Understanding: DL and Automated Reasoning with OWL

Semantic Web (CSC751)

Ubbo Visser

Department of Computer Science
University of Miami

October 24, 2023

UNIVERSITY
OF MIAMI

Outline

1. Description logic
2. Model-theoretic semantics of OWL
3. Automated reasoning with OWL

$$Unicornio(UnicornioBelleza)$$
$$Unicornio \sqsubseteq Ficticio$$
$$Unicornio \sqsubseteq Animal$$

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.

$\mathcal{ALC}$

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

$\mathcal{ALC}$

- Basic building blocks of $\mathcal{ALC}$:
  - 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 $\{\forall R.C$, **owl:allValuesFrom**$\}$ and $\{\exists R.C$, **owl:someValuesFrom**$\}$ are class expressions. E.g., $Exam \sqsubseteq \forall hasExaminer.Professor$ $\rightsquigarrow$ all examiners of an exam must be professors, and $Exam \sqsubseteq \exists hasExaminer.Professor$ $\rightsquigarrow$ must have at least one examiner who is a professor.
  - Quantifiers can be nested arbitrarily.
- $\bot \equiv$ **owl:Nothing**; $\bot \equiv C \sqcap \neg C$ for some arbitrary class **C**.
- $\top \equiv$ **owl:Thing**; $\top \equiv C \sqcup \neg C$ for some arbitrary class **C**.
- $\top \equiv \neg\bot$.
- **owl:disjointWith**; $C \sqcap D \sqsubseteq \bot \equiv C \sqsubseteq \neg D$ for two classes **C** and **D**.
- **rdfs:range**; $\top \sqsubseteq \forall R.C$ states that **C** is the range of role **R**, and
- **rdfs:domain**; $\exists R.\top \sqsubseteq C$ states that **C** is the domain of role **R**.

$\mathcal{ALC}$

- 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**,
  $C, D ::= A \mid \top \mid \bot \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$.
- Terminological knowledge (T-Box) axioms (formula):
  - **Contains statements of the form** $C \equiv D$ or $C \sqsubseteq D$, where C and D are class expressions.
  - Axioms of the form $C \sqsubseteq D$ 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 $R(a, b)$
- $\mathcal{ALC}$ **KB** $\equiv$ $\mathcal{ALC}$ **T-Box** plus $\mathcal{ALC}$ **A-Box**.

$\mathcal{ALC}$ to $\mathcal{SHOIN}(\mathcal{D})$

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

$\mathcal{SHOIN}(\mathcal{D})$

- **owl:oneOf**: this represents closed classes (a.k.a. union of nominals) that contains exactly $\{a_1, \ldots, a_n\} \equiv \{a_1\} \sqcap \ldots \sqcap \{a_n\} \sqsubseteq \bot$ individuals.
- **owl:minCardinality, owl:maxCardinality**, and **owl:cardinality**: $\geq nR$, $\leq nR$, and $= nR$. These are part of **unqualified cardinality restrictions**.
- **Individual relationships** for equivalence $\{a\} \equiv \{b\}$, and disjointness $\{a\} \sqcap \{b\} \sqsubseteq \bot$.
- **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**: $\mathrm{Tra}(R)$, and **symmetry**: R as $R \equiv R^-$.
- **Functionality**: $\top \sqsubseteq \leq 1R$, and **inverse functionality**: $\top \sqsubseteq \leq 1R^-$.
- **Datatypes**.
- Role functionality and inverse functionality are implemented using cardinality restrictions. Thus, $\mathcal{SHOINF}(\mathcal{D})$ is implicit in $\mathcal{SHOIN}(\mathcal{D})$.

$\mathcal{SHOIN}(\mathcal{D})$ to $\mathcal{SHOIQ}(\mathcal{D})$

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

$\mathcal{SHOIQ}(\mathcal{D})$ to $\mathcal{SROIQ}(\mathcal{D})$

- **Generalized role inclusion**: $R_1 \circ \ldots \circ R_n \sqsubseteq R$ 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: $\mathcal{SHOIN}(\mathcal{D})$.
- OWL 1 Lite: $\mathcal{SHIF}(\mathcal{D})$.
- OWL 2 Full: is not a description logic.
- OWL 2 DL: $\mathcal{SROIQ}(\mathcal{D})$.
- OWL 2 EL: $\mathcal{EL}^{++}$.
- OWL 2 QL: DL-Lite.

$\mathcal{SROIQ}(\mathcal{D})$

- $\mathcal{SROIQ}(\mathcal{D})$ 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.

$\mathcal{SROIQ}(\mathcal{D})$

- **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 **generalized role hierarchy**.
- **Generalized role hierarchy** is **regular** if **there exists** a **strict partial order** $\prec$ $(\forall x, y \in X : x < y$ iff $x \leq y$ and $x \neq y)$ on R such that:
  - $S \prec R$ iff $S^- \prec R$.
  - **Every** role inclusion axioms is one of the forms:
    $R \circ R \sqsubseteq R,\quad R^- \sqsubseteq R,\quad S_1 \circ \ldots S_n \sqsubseteq R,$
    $R \circ S_1 \circ \ldots S_n \sqsubseteq R,\quad S_1 \circ \ldots S_n \circ R \sqsubseteq R$
    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 $\mathcal{SROIQ}(\mathcal{D})$.
    e.g.,
    $hasParent \circ hasHusband \sqsubseteq hasFather$, and $hasFather \sqsubseteq hasParent$ enforces $hasParent \prec hasFather$ and $hasFather \prec hasParent$, and the role hierarchy is **not regular**, because $S \prec R$ must be strict.

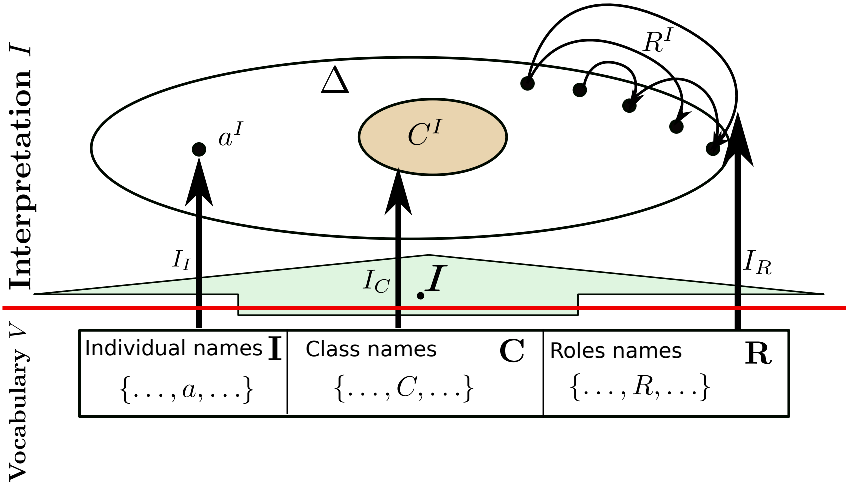
$\mathcal{SROIQ}(\mathcal{D})$

- 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., $\{R \sqsubseteq R_1; \quad R_1 \circ R_2 \sqsubseteq R_3; \quad R_3 \sqsubseteq R_4\}$ then, the simple roles of the role hierarchy is $\{R, R^-, R_1, R_1^-, R_2, R_2^-\}$.
- $\mathcal{SROIQ}(\mathcal{D})$ expresses $\{Tran(R), R \circ R \sqsubseteq R\}$, $\{Sym(R), R^- \sqsubseteq R\}$, $Asy(R)$, $Ref(R)$, and $Dis(S, R)$. These axioms are decidable iff they include simple roles (a.k.a. role characteristics).
- Therefore, $\mathcal{SROIQ}(\mathcal{D})$ R-Box is the union of role characteristics and a role hierarchy, and it is regular if its role hierarchy is regular.

$\mathcal{SROIQ}(\mathcal{D})$ 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, $R, S \in \mathbf{R}$ 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 $C \equiv D$, 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 $C \in \mathbf{C}$, $R \in \mathbf{R}$ and a and b are individuals.
- $\mathcal{SROIQ}(\mathcal{D})$ **KB** $\equiv$ union or regular $\mathcal{SROIQ}(\mathcal{D})$ **R-Box R** $\mathcal{SROIQ}(\mathcal{D})$ **T-Box** and $\mathcal{SROIQ}(\mathcal{D})$ **A-Box** for **R**.

Model-theoretic semantics of OWL



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** Δ with a set of entities (resources, individuals or single objects).
- Then we provide an **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 pairs of elements (**the property extension**).
- Δ is arbitrary and the implementation of functions I_I, I_C, and I_R has a lot of freedom.

E.g., (1)

$$
\begin{aligned}
Professor &\sqsubseteq FacultyMember \\
&Professor(ubboVisser) \\
&hasAffiliation(ubboVisser, uofm)
\end{aligned}
$$

Let,

$$
\begin{aligned}
\Delta &= \{\clubsuit, \spadesuit, \heartsuit\} \\
I_I(ubboVisser) &= \heartsuit \\
I_I(uofm) &= \spadesuit \\
I_C(Professor) &= \{\clubsuit\} \\
I_C(FacultyMember) &= \{\clubsuit, \spadesuit\} \\
I_R(hasAffiliation) &= \{(\clubsuit, \spadesuit), (\spadesuit, \heartsuit)\}
\end{aligned}
$$

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 Δ. In logic, we abstract from these symbols. i.e., we can rename things in Δ without compromising logical meaning.
  - Whether the interpretation faithfully captures the relations between entities as stated in the knowledge base. $I_I(ubboVisser) \notin I_C(Professor)$, and $(I_I(ubboVisser), I_I(uofm)) \notin I_R(hasAffiliation)$ 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 $\mathcal{SROIQ}$ knowledge base consists of a domain Δ and an interpretation function $.^{\mathcal{I}}$ which satisfy the constraints given in the next slide.
- There are many degrees of freedom for choosing Δ, I_I, I_C, and I_R. As we have shown in the above example, the interpretations may not intuitively meaningful.

$\top^{\mathcal{I}} = \Delta$ and $\bot^{\mathcal{I}} = \emptyset$	-
$(\neg C)^{\mathcal{I}} = \Delta \backslash C^{\mathcal{I}}$	$\neg C$ describes things which are not in C
$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$	$C \sqcap D$ describes things which are both in C and in D
$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$	$C \sqcap D$ describes things which are both in C or in D
$(\exists R.C)^{\mathcal{I}} = \{x \mid$ there is some y with $(x,y) \in R^{\mathcal{I}} \cap y \in C^{\mathcal{I}}\}$	$\exists R.C$ describes those things which are connected via R with something in C
$(\forall R.C)^{\mathcal{I}} = \{x \mid$ for all y with $(x,y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$	$\forall R.C$ describes those things x for which every y which connects from x via role R is in the class C
$(\leq nR.C)^{\mathcal{I}} = \{x \mid \#\{(x,y) \in R^{\mathcal{I}} \mid y \in C^{\mathcal{I}}\} \leq n\}$	$\leq nR.C$ describes those things which are connected via R to at most n things in C
$(\geq nR.C)^{\mathcal{I}} = \{x \mid \#\{(x,y) \in R^{\mathcal{I}} \mid y \in C^{\mathcal{I}}\} \geq n\}$	$\geq nR.C$ describes those things which are connected via R to at least n things in C
$\{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}$	$\{a\}$ describes the class containing only a
$(\exists S.Self)^{\mathcal{I}} = \{x \mid (x,x) \in S^{\mathcal{I}}\}$	$\exists S.Self$ describes those things which are connected to themselves via S
$(R^{-})^{\mathcal{I}} = \{(b,a) \mid (a,b) \in R^{\mathcal{I}}\}$	for all $R \in \mathbf{R}$
$U^{\mathcal{I}} = \Delta \times \Delta$	for the universal role U

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 $\mathcal{SROIQ}$ knowledge base K is a **model** of K, $\mathcal{I} \models \mathcal{K}$, 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 $Ref(R) \in K$, then $\{(x, x) \mid x \in \Delta\} \in R^{\mathcal{I}}$.
- If $Asy(R) \in K$, then $(x, y) \notin R^{\mathcal{I}}$ whenever $(y, x) \in R^{\mathcal{I}}$.
- If $Dis(R, S) \in K$, 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 $(ubboVisser^{\mathcal{I}}, uofm^{\mathcal{I}}) \in hasAffiliation^{\mathcal{I}}$, i.e., $I_R(hasAffiliation) = \{(\clubsuit, \spadesuit), (\spadesuit, \heartsuit), (\heartsuit, \spadesuit)\}$.
- Another model for e.g., (1):

$$
\begin{aligned}
\Delta &= \{\alpha, \beta, \gamma\} \\
I_I(ubboVisser) &= \beta \\
I_I(uofm) &= \alpha \\
I_C(Professor) &= \{\beta\} \\
I_C(FacultyMember) &= \{\beta, \gamma\} \\
I_R(hasAffiliation) &= \{(\beta, \alpha)\}
\end{aligned}
$$

- How many models exists for a knowledge base?

Logical consequence

$\cdot^{\mathcal{I}}$	$Model_1$	$Model_2$	$Model_3$
Δ	$\{\alpha, \beta, \gamma\}$	$\{1, 2\}$	$\{\clubsuit\}$
$I_I(ubboVisser)$	β	1	$\clubsuit$
$I_I(uofm)$	α	2	$\clubsuit$
$I_C(Professor)$	$\{\beta\}$	$\{1\}$	$\{\clubsuit\}$
$I_C(FacultyMember)$	$\{\alpha, \beta, \gamma\}$	$\{1, 2\}$	$\{\clubsuit\}$
$I_R(hasAffiliation)$	$\{(\beta, \alpha)\}$	$\{(1, 1), (1, 2)\}$	$\{(\clubsuit, \clubsuit)\}$

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

$K \models C \sqsubseteq D$	iff $(C \sqsubseteq D)^{\mathcal{I}}$ for all $I \models K$	iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all $I \models K$
$K \models C(a)$	iff $(C(a))^{\mathcal{I}}$ for all $I \models K$	iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all $I \models K$
$K \models R(a,b)$	iff $(R(a,b))^{\mathcal{I}}$ for all $I \models K$	iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ for all $I \models K$
$K \models \neg R(a,b)$	iff $(\neg R(a,b))^{\mathcal{I}}$ for all $I \models K$	iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin R^{\mathcal{I}}$ for all $I \models K$

E.g., (2)

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

$$
\begin{aligned}
\Delta &= \{\spadesuit, \clubsuit\} \\
I_I(\mathit{ubboVisser}) &= \clubsuit \\
I_I(\mathit{uofm}) &= \spadesuit \\
I_C(\mathit{Professor}) &= \{\clubsuit\} \\
I_C(\mathit{FacultyMember}) &= \{\clubsuit\} \\
I_R(\mathit{hasAffiliation}) &= \{(\clubsuit, \spadesuit)\}
\end{aligned}
$$

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.,

$$\begin{aligned} & \mathit{Unicorn}(\mathit{beauty}) && (1) \\ \mathit{Unicorn} & \sqsubseteq \mathit{Fictitious} && (2) \\ \mathit{Unicorn} & \sqsubseteq \mathit{Animal} && (3) \\ \mathit{Fictitious} \sqcap \mathit{Animal} & \sqsubseteq \bot && (4) \end{aligned}$$

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.

$\mathcal{SROIQ}$ semantics via first-order predicate logic

- Every $\mathcal{SROIQ}$ knowledge base translates a theory in first-order predicate logic with equality.
- $\pi(K) = \cup_{\alpha \in K} \pi(\alpha)$. $\pi(\alpha)$ definition depends on the type of the axiom α.
- If α is an individual assignment, then $\pi(\alpha)$ is defined as:

$$\begin{aligned} \pi(C(a)) &= C(a) \\ \pi(R(a,b)) &= R(a,b) \\ \pi(\neg R(a,b)) &= \neg R(a,b) \end{aligned}$$

$\mathcal{SROIQ}$ semantics via first-order predicate logic

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

$$\pi(R_1 \sqsubseteq R_2) = \forall x, y(\pi_{x,y}(R_1) \rightarrow \pi_{x,y}(R_2))$$
$$\pi_{x,y}(S) = S(x, y)$$
$$\pi_{x,y}(R^-) = \pi_{y,x}(R)$$
$$\pi_{x,y}(R_1 \circ \ldots \circ R_n) = \exists x_1, \ldots, x_n \left(\pi_{x,x_1}(R_1) \wedge \bigwedge_{i=1}^{n-2} \pi_{x_i,x_{i+1}}(R_{i+1}) \wedge \pi_{n-1,y}(R_n) \right)$$
$$\pi(Ref(R)) = \forall x \pi_{x,x}(R)$$
$$\pi(Asy(R)) = \forall x, y(\pi_{x,y}(R) \rightarrow \neg\pi_{y,x}(R))$$
$$\pi(Dis(R_1, R_2)) = \neg(\exists x, y)(\pi_{x,y}(R_1) \wedge \pi_{x,y}(R_2))$$

$\mathcal{SROIQ}$ semantics via first-order predicate logic

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

$$
\begin{aligned}
\pi(C \sqsubseteq D) &= \forall x(\pi_x(C) \rightarrow \pi_x(D)) \\
\pi_x(A) &= A(x) \\
\pi_x(\neg C) &= \neg \pi_x(C) \\
\pi_x(C \sqcap D) &= \pi_x(C) \wedge \pi_x(D) \\
\pi_x(C \sqcup D) &= \pi_x(C) \vee \pi_x(D) \\
\pi_x(\forall R.C) &= \forall x_1(R(x, x_1) \rightarrow \pi_{x_1}(C)) \\
\pi_x(\exists R.C) &= \exists x_1(R(x, x_1) \wedge \pi_{x_1}(C)) \\
\pi_x(\geq nS.C) &= \exists x_1, \ldots, x_n \left(\bigwedge_{i \neq j} (x_i \neq x_j) \wedge \bigwedge_i (S(x, x_i) \wedge \pi_{x_i}(C)) \right) \\
\pi_x(\leq nS.C) &= \neg(\exists x_1, \ldots, x_{n+1}) \left(\bigwedge_{i \neq j} (x_i \neq x_j) \wedge \bigwedge_i (S(x, x_i) \wedge \pi_{x_i}(C)) \right) \\
\pi_x(\{a\}) &= (x = a) \\
\pi_x(\exists S.Self) &= S(x, x)
\end{aligned}
$$

E.g., (4)

$$\begin{aligned}
Professor &\sqsubseteq FacultyMember \\
\forall x(Professor(x) &\rightarrow FacultyMember(x)) \\
Professor &\sqsubseteq (Person \sqcap FacultyMember) \sqcup (Person \sqcap \neg PhdStudent) \\
\forall x(Professor(x) &\rightarrow ((Person(x) \wedge FacultyMember(x)) \vee \\
&\quad (Person(x) \wedge \neg PhdStudent(x))) \\
Exam &\sqsubseteq \forall hasExaminer.Professor \\
\forall x(Exam(x) &\rightarrow \forall y(hasExaminer(x, y) \rightarrow Professor(y))) \\
Exam &\sqsubseteq \leq 2 hasExaminer \\
\forall x(Exam(x) &\rightarrow \neg(\exists x_1, x_2, x_3)((x_1 \neq x_2) \wedge (x_2 \neq x_3) \wedge (x_1 \neq x_3) \\
&\quad hasExaminer(x, x_1) \wedge hasExaminer(x, x_2) \wedge \\
&\quad hasExaminer(x, x_3)))
\end{aligned}$$

$$Professor(ubboVisser) \qquad Professor(ubboVisser)$$

$$hasAffiliation(ubboVisser, uofm) \qquad hasAffiliation(ubboVisser, uofm)$$

$$\begin{aligned}
hasParent \circ hasBrother &\sqsubseteq hasUncle \\
\forall x, y(\exists x_1(hasParent(x, x_1) \wedge hasBrother(x_1, y)) &\rightarrow hasUncle(x, y))
\end{aligned}$$

Automated reasoning with OWL

Tableaux algorithms

- Formal semantics provides implicit knowledge via logical consequence.
- α is a logical consequence of K, $K \models \alpha$, if and only if every model of K is a model of α.
- 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, HermiT, RacerPro, Konclude, 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 representing the same class?*	$C \equiv D$?
Class disjointness. *Are there common members?*	$C \sqcap D \sqsubseteq \bot$?
Global consistency of a knowledge base. *Is the knowledge base meaningful?*	$K \models false$?
Class consistency. *Is C empty?*	$C \sqsubseteq \bot$?
Instance checking. *Is a contained in C?*	$C(a)$?
Instance retrieval. *Find all known individuals belonging to a given class.*	$\forall x C(x)$?

Inference problem

- Using a 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

Subsumption	$K \models C \sqsubseteq D$ iff $K \cup \{(C \sqcap \neg D)(a)\}$ is unsatisfiable, where a is a new individual not occurring in K.
Class equivalence	$K \models C \equiv D$ iff $K \models C \sqsubseteq D$ and $K \models D \sqsubseteq C$.
Class disjointness	$K \models C \sqcap D \sqsubseteq \bot$ iff $K \cup \{(C \sqcap D)(a)\}$ is unsatisfiable, where a is a new individual not occurring in K.
Global consistency	K is globally consistent if it has a model.
Class consistency	$K \models C \sqsubseteq \bot$ iff $K \cup \{C(a)\}$ is unsatisfiable, where a is a new individual not occurring in K.
Instance checking	$K \models C(a)$ iff $K \cup \{\neg C(a)\}$ is unsatisfiable.
Instance retrieval	To find all individuals belonging to a class C, we have to check for all individuals a whether $K \models C(a)$.

Reduction to satisfiability

- A 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 the 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 the 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 (cf. next slide) exhaustively.
- $NNF(K)$ moves all the negation symbols down into subformulae until they occur directly in front of class names.
- $NNF(K)$ only transforms the T-Box.
- $NNF(K) = \mathcal{A} \cup \mathcal{R} \cup \bigcup_{C \sqsubseteq D \in K} NNF(C \sqsubseteq D)$, where $\mathcal{A}$ and $\mathcal{R}$ are the A-Box and the R-Box of K.
- K and $NNF(K)$ are logically equivalent, i.e., they have identical models.

$$
\begin{aligned}
NNF(C \sqsubseteq D) &= NNF(\neg C \sqcup D) && (5)\\
NNF(C) &= C \quad \text{if } C \text{ is a class name} && (6)\\
NNF(\neg C) &= \neg C \quad \text{if } C \text{ is a class name} && (7)\\
NNF(\neg\neg C) &= NNF(C) && (8)\\
NNF(C \sqcup D) &= NNF(C) \sqcup NNF(D) && (9)\\
NNF(C \sqcap D) &= NNF(C) \sqcap NNF(D) && (10)\\
NNF(\neg(C \sqcup D)) &= NNF(\neg C) \sqcap NNF(\neg D) && (11)\\
NNF(\neg(C \sqcap D)) &= NNF(\neg C) \sqcup NNF(\neg D) && (12)\\
NNF(\forall R.C) &= \forall R.NNF(C) && (13)\\
NNF(\exists R.C) &= \exists R.NNF(C) && (14)\\
NNF(\neg\forall R.C) &= \exists R.NNF(\neg C) && (15)\\
NNF(\neg\exists R.C) &= \forall R.NNF(\neg C) && (16)\\
NNF(\leq nR.C) &= \leq nR.NNF(C) && (17)\\
NNF(\geq nR.C) &= \geq nR.NNF(C) && (18)\\
NNF(\neg \leq nR.C) &= \geq (n+1)R.NNF(C) && (19)\\
NNF(\neg \geq (n+1)R.NNF(C)) &= \leq nR.NNF(C) && (20)\\
NNF(\neg \geq 0.R.C) &= \bot && (21)
\end{aligned}
$$

E.g.,

$$P \sqsubseteq (E \sqcap U) \sqcup \neg(\neg E \sqcup D)$$

Lets transform this formula to NNF

$$\neg P \sqcup (E \sqcap U) \sqcup \neg(\neg E \sqcup D)$$

$$\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D) \quad \square$$

Naíve 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". $\forall x \ \top \in \mathcal{L}(x)$, 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 consequences of a knowledge base.

Illustration

- Assume that the knowledge base is transformed to NNF.

$$K \models C(a) \tag{22}$$
$$K \models (\neg C \sqcap D)(a) \tag{23}$$
$$(\neg C \sqcap D)(a) \models \neg C(a) \tag{24}$$

- 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,

$$\begin{aligned} K &\models C(a) \\ K &\models \neg C \sqcup D \\ K &\models \neg D(a) \end{aligned}$$

- 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

$$K \models C(a) \quad (25)$$

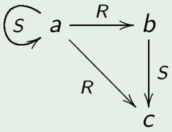
$$K \models \neg C \sqcup D \quad (26)$$

$$K \models \neg D(a) \quad (27)$$

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

Illustration: Roles

$$
\begin{aligned}
K &\models R(a, b)\\
K &\models S(a, a)\\
K &\models R(a, c)\\
K &\models S(b, c)
\end{aligned}
$$



$$K \models \exists R.\exists S.C(a)$$

$$a \xrightarrow{R} x \xrightarrow{S} y$$

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 naíve 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($NNF(K)$)

Data: $NNF(K)$
Result: Satisfiability status of K
$initialTableaux$ = INITIALIZE_Tableaux($NNF(K)$);
return APPLY_RULES($initialTableaux$, $NNF(K)$);

Algorithm

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 $\mathcal{L}(a) = \emptyset$.
- For all pairs a, b of individuals, set $\mathcal{L}(a, b) = \emptyset$.
- 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 $L(a, b) \leftarrow R$.

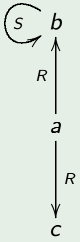
Algorithm 3: APPLY_RULES(*initialTableaux*, $NNF(K)$)

- **In each step**, nondeterministically apply the following rules:
  - $\sqcap$-**rule**: If $C \sqcap D \in \mathcal{L}(x)$ and $\{C, D\} \not\subseteq \mathcal{L}(x)$, then set $\mathcal{L}(x) \leftarrow \{C, D\}$.
  - $\sqcup$-**rule**: If $C \sqcup D \in \mathcal{L}(x)$ 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 $C \notin \mathcal{L}(y)$, then set $\mathcal{L}(y) \leftarrow C$.
  - **T-Box-rule**: If C is a T-Box statement and $C \notin \mathcal{L}(x)$, then set $\mathcal{L}(x) \leftarrow C$.
- **Terminates**,
  - 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 non-determinism:
  - Which expansion rule to apply next: whatever rule we choose, it will **not** get us onto the wrong track, though the algorithm may take more steps to terminate. This leads to **don't care non-determinism**.
  - The choice which has to be made when applying the $\sqcup$-rule: bad choice gets us on to the wrong track. This is because, if we choose to set $\mathcal{L}(x) \leftarrow C$, 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 non-determinism**.
- 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

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

Algorithm

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

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

$\downarrow R$

x — $\mathcal{L}(x) = \{D, \neg D \sqcup E, \underbrace{E, \neg E}_{\text{contradiction}}\}$

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

Algorithm

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

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

$\downarrow R$

x $\quad \mathcal{L}(x) = \{D, \neg D \sqcup E, \underbrace{E, \neg E}_{\text{contradiction}}\}$

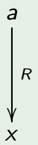
Tableaux example

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

Algorithm

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



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

$$\mathcal{L}(x) = \{D, \neg D \sqcup E, \underbrace{E, \neg E}_{\text{contradiction}}\}$$

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.
- $NNF(\neg(\exists R.E)(a)) = \forall R.\neg E(a)$
- $NNF(K) = \{C(a), \neg C \sqcup \exists R.D, \neg D \sqcup E \sqcup F, \neg F \sqcup E, \forall R.\neg E(a)\}$

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.
- $\mathcal{L}(a) \leftarrow \exists R.D$

Algorithm

- $\mathcal{L}(x) \leftarrow \neg E$($\forall$-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 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.
- $NNF(\neg(\exists R.E)(a)) = \forall R.\neg E(a)$
- $NNF(K) = \{C(a), \neg C \sqcup \exists R.D, \neg D \sqcup E \sqcup F, \neg F \sqcup E, \forall R.\neg E(a)\}$

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.
- $\mathcal{L}(a) \leftarrow \exists R.D$

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

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

$\downarrow R$

x $\quad \mathcal{L}(x) = \{D, \neg E, \cancel{\neg D} \sqcup \cancel{E} \sqcup F, \cancel{\neg F} \sqcup \cancel{E}\}$

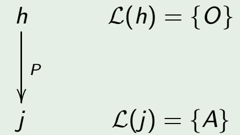
Tableaux example

$$Human \sqsubseteq \exists hasParent.Human$$
$$Orphan \sqsubseteq Human \sqcap \forall hasParent.\neg Alive$$
$$Orphan(harryPotter)$$
$$hasParent(harryPotter, jamesPotter)$$

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

$$\neg H \sqcup \exists P.H$$
$$\neg O \sqcup (H \sqcap \forall P.\neg A)$$
$$O(h)$$
$$P(h, j)$$
$$A(j)$$

Algorithm



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

Tableaux

$$h \xrightarrow{P} j$$

$$\mathcal{L}(h) = \{O, \neg O \sqcup (H \sqcap \forall P.\neg A), H \sqcap \forall P.\neg A, H, \forall P.\neg A\}$$

$$\mathcal{L}(j) = \{A, \neg A\}$$

Tableaux example

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

- From Algorithm 2,

  $a \qquad \mathcal{L}(a) = \{C, \forall R.\neg E\}$
- 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: $a \qquad \mathcal{L}(a) = \{C, \forall R.\neg E, \neg C \sqcup \exists R.D, \exists R.D\}$

    $\downarrow R$

    $x \qquad \mathcal{L}(x) = \{D\}$

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.
  - $\mathcal{L}(x) \leftarrow E$

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

$$\downarrow R$$

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

  - $\forall R.\neg E \in \mathcal{L}(a)$, 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., $K \models (\exists R.E)(a)$.

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

$$\downarrow R$$

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

The tableaux algorithm with blocking for $\mathcal{ALC}$

- Algorithm 1 for $\mathcal{ALC}$ does not always terminate.
- Consider: $K = \{\exists R.\top, \top(a_1)\}$.
  - 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 $\mathcal{L}(a_1) = \{\top\}$.
  - T-Box-rule: $\mathcal{L}(a_1) \leftarrow \exists R.T$.
  - $\exists$-rule: creates a new node x with $\mathcal{L}(a_1, x) = \{R\}$ 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.

$$a_{1\,\mathcal{L}(a_1)=\{\top,\exists R.T\}} \xrightarrow{R} x_{\mathcal{L}(x)=\{\top,\exists R.T\}} \xrightarrow{R} y_{\mathcal{L}(y)=\{\top,\exists R.T\}} \xrightarrow{R} \ldots$$

Blocking continue

- We said that $\mathcal{ALC}$ or $\mathcal{SROIQ}$ 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
  - $\mathcal{L}(x) \subseteq \mathcal{L}(y)$.

Blocking continue

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

  $$a_{1\,\mathcal{L}(a_1)=\{\top,\exists R.T\}} \xrightarrow{R} x_{\mathcal{L}(x)=\{\top\}}$$
- 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 would be cyclic.

Blocking example

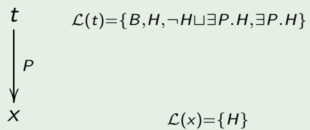
$$K = \{H \sqsubseteq \exists P.H, B(t)\}$$

Which stands for: $Human \sqsubseteq \exists hasParent.Human$
$Bird(tweety)$

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

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

Initialized	$\mathcal{L}(t) = \{B, H\}$
T-Box-rule	$\mathcal{L}(t) \leftarrow \neg H \sqcup \exists P.H$
$\sqcup$-rule	$\mathcal{L}(t) \leftarrow \neg H$ (contradiction)
	$\mathcal{L}(t) \leftarrow \exists P.H$
$\exists$-rule	create a node with label x, $\mathcal{L}(t, x) = \{P\}$, and $\mathcal{L}(x) = \{H\}$
	node x is blocked by t

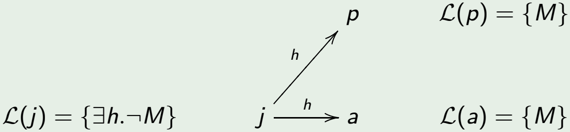


Open world assumption

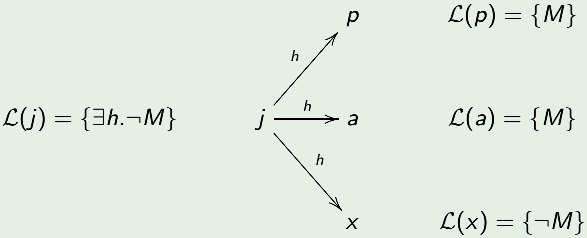
- Let
$$K = \{h(j,p), h(j,a), M(p), M(a)\}$$
which stands for
$$hasChild(john, peter), hasChild(john, alex), Male(peter), Male(alex)$$
- $K \not\models \forall hasChild.Male(john)$ (*not* a logical consequence of the knowledge base) $(K \not\models \forall\ x\ hasChild(x, john) \rightarrow Male(john))$.
- Add the negation of the statement $\neg\forall h.M(j)$ to K, and show that $NNF(K')$ is satisfiable.
- OWA for K' satisfiability:
    - There is no information whether or not *john* has only *peter* and *alex* as children.
    - There may be that *john* has additional children who are not listed in K'.
    - Therefore, it is not possible to infer that all *john*'s children are *Male*.

Illustration

- $NNF(K') = \{h(j,p), h(j,a), M(p), M(a), \exists h.\neg M(j)\}$.
- Algorithm 2 yields:



- Algorithm 3 yields: $\exists$-rule $\mathcal{L}(j,x) = \{h\}$ and $\mathcal{L}(x) = \{\neg M\}$.



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.
- The 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 $\neg C \sqcup \forall S.A$.
  - $\forall$-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 $\neg A \sqcup \exists R.\exists S.A$.
  - $\sqcup$-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

Description logic	Combined complexity	Data complexity
$\mathcal{ALC}$	ExpTime-complete	NP-complete
$\mathcal{SHIQ}$	ExpTime-complete	NP-complete
$\mathcal{SHOIN}(\mathcal{D})$	NExpTime-complete	NP-hard
$\mathcal{SROIQ}(\mathcal{D})$	N2ExpTime-complete	NP-hard
$\mathcal{EL}^{++}$	P-complete	P-complete
$\mathcal{DLP}$	P-complete	P-complete
$\mathcal{DL}$-Lite	P	LOGSPACE

- 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**.

$\mathcal{SHIQ}$

- The $\mathcal{ALC}$ full tableau algorithm has been extended to $\mathcal{SHIQ}$ adding two more constraints to Algorithm 2 and few more rules to Algorithm 3.
- The termination conditions are modified to handle the other constructs introduced in the extended algorithm.
- We will not pursue on the $\mathcal{SHIQ}$ tableaux algorithm. You can find an expressive description of the algorithm in section 5.3.4 [HKR09].

Acknowledgement

Acknowledgement

The slides for this course have been prepared by Saminda Abeyruwan.

Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph.
Foundations of Semantic Web Technologies.
Chapman & Hall/CRC, 2009.