Understanding: DL and Automated Reasoning with OWL
Semantic Web (CSC751)

Ubbo Visser
Department of Computer Science
University of Miami

February 27, 2020
Outline

1. Description logic

2. Model-theoretic semantics of OWL

3. Automated reasoning with OWL

Unicornio(UnicornioBelleza)
Unicornio ⊆ Ficticio
Unicornio ⊆ Animal
Description logic

- It is identified as the decidable fragment of first-order predicate logic, with favorable tradeoffs between expressivity, scalability, and computational complexity.
- DLs are decidable and there are efficient algorithms for reasoning with them available.
- Main purpose: entail implicit knowledge from logic-based semantics.
- During this lecture we learn:
  - Direct model-theoretic semantics.
  - Semantics using a translation into first-order predicate logic.
  - Tableaux algorithm for $\mathcal{ALC}$.
  - Tableaux algorithm for $\mathcal{SHIQ}$, and
  - Computational complexities.
ALC

- **ALC** stands for **Attribute Language with Complement**.
- Basic building blocks of **ALC**: classes, roles, and individuals. Individuals put into relationships with each other.
  - Expression: *Professor(ubboVisser) ⊆ ubboVisser* belongs to class *Professor*.
  - Expression: *hasAffiliation(ubboVisser, universityOfMiami) ⊆ hasAffiliation* abstract role describes that *UbboVisser* is affiliated with *UniversityOfMiami*.
  - Expression: *Professor ⊆ FacultyMember* ⊆ *Professor* is a subclass of the class *FacultyMember*.
  - Expression: *Professor ≡ Prof* ⊆ *Professor* is equivalent to the class *Prof*.
  - Complex class relationships are constructed using **conjunction** \{\(\sqcap, \text{owl:intersectionOf}\}\), **disjunction** \{\(\sqcup, \text{owl:unionOf}\}\), and **negation** \(\neg, \text{owl:complementOf}\)\}. These constructors can be nested arbitrarily.
    - *Professor ⊆ (Person \(\sqcap\) FacultyMember) \(\sqcup\) (Person \(\sqcap\) \(\neg\)PhdStudent).*
### ALC

- **Basic building blocks of ALC:**
  - Complex classes can also be described using quantifiers, which corresponds to role restrictions in OWL. Let $R$ be a role and $C$ a class, then $\{\forall R.C, \text{owl:allValuesFrom}\}$ and $\{\exists R.C, \text{owl:someValuesFrom}\}$ are class expressions. E.g., $\text{Exam} \sqsubseteq \forall \text{hasExaminer.Professor}$ \(\rightsquigarrow\) all examiners of an exam must be professors, and $\text{Exam} \sqsubseteq \exists \text{hasExaminer.Professor}$ \(\rightsquigarrow\) must have at least one examiner who is a professor.
  - Quantifiers can be nested arbitrarily.

- $\bot \equiv \text{owl:Nothing}$; $\bot \equiv C \sqcap \neg C$ for some arbitrary class $C$.
- $\top \equiv \text{owl:Thing}$; $\top \equiv C \sqcup \neg C$ for some arbitrary class $C$.
- $\top \equiv \neg \bot$.
- $\text{owl:disjointWith}$; $C \sqcap D \sqsubseteq \bot \equiv C \sqsubseteq \neg D$ for two classes $C$ and $D$.
- $\text{rdfs:range}$; $\top \sqsubseteq \forall R.C$ states that $C$ is the range of role $R$, and
- $\text{rdfs:domain}$; $\exists R.\top \sqsubseteq C$ states that $C$ is the domain of role $R$. 
Let $A$ be an atomic class (a class name), and let $R$ be an abstract role (extension is direct for concrete roles). Let $C, D$ be class expressions, which will be constructed using following rule,

$$C, D ::= A \mid \top \mid \bot \mid \neg C \mid C \cap D \mid C \cup D \mid \forall R . C \mid \exists R . C.$$ 

- Terminological knowledge (T-Box) axioms (formula):
  - Contains statements of the form $C \equiv D$ or $C \subseteq D$, where $C$ and $D$ are class expressions.
  - Axioms of the form $C \subseteq D$ are called General Class Inclusion (GCI) axioms.

- Assertional knowledge (A-Box) axioms (formula):
  - If $C$ is a class expression, $R$ be a role, and $a, b$ are individuals, then A-Box contains statements of the form $C(a)$, and $R(a, b)$

$$\text{ALC KB} \equiv \text{ALC T-Box} \text{ plus } \text{ALC A-Box}.$$
**ALC to SHOIN(D)**

- We extend **ALC** to **SUXION(D)**, i.e., **ALC ⊆ SUXION(D)**.
- Letters behind these names are systematic: they describe the language constructs allowed in DL.
  - **S** stands for **ALC** plus role transitivity,
  - **H** stands for role hierarchies, i.e., role inclusion axioms,
  - **O** stands for nominals, i.e., for closed classes with one element,
  - **I** stands for inverse roles,
  - **N** stands for cardinality restrictions,
  - **D** stands for datatypes,
  - **F** stands for role functionality,
  - **Q** stands for qualified cardinality restrictions,
  - **R** stands for generalized role inclusion axioms, and
  - **E** stands for existential role restrictions.
\textbf{SHOIN(D)}

- **owl:oneOf**: this represents closed classes (a.k.a. union of nominals) that contains exactly \( \{a_1, \ldots, a_n\} \equiv \{a_1\} \cap \ldots \cap \{a_n\} \subseteq \bot \) individuals.

- **owl:minCardinality, owl:maxCardinality, and owl:cardinality**: \( \geq nR, \leq nR, \text{ and } = nR \). These are part of \textit{unqualified cardinality restrictions}.

- **Individual relationships** for equivalence \( \{a\} \equiv \{b\} \), and disjointness \( \{a\} \cap \{b\} \subseteq \bot \).

- **Role inclusion axioms**: \( R \sqsubseteq S \), and equivalence: \( R \equiv S \).

- **Inverse roles**: \( S \equiv R^- \) states that \( S \) is the inverse of \( R \).

- **Transitivity**: \( \text{Tra}(R) \), and symmetry: \( R \) as \( R \equiv R^- \).

- **Functionality**: \( \top \sqsubseteq 1R \), and inverse functionality: \( \top \sqsubseteq 1R^- \).

- **Datatypes**.

- Role functionality and inverse functionality are implemented using cardinality restrictions. Thus, \( \text{SHOINF}(D) \) is implicit in \( \text{SHOIN}(D) \).
**SHOIN(D) to SHOIQ(D)**

- **Qualified cardinality restrictions:** $\geq_n R.C$, and $\leq_n R.C$. This extends $SHOIN(D)$ to $SHOIQ(D)$

**SHOIQ(D) to SROIQ(D)**

- **Generalized role inclusion:** $R_1 \circ \ldots \circ R_n \subseteq R$ says that the concatenation of $R_1, \ldots, R_n$ is a subrole of $R$.

**OWL description logic variants**

- OWL 1 Full: is not a description logic.
- **OWL 1 DL:** $SHOIN(D)$.
- OWL 1 Lite: $SHIF(D)$.
- OWL 2 Full: is not a description logic.
- **OWL 2 DL:** $SROIQ(D)$.
- OWL 2 EL: $\mathcal{EL}^{++}$.
- OWL 2 QL: DL-Lite.
**$SROIQ(D)$**

- $SROIQ(D)$ has a T-Box for terminological knowledge, A-Box for assertional knowledge, and an **R-Box** for roles.

- Let $R$ be a set of **atomic roles** that represents R-Box, i.e., $R$ contains all role names, all inverse role names ($\{R^-, R\}$), and the **universal (abstract/concrete) role** $U$. $U$ is like $\top$ for roles, that is the superrole of all roles and inverse roles. $U$ relates all possible pairs of individuals.
\textbf{SROIQ(\mathcal{D})}

- Generalized role inclusion axiom: a statement of the form $S_1 \circ \ldots \circ S_n \sqsubseteq R$.
- A set of generalized role inclusion axioms form a \textbf{generalized role hierarchy}.
- \textbf{Generalized role hierarchy} is regular if there exists a \textbf{strict partial order} $\prec$ \hspace{1cm} $(\forall x, y \in X : x \prec y \iff x \leq y \text{ and } x \neq y)$ on $R$ such that:
  - $S \prec R$ iff $S^- \prec R$.
  - \textbf{Every} role inclusion axioms is one of the forms:
    \begin{align*}
    & R \circ R \sqsubseteq R, \quad R^- \sqsubseteq R, \quad S_1 \circ \ldots \circ S_n \sqsubseteq R, \\
    & R \circ S_1 \circ \ldots \circ S_n \sqsubseteq R, \quad S_1 \circ \ldots \circ S_n \circ R \sqsubseteq R \\
    \end{align*}
  s.t. $R$ is non-inverse role name, and $S_i \prec R$ for $i = 1, \ldots, n$.
  - This restriction eliminates cycles in generalized role hierarchies and provides decidability guarantees for $SROIQ(\mathcal{D})$.
  - e.g.,
    \begin{align*}
    & \text{hasParent} \circ \text{hasHusband} \sqsubseteq \text{hasFather}, \text{ and } \text{hasFather} \sqsubseteq \text{hasParent} \text{ enforces} \\
    & \text{hasParent} \prec \text{hasFather} \text{ and } \text{hasFather} \prec \text{hasParent}, \text{ and the role hierarchy is not regular,} \\
    & \text{because } S \prec R \text{ must be strict.}
    \end{align*}
**SROIQ(𝒟)**

- Thus, regular role hierarchies must avoid equivalence. Role equivalence introduces synonyms. But in practice, synonyms are internally replaced by another symbol.

- **Simple roles** guarantees decidability. It is defined as follows:
  - \( \{R, R^-\} \) does not occur on the right-hand side, then it is simple.
  - Inverse of a simple role is simple.
  - If \( R \) occurs only on the right-hand side of a role inclusion axiom, \( S \sqsubseteq R \) with \( S \) simple, then \( R \) is simple.
  - \( R \) does not occur on the right-hand side of a role inclusion axiom containing concatenation \( \circ \).
  - e.g., \( \{R \sqsubseteq R_1; R_1 \circ R_2 \sqsubseteq R_3; R_3 \sqsubseteq R_4\} \)
  - then, the simple roles of the role hierarchy is \( \{R, R^-, R_1, R_1^-, R_2, R_2^-\} \).

- \( SROIQ(𝒟) \) expresses \( \{\text{Tran}(R), R \circ R \sqsubseteq R\}, \{\text{Sym}(R), R^- \sqsubseteq R\}, \text{Asy}(R), \text{Ref}(R), \) and \( \text{Dis}(S, R) \). These axioms are decidable iff they include simple roles (a.k.a. role characteristics).

- Therefore, \( SROIQ(𝒟) \) R-Box is the union of role characteristics and a role hierarchy, and it is regular if its role hierarchy is regular.
**$SROIQ(D)$ KB**

- Given a regular R-Box set $R$, then the class expression set $C$ is defined as:
  - Every class name is a class expression.
  - $\top$ and $\bot$ are class expressions.
  - If $C, D$ are class expressions, $R, S \in R$ and $S$ is simple, $a, a_1, \ldots, a_n$ are individuals, and $n$ is a non-negative integer, then the following are class expressions:
    - $\neg C$, $C \sqcap D$, $C \sqcup D$, $\{a\}$, $\{a_1, \ldots, a_n\}$, $\forall R.C$, $\exists R.C$, $\exists S.Self$, $\leq nR.C$, $\geq nR.C$.
  - **T-Box**: a set of **class inclusion axioms** $C \sqsubseteq D$ and $C \sqsubseteq D$, where $C$ and $D$ are class expressions.
  - **A-Box**: a set of **individual assignments** $C(a)$, $R(a, b)$, or $\neg R(a, b)$, where $C \in C$, $R \in R$ and $a$ and $b$ are individuals.

$SROIQ(D)$ KB $\equiv$ union or regular $SROIQ(D)$ R-Box $R$ $SROIQ(D)$ T-Box and $SROIQ(D)$ A-Box for $R$. 
Model-theoretic semantics of OWL

Vocabulary $V$
- Individual names $I \{ \ldots, a, \ldots \}$
- Class names $C \{ \ldots, C, \ldots \}$
- Roles names $R \{ \ldots, R, \ldots \}$

Interpretation $I$
- $a^I$
- $C^I$
- $\Delta$

Models $\Gamma$
- $I_I$
- $I_C$
- $I_R$

DL
- MT-OWL
- AROWL
Interpreting individuals, classes, and roles

- First we fix the symbols of the vocabulary $V$ through:
  - a set $I$ of symbols for individuals,
  - a set $C$ of symbols for class names, and
  - a set $R$ of symbols for roles.

- Ignoring punning, the sets $I$, $C$, and $R$ must be mutually disjoint.

- There exist a **domain of interpretation** $\Delta$ with a set of entities (resources, individuals or single objects).

- Then we provide **interpretation** for individuals, class names, and roles by means of the **functions**:
  - $I_i : I \rightarrow \Delta$, which maps individuals to elements of the domain,
  - $I_C : C \rightarrow 2^\Delta$, which maps class names to subsets of the domain (**the class extension**), and
  - $I_R : R \rightarrow 2^\Delta \times \Delta$, which maps roles to binary relations of the domain, i.e., a set of pair of elements (**the property extension**).

- $\Delta$ is arbitrary and the implementation of functions $I_i$, $I_C$, and $I_R$ has a lot of freedom.
E.g., (1)

Professor ⊑ FacultyMember

Professor(ubboxVisser)

hasAffiliation(ubboxVisser, uofm)

Let,

\[ \Delta = \{ \spadesuit, \spade, \heartsuit \} \]

\[ I_I(ubboxVisser) = \heartsuit \]

\[ I_I(uofm) = \spadesuit \]

\[ I_C(Professor) = \{ \spadesuit \} \]

\[ I_C(FacultyMember) = \{ \spadesuit, \spade \} \]

\[ I_R(hasAffiliation) = \{ (\spadesuit, \spade), (\spade, \heartsuit) \} \]

*These settings are nonsense, yet, they provide a valid interpretation.*
A word on interpretation

- We mentioned that the mapping is nonsense.
  - The choice of the names in the elements in $\Delta$. In logic, we abstract from these symbols. i.e., we can rename things in $\Delta$ without compromising logical meaning.
  - Whether the interpretation faithfully captures the relations between entities as stated in the knowledge base. $I_l(ubboVisser) \notin I_C(Professor)$, and $(I_l(ubboVisser), I_l(uofm)) \notin R(hasAffiliation)$ although the knowledge base states that it should, i.e., whether the interpretation captures the structure of the knowledge base.
- Interpretations that do make sense for a knowledge base are models of that knowledge base.
Complex class and role expressions

- How do we provide an interpretation for complex classes and role expressions?
- We define an interpretation function \( \mathcal{I} \), which lifts the interpretation of individuals, class names, and roles names to complex classes and role expressions.
- An interpretation for a given SROIQ knowledge base consists of a domain \( \Delta \) and an interpretation function \( \mathcal{I} \) which satisfy the constraints given in the next slide.
- There are many degrees of freedom for choosing \( \Delta, l_I, l_C, \) and \( l_R \). As we have shown in the above example, the interpretations may not intuitively meaningful.
<table>
<thead>
<tr>
<th>Formula</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T^I = \Delta$ and $\bot^I = \emptyset$</td>
<td>-</td>
</tr>
<tr>
<td>$(-C)^I = \Delta \setminus C^I$</td>
<td>$\neg C$ describes things which are not in $C$</td>
</tr>
<tr>
<td>$(C \cap D)^I = C^I \cap D^I$</td>
<td>$C \cap D$ describes things which are both in $C$ and in $D$</td>
</tr>
<tr>
<td>$(C \cup D)^I = C^I \cup D^I$</td>
<td>$C \cup D$ describes things which are both in $C$ or in $D$</td>
</tr>
<tr>
<td>$\exists R.C)^I = {x\mid \text{there is some } y \text{ with } (x, y) \in R^I \cap y \in C^I}$</td>
<td>$\exists R.C$ describes those things which are connected via $R$ with something in $C$</td>
</tr>
<tr>
<td>$(\forall R.C)^I = {x\mid \text{for all } y \text{ with } (x, y) \in R^I \Rightarrow y \in C^I}$</td>
<td>$\forall R.C$ describes those things $x$ for which every $y$ which connects from $x$ via role $R$ is in the class $C$</td>
</tr>
<tr>
<td>$(\leq n R.C)^I = {x\mid #{(x, y) \in R^I \mid y \in C^I} \leq n}$</td>
<td>$\leq n R.C$ describes those things which are connected via $R$ to at most $n$ things in $C$</td>
</tr>
<tr>
<td>$(\geq n R.C)^I = {x\mid #{(x, y) \in R^I \mid y \in C^I} \geq n}$</td>
<td>$\geq n R.C$ describes those things which are connected via $R$ to at least $n$ things in $C$</td>
</tr>
<tr>
<td>${a}^I = {a^I}$</td>
<td>${a}$ describes the class containing only $a$</td>
</tr>
<tr>
<td>$(\exists S.Self)^I = {x \mid (x, x) \in S^I}$</td>
<td>$\exists S.Self$ describes those things which are connected to themselves via $S$</td>
</tr>
<tr>
<td>$(R^-)^I = {(b, a) \mid (a, b) \in R^I}$</td>
<td>for all $R \in R$</td>
</tr>
<tr>
<td>$U^I = \Delta \times \Delta$</td>
<td>for the universal role $U$</td>
</tr>
</tbody>
</table>
Interpreting axioms

- Models capture the structure of the knowledge base. This is done by providing a faithful representation of the axioms in terms of **sets**.
- Models of a knowledge base are interpretations that satisfy additional constraints that are determined by the axioms of the knowledge base.

An interpretation $\mathcal{I}$ of a $SROIQ$ knowledge base $K$ is a model of $K$, $\mathcal{I} \models K$, if the model holds the following additional constraints: 10.8

- If $C(a) \in K$, then $a^\mathcal{I} \in C^\mathcal{I}$.
- If $R(a, b) \in K$, then $(a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}$.
- If $\neg R(a, b) \in K$, then $(a^\mathcal{I}, b^\mathcal{I}) \notin R^\mathcal{I}$.
- If $C \sqsubseteq D \in K$, then $C^\mathcal{I} \subseteq D^\mathcal{I}$.
- If $S \sqsubseteq R \in K$, then $S^\mathcal{I} \subseteq R^\mathcal{I}$.
- If $S_1 \circ \ldots \circ S_n \sqsubseteq R \in K$, then $\{(a_1, a_{n+1}) \in \Delta \times \Delta \mid$ there are $a_1, \ldots, a_n \in \Delta$ such that $(a_i, a_{i+1}) \in S_i^\mathcal{I}$ for all $i = 1, \ldots, n\} \in R^\mathcal{I}$.
- If $\text{Ref}(R) \in K$, then $\{(x, x) \mid x \in \Delta\} \in R^\mathcal{I}$.
- If $\text{Asy}(R) \in K$, then $(x, y) \notin R^\mathcal{I}$ whenever $(y, x) \in R^\mathcal{I}$.
- If $\text{Dis}(R, S) \in K$, then $R^\mathcal{I} \cap S^\mathcal{I} = \emptyset$. 
Revisit e.g., (1)

Based on the definition for the model in the previous slide, we see that the interpretation in e.g., (1) is **NOT** a model for that knowledge base. In order for that interpretation to be a model, it needs to include \((ubboVisser^I, uofm^I) \in hasAffiliation^I\), i.e., 
\[ I_R(hasAffiliation) = \{(\clubsuit, \spadesuit), (\spadesuit, \heartsuit), (\heartsuit, \spadesuit)\} \].

Another model for e.g., (1):

\[
\Delta = \{\alpha, \beta, \gamma\} \\
I_l(ubboVisser) = \beta \\
I_l(uofm) = \alpha \\
I_C(Professor) = \{\beta\} \\
I_C(FacultyMember) = \{\beta, \gamma\} \\
I_R(hasAffiliation) = \{\beta, \alpha\} \\
\]

How many models exist for a knowledge base?
### Logical consequence

<table>
<thead>
<tr>
<th>$\mathcal{I}$</th>
<th>$Model_1$</th>
<th>$Model_2$</th>
<th>$Model_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta$</td>
<td>${\alpha, \beta, \gamma}$</td>
<td>${1, 2}$</td>
<td>${\spadesuit}$</td>
</tr>
<tr>
<td>$I_l(ubboVisser)$</td>
<td>$\beta$</td>
<td>$1$</td>
<td>$\spadesuit$</td>
</tr>
<tr>
<td>$I_l(uofm)$</td>
<td>$\alpha$</td>
<td>$2$</td>
<td>$\spadesuit$</td>
</tr>
<tr>
<td>$I_C(Professor)$</td>
<td>${\beta}$</td>
<td>${1}$</td>
<td>${\spadesuit}$</td>
</tr>
<tr>
<td>$I_C(FacultyMember)$</td>
<td>${\alpha, \beta, \gamma}$</td>
<td>${1, 2}$</td>
<td>${\spadesuit}$</td>
</tr>
<tr>
<td>$I_R(hasAffiliation)$</td>
<td>${(\beta, \alpha)}$</td>
<td>${(1, 1), (1, 2)}$</td>
<td>${(\spadesuit, \spadesuit)}$</td>
</tr>
</tbody>
</table>

- How do we find logical consequences, i.e., implicit knowledge of a knowledge base, from models? We have to consider all the models.
- A model provides a **possible view or realization** of the knowledge base.
- Each model captures the structure of the knowledge base.
- A model could contain additional relations which are not intended.
- From all models, there are things that are **common among** each model and they provide the logical consequence of the knowledge base.
Logical consequence

Let $K$ be a $SROIQ$ knowledge base and $\alpha$ be a general inclusion axiom or an individual assignment. Then $\alpha$ is **logical consequence** of $K$, $K \models \alpha$, if $\alpha^I$ holds in every model $I$ of $K$. i.e.,

<table>
<thead>
<tr>
<th>$K \models C \subseteq D$</th>
<th>iff $(C \subseteq D)^I$ for all $I \models K$</th>
<th>iff $C^I \subseteq D^I$ for all $I \models K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K \models C(a)$</td>
<td>iff $(C(a))^I$ for all $I \models K$</td>
<td>iff $a^I \in C^I$ for all $I \models K$</td>
</tr>
<tr>
<td>$K \models R(a, b)$</td>
<td>iff $(R(a, b))^I$ for all $I \models K$</td>
<td>iff $(a^I, b^I) \in R^I$ for all $I \models K$</td>
</tr>
<tr>
<td>$K \models \neg R(a, b)$</td>
<td>iff $(\neg R(a, b))^I$ for all $I \models K$</td>
<td>iff $(a^I, b^I) \not\in R^I$ for all $I \models K$</td>
</tr>
</tbody>
</table>
E.g., (2)

- Lets formally show that $K \not\models FacultyMember(uofm)$. (NOT a logical consequence).
- This is done by giving a model $\mathcal{M}$ for the knowledge base where $uofm^\mathcal{M} \not\in FacultyMember^\mathcal{M}$.

\[
\Delta = \{\spadesuit, \heartsuit\}
\]

\[
l_I(ubboVisser) = \clubsuit
\]

\[
l_I(uofm) = \spadesuit
\]

\[
l_C(Professor) = \{\heartsuit\}
\]

\[
l_C(FacultyMember) = \{\heartsuit\}
\]

\[
l_R(hasAffiliation) = \{(\heartsuit, \spadesuit)\}
\]
Useful notations for algorithms

- A knowledge base is **satisfiable** or **consistent** if it has **at least one model**.
- A knowledge base **unsatisfiable**, or **contradictory**, or **inconsistent** if it is not satisfiable.
- A class expression $C$ is **satisfiable** if there is a model $\mathcal{I}$ of the knowledge base s.t $C^\mathcal{I} \neq \emptyset$.
- A class expression $C$ is **unsatisfiable** if $C^\mathcal{I} = \emptyset$. This usually points to modeling errors. It also provides provision to build scalable reasoning algorithms. E.g.,

  $$Unicorn(clover\ Jolly\ Bridle)$$ (1)

  $$Unicorn \sqsubseteq Fictitious$$ (2)

  $$Unicorn \sqsubseteq Animal$$ (3)

  $$Fictitious \sqsubseteq Animal \sqsubseteq \bot$$ (4)

  The knowledge base is inconsistent because (4) contradicts (1). If we remove (1), the knowledge base is consistent, but, *Unicorn* is unsatisfiable, as the existence of a *Unicorn* individual leads to a contradiction.
Every $SROIQ$ knowledge base translates a theory in first-order predicate logic with equality.

$\pi(K) = \bigcup_{\alpha \in K} \pi(\alpha)$. $\pi(\alpha)$ definition depends on the type of the axiom $\alpha$.

If $\alpha$ is an individual assignment, then $\pi(\alpha)$ is defined as:

$$
\begin{align*}
\pi(C(a)) &= C(a) \\
\pi(R(a, b)) &= R(a, b) \\
\pi(\neg R(a, b)) &= \neg R(a, b)
\end{align*}
$$
**SROIQ semantics via first-order predicate logic**

- If $\alpha$ is an R-Box statement, then $\pi(\alpha)$ is defined as ($S$ is a role name):

\[
\pi(R_1 \sqsubseteq R_2) = \forall x, y(\pi_{x,y}(R_1) \rightarrow \pi_{x,y}(R_2))
\]

\[
\pi_{x,y}(S) = S(x, y)
\]

\[
\pi_{x,y}(R^\sim) = \pi_{y,x}(R)
\]

\[
\pi_{x,y}(R_1 \circ \ldots \circ R_n) = \exists x_1, \ldots, x_n \left( \pi_{x,x_1}(R_1) \land \bigwedge_{i=1}^{n-2} \pi_{x_i,x_{i+1}}(R_{i+1}) \land \pi_{n-1,y}(R_n) \right)
\]

\[
\pi(\text{Ref}(R)) = \forall x \pi_{x,x}(R)
\]

\[
\pi(\text{Asy}(R)) = \forall x, y(\pi_{x,y}(R) \rightarrow \neg \pi_{y,x}(R))
\]

\[
\pi(\text{Dis}(R_1, R_2)) = \neg(\exists x, y)(\pi_{x,y}(R_1) \land \pi_{x,y}(R_2))
\]
**SROIQ semantics via first-order predicate logic**

- If \( \alpha \) is a class inclusion axiom \( (C \sqsubseteq D) \), then \( \pi(\alpha) \) is defined as \( (A \) is a class name):

\[
\begin{align*}
\pi(C \sqsubseteq D) & = \forall x(\pi_x(C) \rightarrow \pi_x(D)) \\
\pi_x(A) & = A(x) \\
\pi_x(\neg C) & = \neg \pi_x(C) \\
\pi_x(C \cap D) & = \pi_x(C) \land \pi_x(D) \\
\pi_x(C \cup D) & = \pi_x(C) \lor \pi_x(D) \\
\pi_x(\forall R.C) & = \forall x_1(R(x, x_1) \rightarrow \pi_{x_1}(C)) \\
\pi_x(\exists R.C) & = \exists x_1(R(x, x_1) \land \pi_{x_1}(C)) \\
\pi_x(\geq nS.C) & = \exists x_1, \ldots, x_n \left( \bigwedge_{i \neq j} (x_i \neq x_j) \land \bigwedge_i (S(x, x_i) \land \pi_{x_i}(C)) \right) \\
\pi_x(\leq nS.C) & = \neg (\exists x_1, \ldots, x_{n+1}) \left( \bigwedge_{i \neq j} (x_i \neq x_j) \land \bigwedge_i (S(x, x_i) \land \pi_{x_i}(C)) \right) \\
\pi_x(\{a\}) & = (x = a) \\
\pi_x(\exists S.Self) & = S(x, x)
\end{align*}
\]
E.g., (4)

\[
\begin{align*}
\text{Professor} & \sqsubseteq \text{FacultyMember} \\
\forall x (\text{Professor}(x) & \rightarrow \text{FacultyMember}(x)) \\
\text{Professor} & \sqsubseteq (\text{Person} \sqcap \text{FacultyMember}) \sqcup (\text{Person} \sqcap \neg \text{PhdStudent}) \\
\forall x (\text{Professor}(x) & \rightarrow ((\text{Person}(x) \land \text{FacultyMember}(x)) \lor \\
& (\text{Person}(x) \land \neg \text{PhdStudent}(x)))) \\
\text{Exam} & \sqsubseteq \forall \text{hasExaminer} \cdot \text{Professor} \\
\forall x (\text{Exam}(x) & \rightarrow \forall y (\text{hasExaminer}(x, y) \rightarrow \text{Professor}(y))) \\
\text{Exam} & \sqsubseteq \leq \text{hasExaminer} \\
\forall x (\text{Exam}(x) & \rightarrow \neg (\exists x_1, x_2, x_3)((x_1 \neq x_2) \land (x_2 \neq x_3) \land (x_1 \neq x_3) \\
& \text{hasExaminer}(x, x_1) \land \text{hasExaminer}(x, x_2) \land \\
& \text{hasExaminer}(x, x_3)))) \\
\text{Professor}(\text{ubboVisser}) & \quad \text{Professor}(\text{ubboVisser}) \\
\text{hasAffiliation}(\text{ubboVisser}, \text{uofm}) & \quad \text{hasAffiliation}(\text{ubboVisser}, \text{uofm}) \\
\text{hasParent} \circ \text{hasBrother} & \sqsubseteq \text{hasUncle} \\
\forall x, y (\exists x_1 (\text{hasParent}(x, x_1) \land \text{hasBrother}(x_1, y)) & \rightarrow \text{hasUncle}(x, y))
\end{align*}
\]
Automated reasoning with OWL

Tableaux algorithms

- Formal semantics provides implicit knowledge via logical consequence.
- $\alpha$ is a logical consequence of $K$, $K \models \alpha$, if and only if every model of $K$ is a model of $\alpha$.
- An algorithm based on the prior definition requires checking every possible model of the knowledge base, which is not feasible.
- We need an algorithm that finds the logical consequence based on syntax. We use Tableaux algorithms. (Pellet, RacerPro, and FaCT++)
- But its soundness and completeness needed to be proven formally, which requires substantial mathematical build-up.
- We consider only the algorithm, and the proofs are taken for granted.
- We start with tableaux algorithm for $\mathcal{ALC}$. 
### Inference types

| Subsumption or class inclusion. *Structuring knowledge bases* | $C \sqsubseteq D?$ |
| Class equivalence. *Are two classes represent the same class?* | $C \equiv D?$ |
| Class disjointness. *Are their common members?* | $C \cap D \sqsubseteq \bot?$ |
| Global consistency of a knowledge base. *Is the knowledge base meaningful?* | $K \models \text{false}$ |
| Class consistency. *Is C empty?* | $C \sqsubseteq \bot?$ |
| Instance checking. *Is a contained in C?* | $C(a)?$ |
| Instance retrieval. *Find all known individuals belong to a given class.* | $\forall x C(x)?$ |

### Inference problem

- Using tableaux algorithm, we reduce the inference types to each other.
- **We check the knowledge base satisfiability**, i.e., whether the knowledge base has at least one model.
# Inference by reduction to unsatisfiability

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subsumption</td>
<td>$K \models C \subseteq D$ iff $K \cup { (C \cap \neg D)(a) }$ is unsatisfiable, where $a$ is a new individual not occurring in $K$.</td>
</tr>
<tr>
<td>Class equivalence</td>
<td>$K \models C \equiv D$ iff $K \models C \subseteq D$ and $K \models D \subseteq C$.</td>
</tr>
<tr>
<td>Class disjointness</td>
<td>$K \models C \cap D \not\subseteq \bot$ iff $K \cup { (C \cap D)(a) }$ is unsatisfiable, where $a$ is a new individual not occurring in $K$.</td>
</tr>
<tr>
<td>Global consistency</td>
<td>$K$ is globally consistent if it has a model by failure to find a model.</td>
</tr>
<tr>
<td>Class consistency</td>
<td>$K \models C \not\subseteq \bot$ iff $K \cup { C(a) }$ is unsatisfiable, where $a$ is a new individual not occurring in $K$.</td>
</tr>
<tr>
<td>Instance checking</td>
<td>$K \models C(a)$ iff $K \cup { \neg C(a) }$ is unsatisfiable.</td>
</tr>
<tr>
<td>Instance retrieval</td>
<td>To find all individuals belonging to a class $C$, we have to check for all individuals $a$ whether $K \models C(a)$.</td>
</tr>
</tbody>
</table>
Reduction to satisfiability

- Tableaux algorithm determines if a knowledge base is satisfiable.
- It attempts to construct a **model** of the knowledge base in a **general** and an **abstract** manner.
- If the construction fails, then there is **no model** of the knowledge base or knowledge base is unsatisfiable.
- Otherwise the knowledge base is satisfiable.
- The formal proofs that verify these claims are omitted from this lecture.
- The reduction of all inference problems to the checking of unsatisfiability of the knowledge base.
- Keep in mind that tableaux algorithms attempt to construct models, which is why it is used in DL automated reasoning.
Tableaux algorithm for $\mathcal{ALC}$

Preprocessing of $\mathcal{ALC}$ knowledge base

- $\mathcal{ALC}$ A-Box does not allow statements such as $\neg C(a)$ or $(C \sqcap \neg D)(a)$.
- But these are just class expressions. We introduce a new class name $A$ in T-Box with $A \equiv C$ and re-write the A-Box statement as $A(a)$.
- Replace $C \equiv D$ by $C \sqsubseteq D$ and $D \sqsubseteq C$.
- Replace $C \sqsubseteq D$ by $\neg C \sqcup D$.
- Transform the knowledge base $K$ into **Negation Normal Form (NNF)** by applying equations in 5-21 exhaustively.
- $\text{NNF}(K)$ moves all the negation symbols down into subformulae until they occur directly in front of class names.
- $\text{NNF}(K)$ only transforms the T-Box.
- $\text{NNF}(K) = \mathcal{A} \cup \mathcal{R} \cup \bigcup_{C \sqsubseteq D \in K} \text{NNF}(C \sqsubseteq D)$, where $\mathcal{A}$ and $\mathcal{R}$ are the A-Box and the R-Box of $K$.
- $K$ and $\text{NNF}(K)$ are logically equivalent, i.e., they have identical models.
\[\text{NNF}(C \sqsubseteq D) = \text{NNF}(\neg C \sqcup D)\]  \hspace{1cm} (5)

\[\text{NNF}(C) = C \quad \text{if } C \text{ is a class name} \]  \hspace{1cm} (6)

\[\text{NNF}(\neg C) = \neg C \quad \text{if } C \text{ is a class name} \]  \hspace{1cm} (7)

\[\text{NNF}(\neg \neg C) = \text{NNF}(C) \]  \hspace{1cm} (8)

\[\text{NNF}(C \sqcup D) = \text{NNF}(C) \sqcup \text{NNF}(D) \]  \hspace{1cm} (9)

\[\text{NNF}(C \sqcap D) = \text{NNF}(C) \sqcap \text{NNF}(D) \]  \hspace{1cm} (10)

\[\text{NNF}(\neg(C \sqcup D)) = \text{NNF}(\neg C) \sqcap \text{NNF}(\neg D) \]  \hspace{1cm} (11)

\[\text{NNF}(\neg(C \sqcap D)) = \text{NNF}(\neg C) \sqcup \text{NNF}(\neg D) \]  \hspace{1cm} (12)

\[\text{NNF}(\forall R.C) = \forall R.\text{NNF}(C) \]  \hspace{1cm} (13)

\[\text{NNF}(\exists R.C) = \exists R.\text{NNF}(C) \]  \hspace{1cm} (14)

\[\text{NNF}(\neg \forall R.C) = \exists R.\text{NNF}(\neg C) \]  \hspace{1cm} (15)

\[\text{NNF}(\neg \exists R.C) = \forall R.\text{NNF}(\neg C) \]  \hspace{1cm} (16)

\[\text{NNF}(\leq n R.C) = \leq n R.\text{NNF}(C) \]  \hspace{1cm} (17)

\[\text{NNF}(\geq n R.C) = \geq n R.\text{NNF}(C) \]  \hspace{1cm} (18)

\[\text{NNF}(\neg \leq n R.C) = \geq (n+1) R.\text{NNF}(C) \]  \hspace{1cm} (19)

\[\text{NNF}(\neg \geq (n+1) R.\text{NNF}(C)) = \leq n R.\text{NNF}(C) \]  \hspace{1cm} (20)

\[\text{NNF}(\neg \geq 0 R.C) = \bot \]  \hspace{1cm} (21)
E.g.,

\[ P \subseteq (E \cap U) \cup \neg (\neg E \cup D) \]

Let's transform this formula to NNF

\[ \neg P \cup (E \cap U) \cup \neg (\neg E \cup D) \]

\[ \neg P \cup (E \cap U) \cup (E \cap \neg D) \]

Naïve Tableaux algorithm

- Reduction to unsatisfiability/satisfiability.
- Given: the knowledge base $K$.
- Construct: a special graph called the Tableaux, which represents a model of $K$.
- If this construction fails, then $K$ is unsatisfiable.

Tableaux

- A node represents an element of the domain:
  Every node $x$ is labeled with a set $\mathcal{L}(x)$ of class expressions, i.e., $C \in \mathcal{L}(x)$ means “$x$ is in the extension of $C$”. $\forall x \ T \in \mathcal{L}(x)$, we often do not write this down, and the tableaux does not explicitly derive this.

- An edge represents a role relationship:
  Every edge $(x, y)$ is labeled with a set $\mathcal{L}(x, y)$ of role names, i.e., $R \in \mathcal{L}(x, y)$ means “$(x, y)$ is in the extension of $R$”.

- This is a structured way of deriving and representing logical consequence of a knowledge base.
Assume that the knowledge base is transformed to NNF.

\[ K \models C(a) \] \hspace{1cm} (22)
\[ K \models (¬C \sqcap D)(a) \] \hspace{1cm} (23)
\[ (¬C \sqcap D)(a) \models \neg C(a) \] \hspace{1cm} (24)

Formulae 22 and 24 cause a contradiction. Therefore, \( K \) cannot have a model and it is unsatisfiable.

We just constructed a part of tableaux and a contradiction is found. This means that the initial knowledge base is unsatisfiable.
Illustration

- Let,

\[ K \models C(a) \]
\[ K \models \neg C \sqcap D \]
\[ K \models \neg D(a) \]

- We want to derive all class memberships of \( a, \mathcal{L}(a) \).

- Some notations:
  - \( \mathcal{L}(a) \leftarrow C \) means \( \mathcal{L}(a) \) is updated by adding \( C \).
  - If \( \mathcal{L}(a) = \{D\} \), then \( \mathcal{L}(a) \leftarrow C \) causes \( \mathcal{L}(a) = \{C, D\} \).
  - \( \mathcal{L}(a) \leftarrow \{C, D\} \) means subsequent application of \( \mathcal{L}(a) \leftarrow C \) and \( \mathcal{L}(a) \leftarrow D \), which both \( C \) and \( D \) add to \( \mathcal{L}(a) \).
Illustration continued

\[ K \models C(a) \quad (25) \]
\[ K \models \neg C \sqcup D \quad (26) \]
\[ K \models \neg D(a) \quad (27) \]

- From 25, \( L(a) \leftarrow C \), and 27, \( L(a) \leftarrow \neg D \): \( L(a) = \{C, \neg D\} \).
- 26 is a T-Box statement and it might as well hold for \( a \): \( L(a) \leftarrow \neg C \sqcup D \).
- \((\neg C \sqcup D) \in L(a)\), which means that \( \neg C(a) \) or \( D(a) \). This introduces two new cases:
  - If \( \neg C(a) \), then \( L(a) \leftarrow \neg C = \{C, \neg D, \neg C \sqcup D, \neg C\} \), which is a contradiction.
  - If \( D(a) \), then \( L(a) \leftarrow \neg D = \{C, \neg D, \neg C \sqcup D, D\} \), which is a contradiction.
  - Both cases we arrive at a contradiction, which indicates that \( K \) is unsatisfiable.
- Branching leads to nondeterminism of the tableaux algorithm.
Illustration: Roles

\[ K \models R(a, b) \]
\[ K \models S(a, a) \]
\[ K \models R(a, c) \]
\[ K \models S(b, c) \]

![Diagram](image_url)

\[ K \models \exists R. \exists S. C(a) \]

\[ a \xrightarrow{R} x \xrightarrow{S} y \]
Tableaux example

\[ K = \{ C(a), C \sqsubseteq \exists R.D, D \sqsubseteq E \} \]
\[ NNF(K) = \{ C(a), \neg C \sqcup \exists R.D, \neg D \sqcup E \} \]

Is \((\exists R.E)(a)\) a logical consequence of \(K\)?

From inference by reduction to unsatisfiability table:

| Instance checking | \( K \models C(a) \) iff \( K \cup \{ \neg C(a) \} \) is unsatisfiable. |

Therefore, we need to show that \( K \cup \{ \neg (\exists R.E)(a) \} \) is unsatisfiable. From 16, \( NNF(\exists R.E) = \forall R. \neg E \).

\[ NNF(K) = \{ C(a), \neg C \sqcup \exists R.D, \neg D \sqcup E, \forall R. \neg E(a) \} \],

which we need to show that \( NNF(K) \) is unsatisfiable.
The naïve tableaux algorithm for $\mathcal{ALC}$

A tableaux for an $\mathcal{ALC}$ knowledge base consists of:

- a set of nodes, labeled with individual names or variable names,
- directed edges between some pairs of nodes,
- for each node labeled $x$, a set $\mathcal{L}(x)$ of class expressions, and
- for each pair of nodes $x$ and $y$, a set $\mathcal{L}(x, y)$ of role names.

Algorithm

**Algorithm 1:** NAIVE\_ALC\_Tableaux($\text{NNF}(K)$)

**Data:** $\text{NNF}(K)$

**Result:** Satisfiability status of $K$

$\text{initialTableaux} = \text{INITIALIZE\_Tableaux}(\text{NNF}(K));$

return $\text{APPLY\_RULES}(\text{initialTableaux}, \text{NNF}(K));$
Algorithm 2: INITIALIZE_Tableaux($NNF(K)$)

**Data:** $NNF(K)$

**Result:** Initial tableaux

- For each individual $a$ occurring in $K$, create a node labeled $a$ and set $L(a) = \emptyset$.
- For all pairs $a, b$ of individuals, set $L(a, b) = \emptyset$.
- For each A-Box statement $C(a)$ in $K$, set $L(a) \leftarrow C$.
- For each R-Box statement $R(a, b)$ in $K$, set $L(a, b) \leftarrow R$. 
Algorithm 3: APPLY_RULES($\text{initialTableaux}$, $\text{NNF}(K)$)

- **In each step**, nondeterministically apply the following rules:
  - $\Box$-rule: If $C \Box D \in \mathcal{L}(x)$ and $\{C, D\} \not\subseteq \mathcal{L}(x)$, then set $\mathcal{L}(x) \leftarrow \{C, D\}$.
  - $\exists$-rule: If $\exists R.C \in \mathcal{L}(x)$ and $\{C, D\} \cap \mathcal{L}(x) = \emptyset$, then set $\mathcal{L}(x) \leftarrow C$ or $\mathcal{L}(x) \leftarrow D$.
  - $\exists$-rule: If $\exists R.C \in \mathcal{L}(x)$ and there exists no $y$ with $R \in \mathcal{L}(x, y)$ and $C \in \mathcal{L}(y)$, then
    - add a new node with label $y$ (where $y$ is a new node label),
    - set $\mathcal{L}(x, y) = \{R\}$, and
    - set $\mathcal{L}(y) = \{C\}$.
  - $\forall$-rule: If $\forall R.C \in \mathcal{L}(x)$ and there is a node $y$ with $R \in \mathcal{L}(x, y)$ and $C \not\in \mathcal{L}(y)$, then set $\mathcal{L}(y) \leftarrow C$.
  - T-Box-rule: If $C$ is a T-Box statement and $C \not\in \mathcal{L}(x)$, then set $\mathcal{L}(x) \leftarrow C$.

- **Terminates**, if
  - either there is a node $x$ such that $\mathcal{L}(x)$ contains a contradiction, i.e., if there is $C \in \mathcal{L}(x)$ and at the same time $\neg C \in \mathcal{L}(x)$ (also apply for $\top, \bot$),
  - or none of the rules are applicable.
Tableaux example

\[ NNF(K) = \{ A(a), (\exists R.B)(a), R(a, b), R(a, c), S(b, b), (A \sqcup B)(c), \neg A \sqcup (\forall S.B) \} \]

From Algorithm 2,

\[ \mathcal{L}(b) = \emptyset \]

\[ \mathcal{L}(a) = \{ A, \exists R.B \} \]

\[ \mathcal{L}(c) = \{ A \sqcup B \} \]
## An explanation of Algorithm 3

- **$K$ is satisfiable** if the Algorithm 3 terminates without contradiction, otherwise $K$ is unsatisfiable.

- **Sources of nondeterminism.**
  - **Which expansion rule to apply next:** whatever rule we choose, it will **not** get us into wrong track, though the algorithm may take more steps to terminate. This leads to **don’t care nondeterminism.**
  - **The choice which has to be made when applying the $\sqcup$-rule:** bad choice get us on to the wrong track. This is because, if we choose to set $\mathcal{L}(x) \leftarrow \mathcal{C}$, then it is no longer possible to set $\mathcal{L}(x) \leftarrow D$ as the rule $\{C, D\} \cap \mathcal{L}(x) = \emptyset$ prevent this. If the choice leads to a contradiction, then we have to backtrack to that choice point and try another alternative. This leads to **don’t know nondeterminism.**

- **If you can make a choice of rule applications such that no contradiction occurs and the process terminates,** then the knowledge base is satisfiable.
**Tableaux example**

- \( K = \{ C(a), \forall R. D, D \subseteq E \} \)
- Question: \( K \models (\exists R. E)(a) \)
  - Problem: Instance checking.
  - Solution: \( K \models C(a) \) iff \( K \cup \{ \neg C(a) \} \) is unsatisfiable.
  - \( NNF(\neg(\exists R. E)(a)) = \forall R. \neg E(a) \)
  - \( NNF(K) = \{ C(a), \neg C \sqcup \exists R. D, \neg D \sqcup E, \forall R. \neg E(a) \} \)

**Algorithm**

- \( L(a) = \{ C, \forall R. \neg E \} \)
- \( L(a) \leftarrow \neg C \sqcup \exists R. D \)
- \( L(a) \leftarrow \neg C \) contradiction.
- \( L(a) \leftarrow \exists R. D \)
- \( L(x) \leftarrow \neg D \sqcup E \)
- \( L(x) \leftarrow \neg D \) contradiction.
- \( L(x) \leftarrow E \)
- \( L(x) \leftarrow \neg E(\forall\text{-rule}) \) contradiction.

**Tableaux**

\[
\begin{array}{c|c}
  a & L(a) = \{ C, \forall R. \neg E, \exists R. D \} \\
  \hline
  R & \\
  \hline
  y & L(x) = \{ D, \neg D \sqcup E, E, \neg E \} \\
  \hline
  x & \text{contradiction}
\end{array}
\]
Tableaux example

- $K = \{ C(a), C \sqsubseteq \exists R.D, D \sqsubseteq E \}$
- Question: $K \models (\exists R.E)(a)$
- Problem: Instance checking.
  - Solution: $K \models C(a)$ iff $K \cup \{ \neg C(a) \}$ is unsatisfiable.
  - $\text{NNF}(\neg (\exists R.E)(a)) = \forall R.\neg E(a)$
  - $\text{NNF}(K) = \{ C(a), \neg C \sqcup \exists R.D, \neg D \sqcup E, \forall R.\neg E(a) \}$

Algorithm

- $\mathcal{L}(a) = \{ C, \forall R.\neg E \}$
- $\mathcal{L}(a) \leftarrow \neg C \sqcup \exists R.D$
- $\mathcal{L}(a) \leftarrow \neg C$ contradiction.
- $\mathcal{L}(a) \leftarrow \exists R.D$
- $\mathcal{L}(x) \leftarrow \neg D \sqcup E$
- $\mathcal{L}(x) \leftarrow \neg D$ contradiction.
- $\mathcal{L}(x) \leftarrow E$
- $\mathcal{L}(x) \leftarrow \neg E(\forall\text{-rule})$ contradiction.

Tableaux

\[
\begin{array}{c}
a \quad \mathcal{L}(a) = \{ C, \forall R.\neg E, \exists R.D \} \\
\quad \downarrow R \\
\quad \downarrow Y \\
\quad x \quad \mathcal{L}(x) = \{ D, \neg D \sqcup E, E, \neg E \} \quad \text{contradiction}
\end{array}
\]
Tableaux example

- $K = \{C(a), C \sqsubseteq \exists R.D, D \sqsubseteq E\}$
- Question: $K \models (\exists R.E)(a)$
- Problem: Instance checking.
- Solution: $K \models C(a)$ iff $K \cup \{\neg C(a)\}$ is unsatisfiable.
- $\text{NNF}(\neg (\exists R.E)(a)) = \forall R.\neg E(a)$
- $\text{NNF}(K) = \{C(a), \neg C \sqcup \exists R.D, \neg D \sqcup E, \forall R.\neg E(a)\}$

Algorithm

- $\mathcal{L}(a) = \{C, \forall R.\neg E\}$
- $\mathcal{L}(a) \leftarrow \neg C \sqcup \exists R.D$
- $\mathcal{L}(a) \leftarrow \neg C$ contradiction.
- $\mathcal{L}(a) \leftarrow \exists R.D$
- $\mathcal{L}(x) \leftarrow \neg D \sqcup E$
- $\mathcal{L}(x) \leftarrow \neg D$ contradiction.
- $\mathcal{L}(x) \leftarrow E$
- $\mathcal{L}(x) \leftarrow \neg E(\forall\text{-rule})$ contradiction.
Tableaux example

- $K = \{C(a), C \sqsubseteq \exists R.D, D \sqsubseteq E \sqcup F, F \sqsubseteq E\}$
- **Question:** $K \models (\exists R.E)(a)$
- **Problem:** Instance checking.
- **Solution:** $K \models C(a)$ iff $K \cup \{\neg C(a)\}$ is unsatisfiable.
- $\text{NNF}(\neg(\exists R.E)(a)) = \forall R.\neg E(a)$
- $\text{NNF}(K) = \{C(a), \neg C \sqcup \exists R.D, \neg D \sqcup E \sqcup F, \neg F \sqcup E \forall R.\neg E(a)\}$

Algorithm

- $L(x) \leftarrow \neg E(\forall\text{-rule})$
- $L(x) \leftarrow \neg D \sqcup E \sqcup F$
- $L(x) \leftarrow \neg D$ contradiction.
- $L(x) \leftarrow E \sqcup F$
- $L(x) \leftarrow E$ contradiction.
- $L(x) \leftarrow F$
- $L(x) \leftarrow \neg F \sqcup E$
- $L(x) \leftarrow \neg F$ contradiction.
- $L(x) \leftarrow E$ contradiction.
Tableaux example

- $K = \{C(a), C \sqsubseteq \exists R.D, D \sqsubseteq E \sqcup F, F \sqsubseteq E\}$
- Question: $K \models (\exists R.E)(a)$
- Problem: Instance checking.
- Solution: $K \models C(a)$ iff $K \cup \{\neg C(a)\}$ is unsatisfiable.
- $\text{NNF}(\neg(\exists R.E)(a)) = \forall R.\neg E(a)$
- $\text{NNF}(K) = \{C(a), \neg C \sqcup \exists R.D, \neg D \sqcup E \sqcup F, \neg F \sqcup E \forall R.\neg E(a)\}$

Algorithm

- $\mathcal{L}(x) \leftarrow \neg E (\forall\text{-rule})$
- $\mathcal{L}(x) \leftarrow \neg D \sqcup E \sqcup F$
- $\mathcal{L}(x) \leftarrow \neg D$ contradiction.
- $\mathcal{L}(x) \leftarrow E \sqcup F$
- $\mathcal{L}(x) \leftarrow E$ contradiction.
- $\mathcal{L}(x) \leftarrow F$
- $\mathcal{L}(x) \leftarrow \neg F \sqcup E$
- $\mathcal{L}(x) \leftarrow \neg F$ contradiction.
- $\mathcal{L}(x) \leftarrow E$ contradiction.
Tableaux

\[ \mathcal{L}(a) = \{ C, \forall R. \neg E, \exists R. D \} \]

\[ \mathcal{L}(x) = \{ D, \neg E, \neg D \sqcup E \sqcup F, E \sqcup D, F, \neg F \sqcup E, E \} \]
### Tableaux example

- \( \text{Human} \sqsubseteq \exists \text{hasParent} \cdot \text{Human} \)
- \( \text{Orphan} \sqsubseteq \text{Human} \cap \forall \text{hasParent} . \neg \text{Alive} \)
- \( \text{Orphan(harryPotter)} \)
- \( \text{hasParent(harryPotter, jamesPotter)} \)

- \( K \models \neg \text{Alive(jamesPotter)}? \)
- We need
  \( \neg \neg \text{Alive(jamesPotter)} = \text{Alive(jamesPotter)} \)
  and show \( NNF(K \cup \text{Alive(jamesPotter)}) \)
  unsatisfiable.

\[
\begin{align*}
\neg H & \cup \exists P . H \\
\neg O & \cup (H \cap \forall P . \neg A) \\
O(h) & \\
P(h, j) & \\
A(j) &
\end{align*}
\]

### Algorithm

\[
\begin{align*}
h \quad \mathcal{L}(h) &= \{O\} \\
\downarrow P \\
j \quad \mathcal{L}(j) &= \{A\}
\end{align*}
\]

- T-Box-rule: \( \mathcal{L}(h) \leftarrow \neg O \cup (H \cap \forall P . \neg A) \)
- \( \sqcup \)-rule: \( \mathcal{L}(h) \leftarrow \neg O \) contradiction.
- \( \sqcap \)-rule: \( \mathcal{L}(h) \leftarrow \{H, \forall P . \neg A\} \)
- \( \sqcap \)-rule: \( \forall P . \neg A \in \mathcal{L}(h) \)
- \( \sqcup \)-rule: \( \forall P . \neg A \in \mathcal{L}(h) \)
- \( \mathcal{L}(j) \leftarrow \neg A \) contradiction.
Tableaux

\[ h \quad \mathcal{L}(h) = \{ O, \neg O \cup (H \cap \forall P. \neg A), H \cap \forall P. \neg A, H, \forall P. \neg A \} \]

\[ j \quad \mathcal{L}(j) = \{ A, \neg A \} \]
Tableaux example

\[ NNF(K) = \{ C(a), \neg C \sqcup \exists R.D, \neg D \sqcup E, \forall R.\neg E(a) \} \]

- From Algorithm 2,
  \[ a \quad \mathcal{L}(a) = \{ C, \forall R.\neg E \} \]
- From Algorithm 3,
  - T-Box-rule: \( \mathcal{L}(a) \leftarrow \neg C \sqcup \exists R.C \).
  - \( \sqcup \)-rule: \( \mathcal{L}(a) \leftarrow \neg C \) contradicts with \( C \).
  - \( \mathcal{L}(a) \leftarrow \exists R.D \).
  - \( \exists \)-rule: \( a \quad \mathcal{L}(a) = \{ C, \forall R.\neg E, \neg C \sqcup \exists R.D, \exists R.D \} \)
  \[ \begin{array}{c}
  R \\
  \downarrow \\
  x \\
  \end{array} \]
  \[ \mathcal{L}(x) = \{ D \} \]
Tableaux example

- From Algorithm 3,
  - T-Box-rule: $\mathcal{L}(x) \leftarrow \neg D \sqcup E$.
  - $\sqcup$-rule: $\mathcal{L}(x) \leftarrow \neg D$ contradicts with $D$.
  - $\mathcal{L}(x) \leftarrow E$

  \[
  a \quad \mathcal{L}(a) = \{ C, \forall R. \neg E, \neg C \sqcup \exists R.D, \exists R.D \}
  \]

  \[
  x \quad \mathcal{L}(x) = \{ D, \neg D \sqcup E, E \}
  \]

- $\forall R. \neg E \in \mathcal{L}(a)$, means that everything to which $a$ connects via $R$ must be in $\neg E$. Since, $a$ connects to $x$ via $R$, we set $\mathcal{L}(x) \leftarrow \neg E$, which results in a contradiction.

- Therefore, the knowledge base is unsatisfiable, and the instance checking problem is solved, i.e., $K \models (\exists R.E)(a)$.

  \[
  a \quad \mathcal{L}(a) = \{ C, \forall R. \neg E, \neg C \sqcup \exists R.D, \exists R.D \}
  \]

  \[
  x \quad \mathcal{L}(x) = \{ D, \neg D \sqcup E, E, \neg E \}
  \]
The tableaux algorithm with blocking for $\mathcal{ALC}$

- Algorithm 1 for $\mathcal{ALC}$ does not always terminate.
- Consider: $K = \{ \exists R. T, T(a_1) \}$.
  - Consider the interpretation $I$, with $\Delta = \{ a_1, a_2, \ldots \}$, s.t. $a_i^I = a_i$ and $(a_i, a_{i+1}) \in R^I$ for all $i = 1, 2, \ldots$. This is a model of $K$. Therefore, $K$ is satisfiable.

Let’s try to construct the tableaux for $K$.
- We initialize with a node $a$ and $\mathcal{L}(a) = \{ \top \}$.
- T-Box-rule: $\mathcal{L}(a) \leftarrow \exists R. T$.
- $\exists$-rule: creates a new node $x$ with $\mathcal{L}(x, a) = \{ R \}$ and $\mathcal{L}(x) = \{ \top \}$.
- For the new $x$ we again apply the T-Box-rule, which yields into $\mathcal{L}(x) \leftarrow \exists R. T$.
- $\exists$-rule: creates another new node $y$ with $\mathcal{L}(x, y) = \{ R \}$ and $\mathcal{L}(y) = \{ \top \}$.
- This process repeats and does not terminate.

$$ a_1 \mathcal{L}(a_1) = \{ \top, \exists R. T \} \xrightarrow{R} x \mathcal{L}(x) = \{ \top, \exists R. T \} \xrightarrow{R} y \mathcal{L}(y) = \{ \top, \exists R. T \} \xrightarrow{R} \ldots $$
We said that $ALC$ or $SROIQ$ is decidable.

In order to achieve guaranteed termination, we need to introduce blocking. This simply eliminates the repeats.

If the newly created node $x$ has the same properties as the node $a_1$, then instead of expanding $x$ to a new node $y$, we reuse $a_1$.

Definition: A node with label $x$ is directly blocked by a node with label $y$ if

- $x$ is a variable (i.e., not an individual)
- $y$ is an ancestor of $x$, and
- $\mathcal{L}(x) \subseteq \mathcal{L}(y)$.  

Definition of ancestor: $\forall x \mathcal{L}(z, x) \neq \emptyset$ is called a predecessor or $x$. Every predecessor of $x$, which is not an individual, is an ancestor or $x$, and every predecessor or ancestor or $x$, which is not an individual, is also an ancestor or $x$.

A node with label $x$ is blocked if it is directly blocked or one of its ancestors is blocked.

Full tableaux algorithm: The rules in Algorithm 3 are applied if $x$ is not blocked.

From our example, $\mathcal{L}(x) \subseteq \mathcal{L}(a_1)$. Therefore, $x$ is blocked by $a_1$. The resulting tableaux is:

$$a_1 \mathcal{L}(a_1) = \{ T, \exists R. T \} \xrightarrow{R} x \mathcal{L}(x) = \{ T \}$$

The blocked node $x$ represents the infinite set $\{ a_2, a_3, \ldots \}$.

Therefore, $\mathcal{J}$ is, $\Delta = \{ a_1, a \}$ s.t $a_1^J = a_1, x^J = a$ and $R^J = \{(a_1, a), (a, a)\}$. The model is cyclic.
Blocking example

\[ K = \{ H \sqsubseteq \exists P. H, B(t) \} \]

One interpretation: \( \text{Human} \sqsubseteq \exists \text{hasParent}. \text{Human}, \text{Bird}(\text{tweety}) \)

Question: \( K \models \neg H(t) ? \)

\( \text{NNF}(K') = \{ \neg H \sqcup \exists P. H, B(t), H(t) \} \)

<table>
<thead>
<tr>
<th>Initialized</th>
<th>Database ( \mathcal{L}(t) = { B, H } )</th>
<th>( t )</th>
<th>Database ( \mathcal{L}(t) = { H, B, \neg H \sqcup \exists P. H, \exists P. H } )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-Box-rule</td>
<td>Database ( \mathcal{L}(t) \leftarrow \neg H \sqcup \exists P. H )</td>
<td></td>
<td>( t )</td>
</tr>
<tr>
<td>( \sqcap )-rule</td>
<td>Database ( \mathcal{L}(t) \leftarrow \neg H ) (contradiction)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \exists )-rule</td>
<td>create a node with label ( x ), Database ( \mathcal{L}(t, x) = { R } ), and Database ( \mathcal{L}(x) = { H } )</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>node ( x ) is blocked by ( t )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Node \( x \) is blocked by \( t \).
Open world assumption

Let

\[ K = \{ h(j, p), h(j, a), M(p), M(a) \} \]

which stands for

\[ \text{hasChild}(\text{john}, \text{peter}), \text{hasChild}(\text{john}, \text{alex}), \text{Male}(\text{peter}), \text{Male}(\text{alex}) \]

\[ K \not\models \forall \text{hasChild}.\text{Male}(\text{john}) \] (not a logical consequence of the knowledge base)

\[ (K \not\models \forall x \text{hasChild}(x, \text{john}) \rightarrow \text{Male}(\text{john})). \]

Add the negation of the statement \( \neg \forall h. M(j) \) to \( K \), and show that \( \text{NNF}(K') \) is satisfiable.

OWA for \( K' \) satisfiability:

- There is no information whether or not \textit{john} has only \textit{peter} and \textit{alex} as children.
- There may be that \textit{john} has additional children who are not listed in \( K' \).
- Therefore, it is not possible to infer that all \textit{john}’s children are \textit{Male}.
Illustration

- $\text{NNF}(K') = \{ \text{h}(j, p), \text{h}(j, a), M(p), M(a), \exists h. \neg M(j) \}$. 
- Algorithm 2 yields:
  \[ \begin{array}{c}
  p \\
  \downarrow \text{h} \\
  j \\
  \downarrow \text{h} \\
  a \\
  \downarrow \text{h} \\
  x \\
  \end{array} \]
  \[ \mathcal{L}(p) = \{ M \} \]
  \[ \mathcal{L}(j) = \{ \exists h. \neg M \} \]
  \[ \mathcal{L}(a) = \{ M \} \]
- Algorithm 3 yields: $\exists$-rule $\mathcal{L}(j, x) = \{ h \}$ and $\mathcal{L}(x) = \{ \neg M \}$.
  \[ \begin{array}{c}
  p \\
  \downarrow \text{h} \\
  j \\
  \downarrow \text{h} \\
  a \\
  \downarrow \text{h} \\
  x \\
  \end{array} \]
  \[ \mathcal{L}(p) = \{ M \} \]
  \[ \mathcal{L}(j) = \{ \exists h. \neg M \} \]
  \[ \mathcal{L}(a) = \{ M \} \]
  \[ \mathcal{L}(x) = \{ \neg M \} \]
Illustration

- Algorithm 1 terminates, since none of the rules are applicable. This means that $K'$ is satisfiable. It means that $\forall h. M(j)$ is not a logical consequence of $K$.
- New node $x$ represents a potential child of $john$ who is not a $Male$.
- Indeed the constructed tableaux corresponds to a model of $K'$. 
Final illustration

\[ K = \{ C(a), C(c), R(a, b), R(a, c), S(a, a), S(c, b), C \sqsubseteq \forall S.A, \]
\[ A \sqsubseteq \exists R.\exists S.A, A \sqsubseteq \exists R.C \} \]

- Question \( K \models \exists R.\exists R.\exists S.A(a) \).
- Tableaux can grow considerably large if the expansion rules are chosen randomly!
- Follow this:
  - T-Box-Rule on \( c \rightarrow \neg C \sqcup \forall S.A \).
  - \( \forall \)-rule \( \forall S.A \in \mathcal{L}(c) \).
  - \( \forall \)-rule \( \forall R.\forall R.\forall S.\neg A \in \mathcal{L}(a) \).
  - T-Box-rule on \( b \rightarrow \neg A \sqcup \exists R.\exists S.A \).
  - \( \sqcup \)-rule on \( \neg A \sqcup \exists R.\exists S.A \in \mathcal{L}(b) \).
  - \( \forall \)-rule \( \forall R.\forall S.\neg A \in \mathcal{L}(a) \).
  - \( \exists \)-rule on the new node \( x \exists S.A \in \mathcal{L}(x) \).
  - \( \forall \)-rule \( \forall S.\neg A \in \mathcal{L}(x) \) homes you in a contraction.
  - Draw the tableaux.
### Worst-case complexity classes of some description logic

<table>
<thead>
<tr>
<th>Description logic</th>
<th>Combined complexity</th>
<th>Data complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALC</td>
<td>ExpTime-complete</td>
<td>NP-complete</td>
</tr>
<tr>
<td>SHIQ</td>
<td>ExpTime-complete</td>
<td>NP-complete</td>
</tr>
<tr>
<td>SHOIN(D)</td>
<td>NExpTime-complete</td>
<td>NP-hard</td>
</tr>
<tr>
<td>SROIQ(D)</td>
<td>N2ExpTime-complete</td>
<td>NP-hard</td>
</tr>
<tr>
<td>EL++</td>
<td>P-complete</td>
<td>P-complete</td>
</tr>
<tr>
<td>DLP</td>
<td>P-complete</td>
<td>P-complete</td>
</tr>
<tr>
<td>DL-Lite</td>
<td>P</td>
<td>LOGSPACE</td>
</tr>
</tbody>
</table>

- Complexity of the description logics are usually measured in terms of the size of the knowledge base **combined complexity**.
- Complexity is measured only using ABox is called **data complexity**.
The $ALC$ full tableau algorithm has been extended to $SHIQ$ adding two more constrains to Algorithm 2 and few more rule to Algorithm 3. The termination conditions are modified to handle the other constructs introduce in the extended algorithm. We will not pursue on the $SHIQ$ tableaux algorithm. You can find an expressive description of the algorithm in section 5.3.4 [?].
Acknowledgement

The slides for this course have been prepared by Saminda Abeyruwan.
Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph.

*Foundations of Semantic Web Technologies.*