. e.g.:
- How do we find out whether a formula is valid or not?
- How do we find out whether our knowledge base is satisfiable?

Among others:
- Truth tables
- Tableaux (principal approach for DL reasoners)
- ...
A sound and complete procedure deciding satisfiability is all we need, and the **tableaux method is a decision procedure which checks the existence of a model**

It exhaustively looks at all the possibilities, so that it can eventually prove that no model could be found for unsatisfiable formulas.

- $\phi \models \psi$ iff $\phi \land \neg \psi$ is NOT satisfiable—if it is satisfiable, we have found a counterexample
- Decompose the formula in top-down fashion
A sound and complete procedure deciding satisfiability is all we need, and the **tableaux method is a decision procedure which checks the existence of a model**.

It exhaustively looks at all the possibilities, so that it can eventually prove that no model could be found for unsatisfiable formulas.

\[ \phi \models \psi \text{ iff } \phi \land \neg \psi \text{ is NOT satisfiable—if it is satisfiable, we have found a counterexample} \]

Decompose the formula in top-down fashion.
A sound and complete procedure deciding satisfiability is all we need, and the **tableaux method is a decision procedure which checks the existence of a model**

It exhaustively looks at all the possibilities, so that it can eventually prove that no model could be found for unsatisfiable formulas.

\[ \phi \models \psi \iff \phi \land \neg \psi \text{ is NOT satisfiable} \]

- if it is satisfiable, we have found a counterexample
- Decompose the formula in top-down fashion
Tableaux summary (2/2)

- Tableaux calculus works only if the formula has been translated into **Negation Normal Form**, i.e., all the negations have been pushed inside.
- Recollect the list of equivalences, apply those to arrive at NNF, if necessary.
- If a model satisfies a conjunction, then it also satisfies each of the conjuncts.
- If a model satisfies a disjunction, then it also satisfies one of the disjuncts. It is a non-deterministic rule, and it generates two alternative branches.
- Apply the completion rules until either (a) an explicit contradiction due to the presence of two opposite literals in a node (a clash) is generated in each branch, or (b) there is a completed branch where no more rule is applicable.
Tableaux summary (2/2)

- Tableaux calculus works only if the formula has been translated into **Negation Normal Form**, i.e., all the negations have been pushed inside.
- Recollect the list of equivalences, apply those to arrive at NNF, if necessary.
- If a model satisfies a conjunction, then it also satisfies each of the conjuncts.
- If a model satisfies a disjunction, then it also satisfies one of the disjuncts. It is a non-deterministic rule, and it generates two alternative branches.
- Apply the completion rules until either (a) an explicit contradiction due to the presence of two opposite literals in a node (a clash) is generated in each branch, or (b) there is a completed branch where no more rule is applicable.
Tableaux summary (2/2)

- Tableaux calculus works only if the formula has been translated into **Negation Normal Form**, i.e., all the negations have been pushed inside.
- Recollect the list of equivalences, apply those to arrive at NNF, if necessary.
- If a model satisfies a conjunction, then it also satisfies each of the conjuncts.
- If a model satisfies a disjunction, then it also satisfies one of the disjuncts. It is a non-deterministic rule, and it generates two alternative branches.
- Apply the completion rules until either (a) an explicit contradiction due to the presence of two opposite literals in a node (a clash) is generated in each branch, or (b) there is a completed branch where no more rule is applicable.
Outline

1 FOL Recap
   • Syntax
   • Semantics
   • First Order Structures

2 Description logics
   • Introduction
   • Basic DL: $\mathcal{ALC}$
   • Reasoning services
What are DLs?

- A logical reconstruction and (claimed to be a) unifying formalism for other KR languages, such as frames-based systems, OO modelling, Semantic data models, etc.
- A structured fragment of FOL
- Representation is at the predicate level: no variables are present in the formalism
- Any (basic) Description Logic is a subset of $\mathcal{L}_3$, i.e., the function-free FOL using only at most three variable names.
- Provide theories and systems for declaratively expressing structured information and for accessing and reasoning with it.
- Used for, a.o.: terminologies and ontologies, formal conceptual data modelling, information integration, ....
What are DLs?

- A logical reconstruction and (claimed to be a) unifying formalism for other KR languages, such as frames-based systems, OO modelling, Semantic data models, etc.
- A structured fragment of FOL
- Representation is at the predicate level: no variables are present in the formalism
- Any (basic) Description Logic is a subset of $\mathcal{L}_3$, i.e., the function-free FOL using only at most three variable names.
- Provide theories and systems for declaratively expressing structured information and for accessing and reasoning with it.
- Used for, a.o.: terminologies and ontologies, formal conceptual data modelling, information integration, ....
Description Logic knowledge base

- Description Language
- TBox
- ABox
- Reasoning

KB

Application Programs

Rules
Basic DL: \textit{ALC}

\textbf{ALC}

- **Concepts** denoting entity types/classes/unary predicates/universals, including top \( \top \) and bottom \( \bot \);
- **Roles** denoting relationships/associations/n-ary predicates/properties;
- **Constructors**: and \( \sqcap \), or \( \sqcup \), and not \( \neg \); quantifications for all \( \forall \) and exists \( \exists \);
- **Complex concepts** using constructors
  - Let \( C \) and \( D \) be concept names, \( R \) a role name, then
  - \( \neg C \), \( C \sqcap D \), and \( C \sqcup D \) are concepts, and
  - \( \forall R.C \) and \( \exists R.C \) are concepts
- **Individuals**
**ALC Examples**

- Concepts (primitive, atomic): Book, Course
- Roles: ENROLLED, READS
- Complex concepts:
  - Student $\sqsubseteq \exists$ENROLLED.(Course $\sqcap$ DegreeProgramme) (a primitive concept)
  - Mother $\sqsubseteq$ Woman $\sqcap$ $\exists$PARENTOF.Person
  - Parent $\equiv$ (Male $\sqcap$ Female) $\sqcap$ $\exists$PARENTOF.Mammal $\sqcap$ $\exists$CARESFOR.Mammal (a defined concept)
- Individuals: Student(Shereen), Mother(Sally), $\neg$Student(Sally), ENROLLED(Shereen, COMP101)
- But what does it really mean?
Basic DL: $\mathcal{ALC}$

$\mathcal{ALC}$ Examples

- Concepts (primitive, atomic): Book, Course
- Roles: ENROLLED, READS
- Complex concepts:
  - Student $\sqsubseteq \exists$ENROLLED.(Course $\sqcup$ DegreeProgramme) (a primitive concept)
  - Mother $\sqsubseteq$ Woman $\sqcap$ $\exists$PARENTOF.Person
  - Parent $\equiv$ (Male $\sqcup$ Female) $\sqcap$ $\exists$PARENTOF.Mammal $\sqcap$ $\exists$CARESFOR.Mammal (a defined concept)
- Individuals: Student(Shereen), Mother(Sally), $\neg$Student(Sally), ENROLLED(Shereen, COMP101)
- But what does it really mean?
Semantics of $\mathcal{ALC}$ (1/3)

- Domain $\Delta$ is a non-empty set of objects.
- Interpretation: $\mathcal{I}$ is the interpretation function, domain $\Delta^\mathcal{I}$
  - $\mathcal{I}$ maps every concept name $A$ to a subset $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$
  - $\mathcal{I}$ maps every role name $R$ to a subset $R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$
  - $\mathcal{I}$ maps every individual name $a$ to elements of $\Delta^\mathcal{I}$: $a^\mathcal{I} \in \Delta^\mathcal{I}$
- Note: $\top^\mathcal{I} = \Delta^\mathcal{I}$ and $\bot^\mathcal{I} = \emptyset$
C and D are concepts, R a role

\(-C^I = \Delta^I \setminus C^I\)

\((C \sqcap D)^I = C^I \cap D^I\)

\((C \sqcup D)^I = C^I \cup D^I\)

\((\forall R. C)^I = \{x \mid \forall y. R^I(x, y) \rightarrow C^I(y)\}\)

\((\exists R. C)^I = \{x \mid \exists y. R^I(x, y) \land C^I(y)\}\)
C and D are concepts, R a role, a and b are individuals

An interpretation $\mathcal{I}$ satisfies the statement $C \sqsubseteq D$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$

An interpretation $\mathcal{I}$ satisfies the statement $C \equiv D$ if $C^\mathcal{I} = D^\mathcal{I}$

$C(a)$ is satisfied by $\mathcal{I}$ if $a^\mathcal{I} \in C^\mathcal{I}$

$R(a, b)$ is satisfied by $\mathcal{I}$ if $(a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}$

An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ is a model of a knowledge base $\mathcal{KB}$ if every axiom of $\mathcal{KB}$ is satisfied by $\mathcal{I}$

A knowledge base $\mathcal{KB}$ is said to be satisfiable if it admits a model
C and D are concepts, R a role, a and b are individuals
An interpretation $\mathcal{I}$ satisfies the statement $C \sqsubseteq D$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$
An interpretation $\mathcal{I}$ satisfies the statement $C \equiv D$ if $C^\mathcal{I} = D^\mathcal{I}$
$C(a)$ is satisfied by $\mathcal{I}$ if $a^\mathcal{I} \in C^\mathcal{I}$
$R(a, b)$ is satisfied by $\mathcal{I}$ if $(a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}$

An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ is a model of a knowledge base $\mathcal{KB}$ if every axiom of $\mathcal{KB}$ is satisfied by $\mathcal{I}$
A knowledge base $\mathcal{KB}$ is said to be satisfiable if it admits a model
Semantics of $\mathcal{ALC}$ (3/3)

- C and D are concepts, R a role, a and b are individuals.
- An interpretation $\mathcal{I}$ satisfies the statement $C \sqsubseteq D$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$.
- An interpretation $\mathcal{I}$ satisfies the statement $C \equiv D$ if $C^\mathcal{I} = D^\mathcal{I}$.
- $C(a)$ is satisfied by $\mathcal{I}$ if $a^\mathcal{I} \in C^\mathcal{I}$.
- $R(a, b)$ is satisfied by $\mathcal{I}$ if $(a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}$.

An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ is a model of a knowledge base $\mathcal{KB}$ if every axiom of $\mathcal{KB}$ is satisfied by $\mathcal{I}$.

A knowledge base $\mathcal{KB}$ is said to be satisfiable if it admits a model.
Logical implication

- $KB \models \phi$ if every model of $KB$ is a model of $\phi$

- Example:
  
  TBox: $\exists$TEACHES.Couse $\sqsubseteq$ ¬Undergrad $\sqcup$ Professor
  
  ABox: TEACHES(John, cs101), Course(cs101), Undergrad(John)

- $KB \models$ Professor(John)

- What if:
  
  TBox: $\exists$TEACHES.Course $\sqsubseteq$ Undergrad $\sqcup$ Professor
  
  ABox: TEACHES(John, cs101), Course(cs101), Undergrad(John)

- $KB \models$ Professor(John)? or perhaps $KB \models$ ¬Professor(John)?
Logical implication

- $\mathcal{KB} \models \phi$ if every model of $\mathcal{KB}$ is a model of $\phi$

Example:
- TBox: $\exists \text{TEACHES}.\text{Course} \sqsubseteq \neg \text{Undergrad} \sqcup \text{Professor}$
- ABox: TEACHES(John, cs101), Course(cs101), Undergrad(John)

- $\mathcal{KB} \models \text{Professor}(\text{John})$

What if:
- TBox: $\exists \text{TEACHES}.\text{Course} \sqsubseteq \text{Undergrad} \sqcup \text{Professor}$
- ABox: TEACHES(John, cs101), Course(cs101), Undergrad(John)

- $\mathcal{KB} \models \text{Professor}(\text{John})$? or perhaps $\mathcal{KB} \models \neg \text{Professor}(\text{John})$?
Logical implication

- $\mathcal{KB} \models \phi$ if every model of $\mathcal{KB}$ is a model of $\phi$

Example:

- TBox: $\exists$TEACHES.Course $\sqsubseteq$ ¬Undergrad $\sqcap$ Professor
- ABox: TEACHES(John, cs101), Course(cs101), Undergrad(John)

- $\mathcal{KB} \models $ Professor(John)

- What if:
  - TBox: $\exists$TEACHES.Course $\sqsubseteq$ Undergrad $\sqcap$ Professor
  - ABox: TEACHES(John, cs101), Course(cs101), Undergrad(John)

- $\mathcal{KB} \models $ Professor(John) or perhaps $\mathcal{KB} \models $ ¬Professor(John)?
Logical implication

- $\mathcal{KB} \models \phi$ if every model of $\mathcal{KB}$ is a model of $\phi$

Example:
- TBox: $\exists$TEACHES.Course $\sqsubseteq \neg$Undergrad $\sqcup$ Professor
- ABox: TEACHES(John, cs101), Course(cs101), Undergrad(John)

- $\mathcal{KB} \models \text{Professor}(\text{John})$

What if:
- TBox: $\exists$TEACHES.Course $\sqsubseteq$ Undergrad $\sqcup$ Professor
- ABox: TEACHES(John, cs101), Course(cs101), Undergrad(John)

- $\mathcal{KB} \models \text{Professor}(\text{John})$? or perhaps $\mathcal{KB} \models \neg\text{Professor}(\text{John})$?
Reasoning services for DL-based OWL ontologies

- **Consistency of the knowledge base** ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)
  - Is the \( \mathcal{KB} = (T, A) \) consistent (non-selfcontradictory), i.e., is there at least a model for \( \mathcal{KB} \)?

- **Concept (and role) satisfiability** ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - Is there a model of \( \mathcal{KB} \) in which \( C \) (resp. \( R \)) has a nonempty extension?

- **Concept (and role) subsumption** ($\mathcal{KB} \models C \sqsubseteq D$)
  - I.e., is the extension of \( C \) (resp. \( R \)) contained in the extension of \( D \) (resp. \( S \)) in every model of \( T \)?

- **Instance checking** ($\mathcal{KB} \models C(a)$ or $\mathcal{KB} \models R(a, b)$)
  - Is \( a \) (resp. \( (a, b) \)) a member of concept \( C \) (resp. \( R \)) in \( \mathcal{KB} \), i.e., is the fact \( C(a) \) (resp. \( R(a, b) \)) satisfied by every interpretation of \( \mathcal{KB} \)?

- **Instance retrieval** ($\{ a \mid \mathcal{KB} \models C(a) \}$)
  - Find all members of \( C \) in \( \mathcal{KB} \), i.e., compute all individuals \( a \) s.t. \( C(a) \) is satisfied by every interpretation of \( \mathcal{KB} \)?


Reasoning services for DL-based OWL ontologies

- **Consistency of the knowledge base** ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (T, A)$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?

- **Concept (and role) satisfiability** ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - Is there a model of $\mathcal{KB}$ in which $C$ (resp. $R$) has a nonempty extension?

- **Concept (and role) subsumption** ($\mathcal{KB} \models C \sqsubseteq D$)
  - I.e., is the extension of $C$ (resp. $R$) contained in the extension of $D$ (resp. $S$) in every model of $T$?

- **Instance checking** ($\mathcal{KB} \models C(a)$ or $\mathcal{KB} \models R(a, b)$)
  - Is $a$ (resp. $(a, b)$) a member of concept $C$ (resp. $R$) in $\mathcal{KB}$, i.e., is the fact $C(a)$ (resp. $R(a, b)$) satisfied by every interpretation of $\mathcal{KB}$?

- **Instance retrieval** ($\{a \mid \mathcal{KB} \models C(a)\}$)
  - Find all members of $C$ in $\mathcal{KB}$, i.e., compute all individuals $a$ s.t. $C(a)$ is satisfied by every interpretation of $\mathcal{KB}$.
Reasoning services for DL-based OWL ontologies

- **Consistency of the knowledge base** ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?

- **Concept (and role) satisfiability** ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which $C$ (resp. $R$) has a nonempty extension?

- **Concept (and role) subsumption** ($\mathcal{KB} \models C \sqsubseteq D$)
  - i.e., is the extension of $C$ (resp. $R$) contained in the extension of $D$ (resp. $S$) in every model of $\mathcal{T}$?

- **Instance checking** ($\mathcal{KB} \models C(a)$ or $\mathcal{KB} \models R(a, b)$)
  - is $a$ (resp. $(a, b)$) a member of concept $C$ (resp. $R$) in $\mathcal{KB}$?
  - i.e., is the fact $C(a)$ (resp. $R(a, b)$) satisfied by every interpretation of $\mathcal{KB}$?

- **Instance retrieval** ($\{ a \mid \mathcal{KB} \models C(a) \}$)
  - find all members of $C$ in $\mathcal{KB}$, i.e., compute all individuals $a$ s.t. $C(a)$ is satisfied by every interpretation of $\mathcal{KB}$.
Reasoning services for DL-based OWL ontologies

- **Consistency of the knowledge base** ($\mathcal{KB} \not\models T \subseteq \bot$)
  - Is the $\mathcal{KB} = (T, A)$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?

- **Concept (and role) satisfiability** ($\mathcal{KB} \not\models C \subseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which $C$ (resp. $R$) has a nonempty extension?

- **Concept (and role) subsumption** ($\mathcal{KB} \models C \subseteq D$)
  - i.e., is the extension of $C$ (resp. $R$) contained in the extension of $D$ (resp. $S$) in every model of $T$?

- **Instance checking** ($\mathcal{KB} \models C(a)$ or $\mathcal{KB} \models R(a, b)$)
  - is a (resp. $(a, b)$) a member of concept $C$ (resp. $R$) in $\mathcal{KB}$, i.e., is the fact $C(a)$ (resp. $R(a, b)$) satisfied by every interpretation of $\mathcal{KB}$?

- **Instance retrieval** ($\{a \mid \mathcal{KB} \models C(a)\}$)
  - find all members of $C$ in $\mathcal{KB}$, i.e., compute all individuals $a$ s.t. $C(a)$ is satisfied by every interpretation of $\mathcal{KB}$.
Reasoning services for DL-based OWL ontologies

- Consistency of the knowledge base ($\mathcal{KB} \not\equiv \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (T, A)$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?

- Concept (and role) satisfiability ($\mathcal{KB} \not\equiv C \sqsubseteq \bot$)
  - Is there a model of $\mathcal{KB}$ in which $C$ (resp. $R$) has a nonempty extension?

- Concept (and role) subsumption ($\mathcal{KB} \models C \sqsubseteq D$)
  - i.e., is the extension of $C$ (resp. $R$) contained in the extension of $D$ (resp. $S$) in every model of $T$?

- Instance checking ($\mathcal{KB} \models C(a)$ or $\mathcal{KB} \models R(a, b)$)
  - Is $a$ (resp. $(a, b)$) a member of concept $C$ (resp. $R$) in $\mathcal{KB}$, i.e., is the fact $C(a)$ (resp. $R(a, b)$) satisfied by every interpretation of $\mathcal{KB}$?

- Instance retrieval ($\{a \mid \mathcal{KB} \models C(a)\}$)
  - Find all members of $C$ in $\mathcal{KB}$, i.e., compute all individuals $a$ s.t. $C(a)$ is satisfied by every interpretation of $\mathcal{KB}$.
Reasoning services for DL-based OWL ontologies

- Consistency of the knowledge base ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?

- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - Is there a model of $\mathcal{KB}$ in which $C$ (resp. $R$) has a nonempty extension?

- Concept (and role) subsumption ($\mathcal{KB} \models C \sqsubseteq D$)
  - i.e., is the extension of $C$ (resp. $R$) contained in the extension of $D$ (resp. $S$) in every model of $\mathcal{T}$?

- Instance checking ($\mathcal{KB} \models C(a)$ or $\mathcal{KB} \models R(a, b)$)
  - Is $a$ (resp. $(a, b)$) a member of concept $C$ (resp. $R$) in $\mathcal{KB}$, i.e., is the fact $C(a)$ (resp. $R(a, b)$) satisfied by every interpretation of $\mathcal{KB}$?

- Instance retrieval ($\{a \mid \mathcal{KB} \models C(a)\}$)
  - Find all members of $C$ in $\mathcal{KB}$, i.e., compute all individuals $a$ s.t. $C(a)$ is satisfied by every interpretation of $\mathcal{KB}$.
Reasoning services for DL-based OWL ontologies

- **Consistency of the knowledge base** ($\mathcal{KB} \not\models T \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (T, A)$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?

- **Concept (and role) satisfiability** ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - Is there a model of $\mathcal{KB}$ in which $C$ (resp. $R$) has a nonempty extension?

- **Concept (and role) subsumption** ($\mathcal{KB} \models C \sqsubseteq D$)
  - i.e., is the extension of $C$ (resp. $R$) contained in the extension of $D$ (resp. $S$) in every model of $T$?

- **Instance checking** ($\mathcal{KB} \models C(a)$ or $\mathcal{KB} \models R(a, b)$)
  - Is a (resp. $(a, b)$) a member of concept $C$ (resp. $R$) in $\mathcal{KB}$, i.e., is the fact $C(a)$ (resp. $R(a, b)$) satisfied by every interpretation of $\mathcal{KB}$?

- **Instance retrieval** ($\{a \mid \mathcal{KB} \models C(a)\}$)
  - Find all members of $C$ in $\mathcal{KB}$, i.e., compute all individuals $a$ s.t. $C(a)$ is satisfied by every interpretation of $\mathcal{KB}$.
Reasoning services for DL-based OWL ontologies

- Consistency of the knowledge base ($\mathcal{KB} \neq \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?

- Concept (and role) satisfiability ($\mathcal{KB} \neq C \sqsubseteq \bot$)
  - Is there a model of $\mathcal{KB}$ in which $C$ (resp. $R$) has a nonempty extension?

- Concept (and role) subsumption ($\mathcal{KB} \models C \sqsubseteq D$)
  - i.e., is the extension of $C$ (resp. $R$) contained in the extension of $D$ (resp. $S$) in every model of $\mathcal{T}$?

- Instance checking ($\mathcal{KB} \models C(a)$ or $\mathcal{KB} \models R(a, b)$)
  - Is $a$ (resp. $(a, b)$) a member of concept $C$ (resp. $R$) in $\mathcal{KB}$, i.e., is the fact $C(a)$ (resp. $R(a, b)$) satisfied by every interpretation of $\mathcal{KB}$?

- Instance retrieval ($\{a \mid \mathcal{KB} \models C(a)\}$)
  - Find all members of $C$ in $\mathcal{KB}$, i.e., compute all individuals $a$ s.t. $C(a)$ is satisfied by every interpretation of $\mathcal{KB}$.
Reasoning services for DL-based OWL ontologies

- Consistency of the knowledge base ($\mathcal{KB} \not\models T \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (T, A)$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?

- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which $C$ (resp. $R$) has a nonempty extension?

- Concept (and role) subsumption ($\mathcal{KB} \models C \sqsubseteq D$)
  - i.e., is the extension of $C$ (resp. $R$) contained in the extension of $D$ (resp. $S$) in every model of $T$?

- Instance checking ($\mathcal{KB} \models C(a)$ or $\mathcal{KB} \models R(a, b)$)
  - is $a$ (resp. $(a, b)$) a member of concept $C$ (resp. $R$) in $\mathcal{KB}$, i.e., is the fact $C(a)$ (resp. $R(a, b)$) satisfied by every interpretation of $\mathcal{KB}$?

- Instance retrieval ($\{a \mid \mathcal{KB} \models C(a)\}$)
  - find all members of $C$ in $\mathcal{KB}$, i.e., compute all individuals $a$ s.t. $C(a)$ is satisfied by every interpretation of $\mathcal{KB}$
Reasoning services for DL-based OWL ontologies

- Consistency of the knowledge base ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (T, A)$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?
- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which $C$ (resp. $R$) has a nonempty extension?
- Concept (and role) subsumption ($\mathcal{KB} \models C \sqsubseteq D$)
  - i.e., is the extension of $C$ (resp. $R$) contained in the extension of $D$ (resp. $S$) in every model of $T$?
- Instance checking ($\mathcal{KB} \models C(a)$ or $\mathcal{KB} \models R(a, b)$)
  - is $a$ (resp. $(a, b)$) a member of concept $C$ (resp. $R$) in $\mathcal{KB}$, i.e., is the fact $C(a)$ (resp. $R(a, b)$) satisfied by every interpretation of $\mathcal{KB}$?
- Instance retrieval ($\{a \mid \mathcal{KB} \models C(a)\}$)
  - find all members of $C$ in $\mathcal{KB}$, i.e., compute all individuals $a$ s.t. $C(a)$ is satisfied by every interpretation of $\mathcal{KB}$
Tableau reasoning

- Most common for DL reasoners
- Like for FOL:
  - Unfold the TBox
  - Convert the result into negation normal form
  - Apply the tableau rules to generate more Aboxes
  - Stop when none of the rules are applicable

- $\mathcal{T} \vdash C \sqsubseteq D$ if all Aboxes contain clashes
- $\mathcal{T} \nvdash C \sqsubseteq D$ if some Abox does not contain a clash
Negation Normal Form

- C and D are concepts, R a role
- ¬ only in front of concepts:
  - ¬¬C gives C
  - ¬(C ∩ D) gives ¬C ∪ ¬D
  - ¬(C ∪ D) gives ¬C ∩ ¬D
  - ¬(∀R.C) gives ∃R.¬C
  - ¬(∃R.C) gives ∀R.¬C
Tableau rules

\(\Box\)-rule If \((C_1 \cap C_2)(a) \in S\) but \(S\) does not contain both \(C_1(a)\) and \(C_2(a)\), then
\[ S = S \cup \{C_1(a), C_2(a)\} \]

\(\square\)-rule If \((C_1 \sqcup C_2)(a) \in S\) but \(S\) contains neither \(C_1(a)\) nor \(C_2(a)\), then
\[ S = S \cup \{C_1(a)\} \]
\[ S = S \cup \{C_2(a)\} \]

\(\forall\)-rule If \((\forall R.C)(a) \in S\) and \(S\) contains \(R(a, b)\) but not \(C(b)\), then
\[ S = S \cup \{C(b)\} \]

\(\exists\)-rule If \((\exists R.C)(a) \in S\) and there is no \(b\) such that \(C(b)\) and \(R(a, b)\), then
\[ S = S \cup \{C(b), R(a, b)\} \]
Summary

1. **FOL Recap**
   - Syntax
   - Semantics
   - First Order Structures

2. **Description logics**
   - Introduction
   - Basic DL: $\mathcal{ALC}$
   - Reasoning services