WIMS'11 Tutorial

Sogndal, May 2011

Description Logics Reasoning for the Semantic Web

Anni-Yasmin Turhan

TU Dresden, Germany

TU
Dresden

Semantic Web

Idea:

- resources such as data, services etc. are labelled with annotations
- annotations describe properties of the resource
- annotations: machine processable

Ontologies for annotations:

- agreed vocabulary
- descriptions of categories relevant to the application

W3C:
Web ontology language (OWL) recommended as the ontology language for the semantic web

Reasoning for the Semantic Web

- Intelligent access to resources
  :use information that follows from the descriptions
- Extract implicitly captured information by reasoning
  – automatically!
- Requires formal semantics of ontology formalism

Description Logics (DLs)

- are a family of different logics with varying expressivity
- knowledge representation formalism with formal semantics
- formalism for declarative description of facts
- come with powerful reasoning services:
  make implicit knowledge explicit
- are applied in many practical areas
- are the basis for the web ontology language OWL!

Description Logics research

- investigates reasoning algorithms of DLs
- emphasis on sound, complete and terminating reasoning algorithms
  ➵ guarantees for the reasoning service
- since late nineties:
  - efficient DL reasoning systems available for DLs
    for which reasoning is not tractable
  - investigation of very expressive (but computationally costly) DLs
- since last decade:
  - investigation of DLs that have low expressivity,
    but allow for very efficient reasoning

The OWL standard

OWL 1:

- W3C recommendation of 2004
- OWL DL and OWL Lite: DL-based ontology languages

OWL 2:

- W3C recommendation of 2009
- consists of
  - an expressive language
  - 2 profiles that correspond to light-weight DLs

Tutorial overview

1. DLs: Basic notions
   1.1. concept descriptions,
   1.2. TBox,
   1.3. ABox

   and their reasoning services
2. Reasoning in $\mathcal{ALC}$
3. OWL 2: beyond $\mathcal{ALC}$
4. OWL 2 EL
   4.1 the $\mathcal{EL}$-family
   4.2. Reasoning method for $\mathcal{EL}$
5. OWL 2 QL
   5.1. DL-Lite family
   5.2. Conjunctive queries
6. Conclusions

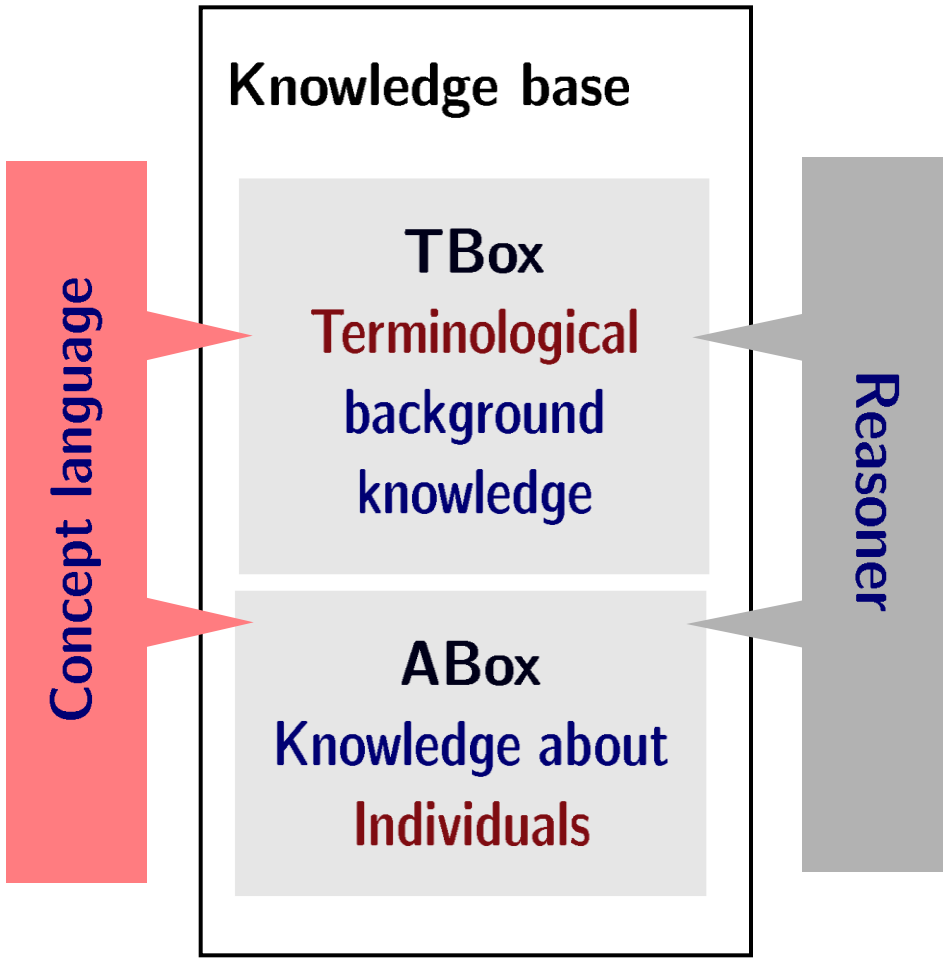
DL knowledge bases

DL knowledge bases have two parts:

1. Representation of the intensional knowledge
   - Representation of categories and relations from the domain of discourse
   - Terminological knowledge stored in the TBox

2. Representation of extensional knowledge
   - Representation of individuals and related tuples from the domain
   - Assertional knowledge stored in the ABox

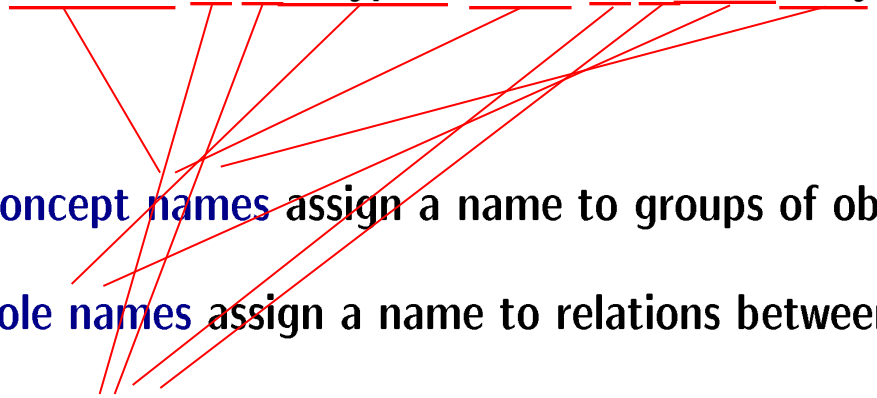
Overview DL systems



Defining Concepts with DLs

The core part of any DL is the concept language

$$\text{Mammal} \sqcap \exists \text{bodypart}.\text{Trunk} \sqcap \forall \text{color}.\text{Grey}$$



- concept names assign a name to groups of objects
- role names assign a name to relations between objects
- constructors allow to related concept names and role names

Complex concepts can be used in concept definitions:

$$\text{Elephant} \equiv \text{Mammal} \sqcap \exists \text{bodypart}.\text{Trunk} \sqcap \forall \text{color}.\text{Grey}$$

A set of such concept definitions is an ontology

The Description Logic $\mathcal{EL}$: Syntax

Atomic types: concept names $A, B, \ldots$ (unary predicates)
role names $R, S, \ldots$ (binary predicates)

Constructors: $C \sqcap D$ (conjunction)

$\exists R.C$ (existential restriction)

Special concept: $\top$ (top concept)

For example: Person $\sqcap$ $\exists$ has-child.Person

The Description Logic $\mathcal{ALC}$: Syntax

Atomic types: concept names $A, B, \ldots$ (unary predicates)

role names $R, S, \ldots$ (binary predicates)

Constructors:

$\neg C$	(negation)
$C \sqcap D$	(conjunction)
$C \sqcup D$	(disjunction)
$\exists R.C$	(existential restriction)
$\forall R.C$	(value restriction)

Special concepts:

$\top$	(top concept)
$\bot$	(bottom concept)

For example: $\neg(\ A\ \sqcup\ \exists R.(\forall S.B \sqcap \neg A))$

$\mathsf{Mammal} \sqcap \exists \mathsf{bodypart}.\mathsf{Trunk} \sqcap \forall \mathsf{color}.\mathsf{Grey}$

Example: $\mathcal{ALC}$ Concept Descriptions

Signature: $N_C = \{$ Person, Male, Happy $\}$,

$N_R = \{$has-child, has-sibling, likes, knows $\}$

Parent $\equiv$

Person $\sqcap$ $\exists$ has-child.Person

Grandparent $\equiv$

Person $\sqcap$ $\exists$ has-child.($\exists$ has-child. Person)

Uncle-of-happy-children $\equiv$

Person $\sqcap$ Male $\sqcap$ $\exists$ has-sibling.($\exists$ has-child.Person)
$\sqcap$ $\forall$ has-sibling.($\forall$ has-child.Happy)

Interpretations

Semantics based on interpretations $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where

- $\Delta^{\mathcal{I}}$ is a non-empty set (the domain)
- $\cdot^{\mathcal{I}}$ is the interpretation function mapping
  each concept name A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and
  each role name R to a binary relation $R^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$.

Semantics of Primitive Concepts

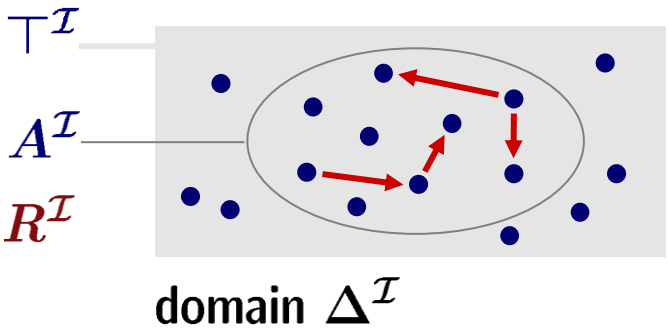
Semantics based on interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$

Concepts: Subsets of domain $\Delta^{\mathcal{I}}$

Roles: binary relations on domain $\Delta^{\mathcal{I}}$

Primitive concepts

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$$
$$\bot^{\mathcal{I}} = \emptyset$$
$$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$$



Complex $\mathcal{ALC}$-Concepts: Semantics

Semantics of complex concepts:

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

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

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

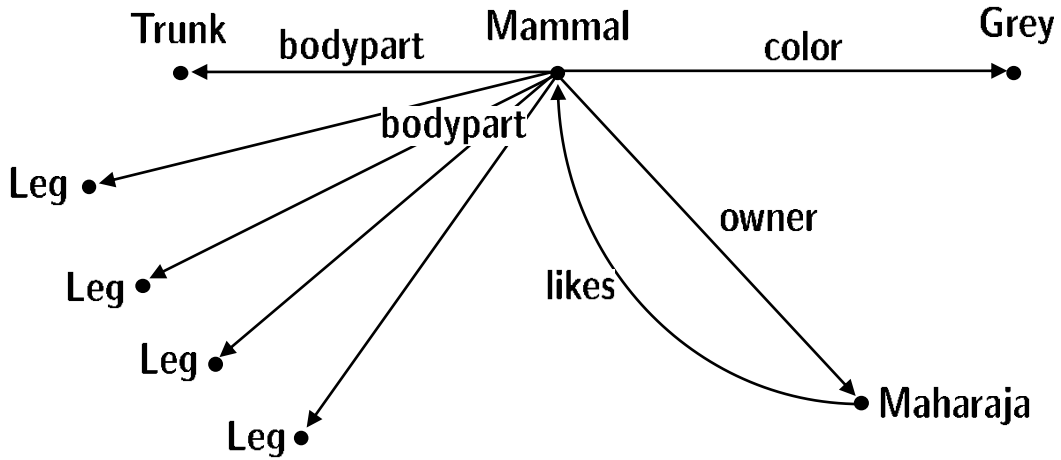
$$(\exists R.C)^{\mathcal{I}} = \{d \mid \text{there is an } e \in \Delta^{\mathcal{I}} \text{ with } (d,e) \in R^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}}\}$$

$$(\forall R.C)^{\mathcal{I}} = \{d \mid \text{for all } e \in \Delta^{\mathcal{I}}, (d,e) \in R^{\mathcal{I}} \text{ implies } e \in C^{\mathcal{I}}\}$$

Example: semantics of complex concepts

$$\mathsf{Mammal} \sqcap \exists \mathsf{bodypart}.\mathsf{Trunk} \sqcap \forall \mathsf{color}.\mathsf{Grey}$$

$$\mathsf{Mammal} \sqcap \forall \mathsf{bodypart}.(\mathsf{Trunk} \sqcup \mathsf{Leg})$$



Reasoning tasks for concepts

1. Concept satisfiability — does there exist a model of C?

   model of C: interpretation $\mathcal{I}$ with $C^{\mathcal{I}} \neq \emptyset$

   If unsatisfiable, the concept contains a contradiction.

2. Concept subsumption — does $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ hold for all $\mathcal{I}$?

   written $C \sqsubseteq D$

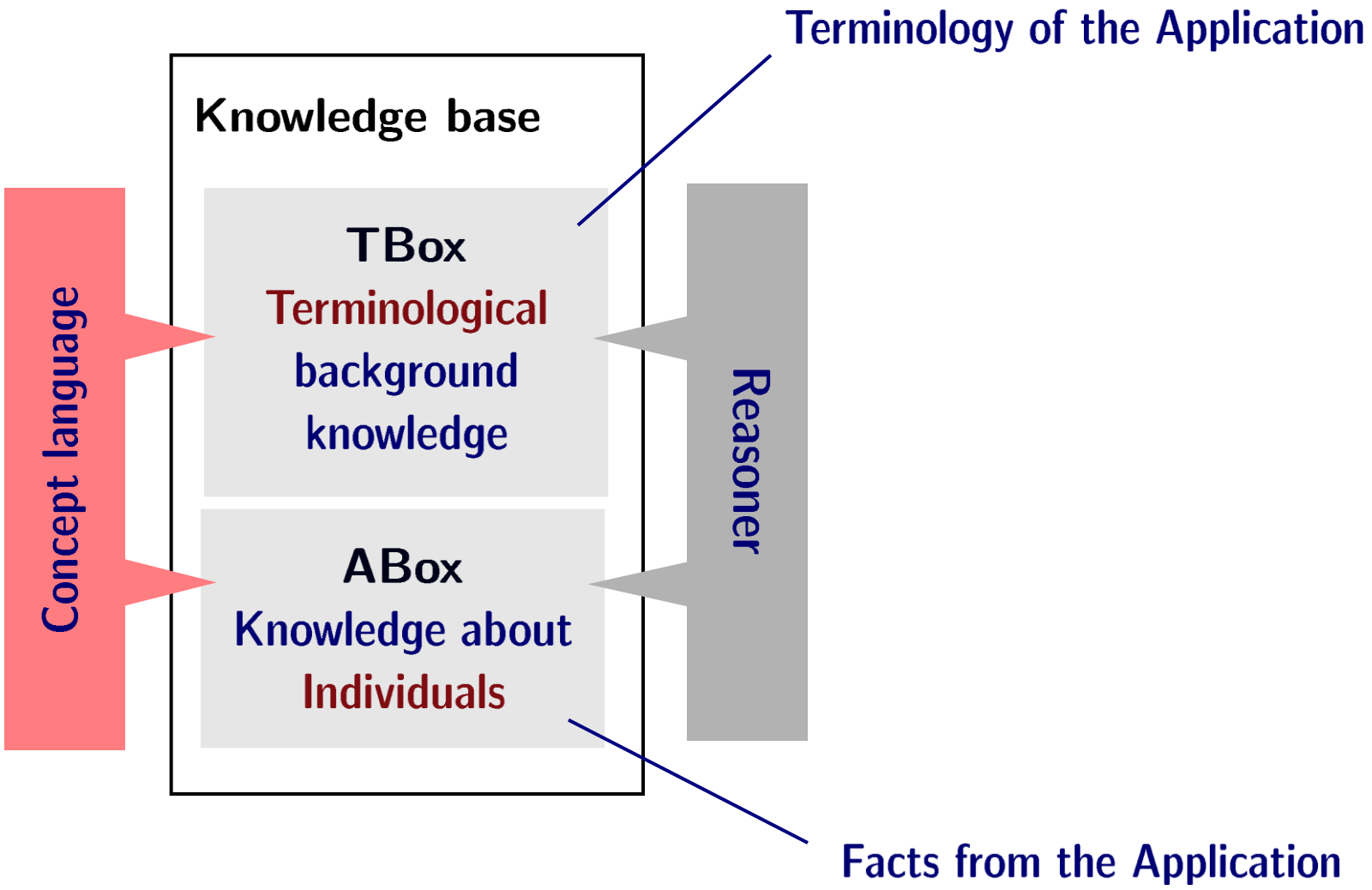
   If $C \sqsubseteq D$, then D is more general than C

Examples

- $\forall \text{owner}.\text{Rich} \sqcap \forall \text{owner}.\text{Famous} \sqsubseteq \forall \text{owner}.(\text{Rich} \sqcap \text{Famous})$
- Set $\top := A \sqcup \neg A$. Then $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ for all $\mathcal{I}$.
  Thus $C \sqsubseteq \top$ for all C.
- Set $\bot := A \sqcap \neg A$. Then $\bot^{\mathcal{I}} = \emptyset$ for all $\mathcal{I}$.
  Thus $\bot \sqsubseteq C$ for all C.
- $C \sqsubseteq D$ if and only if $C \sqcap \neg D$ is not satisfiable
- C is satisfiable if not $C \sqsubseteq \bot$.

➳ Subsumption can be reduced to (un)satisfiability and vice versa.

DL systems are more than a concept language



TBox: Motivation

TBoxes capture:

- Intensional knowledge
- Categories from the application domain in terms of concepts
- Relations from the application domain in terms of roles
- Relations between concepts
  e.g. assign names to complex concepts or
  state super- / sub-concept relationships

TBox: Syntax

TBox $\mathcal{T}$: Finite set of concept axioms.

There are different kinds of concept axioms:

- Primitive concept definition
  $A \sqsubseteq D \qquad A \in N_C$
  ➳ necessary conditions
- Concept definition
  $A \equiv D \qquad A \in N_C$
  ➳ necessary & sufficient conditions
- Concept disjointness axiom
  $C \sqcap D \sqsubseteq \bot$
- General concept equivalence
  $C \equiv D$
- General concept (inclusion) axiom (GCI)
  $C \sqsubseteq D$

GCIs can express other concept axioms

- Primitive concept definition: $A \sqsubseteq D$ $\quad A \in N_C$

  Pick a concept name for left-hand side.

- Concept definition: $A \equiv D$ $\quad A \in N_C$

  $A \sqsubseteq D, \quad D \sqsubseteq A$

- Concept disjointness axiom : $C \sqcap D \sqsubseteq \bot$

  Is a GCI.

- General concept equivalence: $C \equiv D$

  $C \sqsubseteq D, \quad D \sqsubseteq C$

Kinds of TBoxes

1. TBox $\mathcal{T}$ is a general TBox, if
   - it is a finite set of concept axioms
   - cyclic definitions and GCIs are allowed

$$\{\mathsf{WildAnimal} \equiv \mathsf{Animal} \sqcap \neg\exists \mathsf{owner}.\top,$$
$$\mathsf{Mammal} \sqcap \exists \mathsf{bodypart}.\mathsf{Hunch} \equiv \mathsf{Camel} \sqcup \mathsf{Dromedary}\}$$

2. TBox $\mathcal{T}$ is an unfoldable TBox, if it has
   - only (primitive) concept definitions
   - concept names at most once on the left-hand side of definitions
   - no cyclic definitions, no GCIs

$$\{\mathsf{Elephant} \equiv \mathsf{Mammal} \sqcap \exists \mathsf{bodypart}.\mathsf{Trunk}$$
$$\mathsf{Mammal} \equiv \mathsf{Elephant} \sqcup \mathsf{Lion} \sqcup \mathsf{Zebra}\}$$

➽ Unfoldable TBoxes can be conceived as macro definitions.

TBox: Semantics

Semantics of concept axioms

$A \equiv C$ holds in an interpretation $\mathcal{I}$ iff $A^{\mathcal{I}} = C^{\mathcal{I}}$

$C \sqsubseteq D$ holds in an interpretation $\mathcal{I}$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

Semantics of the TBox

An interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ if

$C^{\mathcal{I}} = D^{\mathcal{I}}$ for all $C \equiv D \in \mathcal{T}$ and
$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all $C \sqsubseteq D \in \mathcal{T}$.

Terminological Reasoning Services

Reasoning tasks for TBoxes:

1. **Concept satisfiability w.r.t. TBoxes**

   Given C and $\mathcal{T}$, does there exist a common model of C and $\mathcal{T}$?

   "Does C state a contradiction w.r.t. $\mathcal{T}$?"

2. **Concept subsumption w.r.t. TBoxes** $(C \sqsubseteq_{\mathcal{T}} D)$

   Given C,D and $\mathcal{T}$, does $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ hold in all models of $\mathcal{T}$?

   "Is C more specific than D w.r.t. the axioms in $\mathcal{T}$?"

3. **Concept equivalence w.r.t. TBoxes** $(C \equiv_{\mathcal{T}} D)$

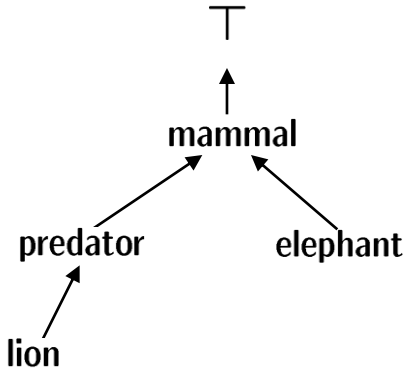
   $C \equiv_{\mathcal{T}} D$ iff $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$

   "Are C and D equivalent w.r.t. $\mathcal{T}$?"

4. **Classification of the TBoxes**

   Computation of all subsumption relationships between all concept names in $\mathcal{T}$.

   $\Longrightarrow$ Subsumption can be used to compute a concept hierarchy:



Example for TBox reasoning

TBox

$$\text{Elephant} \equiv \text{Mammal} \sqcap \exists \text{has-bodypart}.\text{Trunk} \sqcap \forall \text{has-color}.\text{Grey}$$

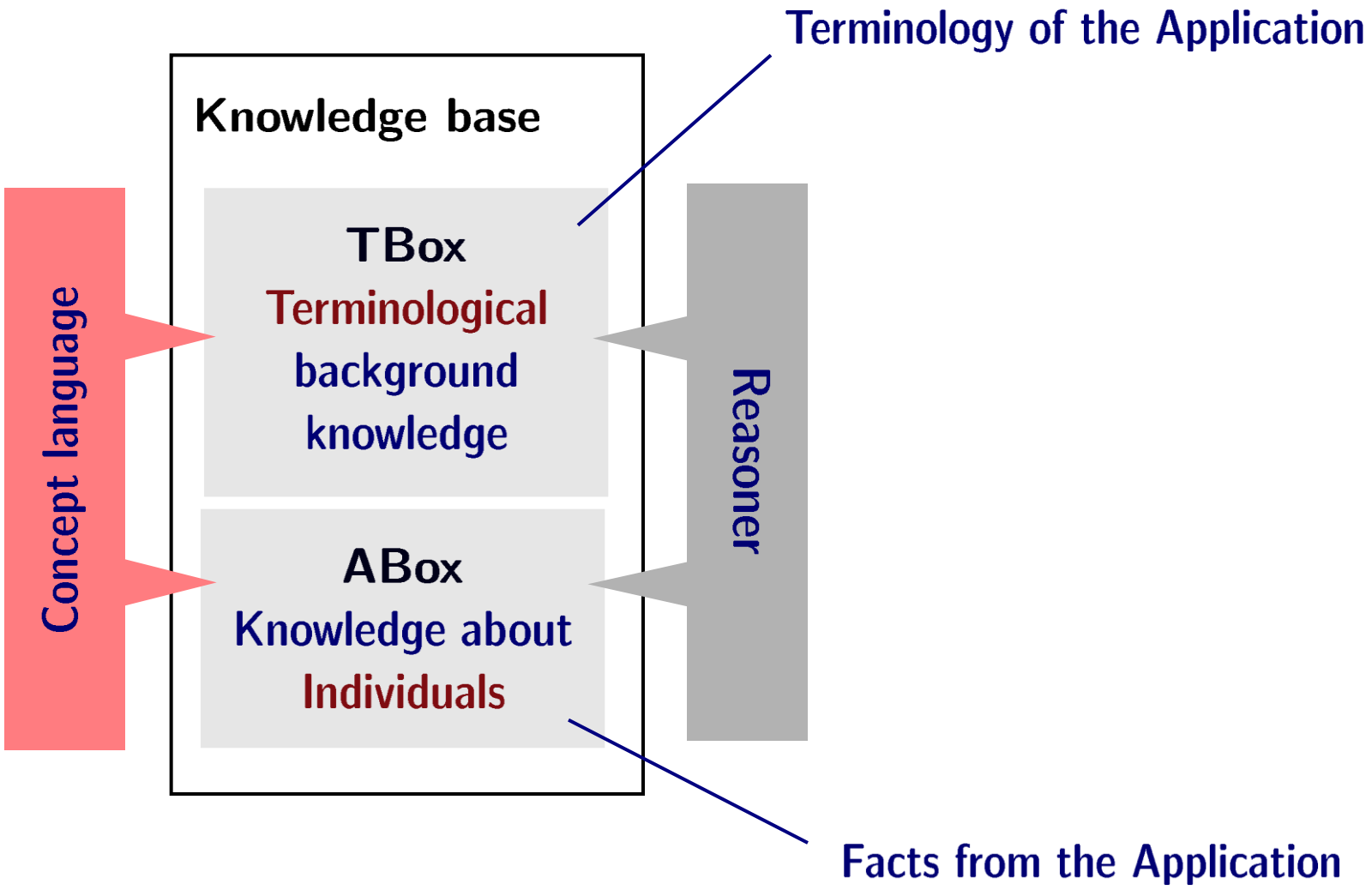
$$\text{AlbinoElephant} \equiv \text{Elephant} \sqcap \forall \text{has-color}.\text{White} \sqcap \exists \text{color}.\text{White}$$

~~$\text{Grey} \sqcap \text{White} \sqsubseteq \bot$~~

$$\text{White} \sqsubseteq \text{Grey}$$

1. TBox is consistent.
2. TBox is inconsistent.
3. AlbinoElephant $\sqsubseteq$ Elephant w.r.t. TBox.

DL systems are more than a concept language



The assertional part

ABoxes capture:

- extensional knowledge
- facts from the application domain
- knowledge about individuals
- knowledge about the relations between individuals

ABoxes: syntax

An ABox $\mathcal{A}$ is a finite set of assertions.

ABox assertions in DL systems are:

- Concept assertions: $C(a)$
- Role assertions: $(a, b) : R$
- (In)equality assertions: $a \equiv b \quad (a \not\equiv b)$

Typically, ABoxes use concept names and role names from the TBox.

ABoxes: semantics

Idea: Interpretation $\mathcal{I}$ maps each individual name a to an element of $\Delta^{\mathcal{I}}$.

Semantics of assertions:

- Concept Assertions: $\mathcal{I}$ satisfies $C(a) \iff a^{\mathcal{I}} \in C^{\mathcal{I}}$
  "$\mathcal{I}$ satisfies $C(a)$ iff the extension of a is in the extension of C."

- Role Assertions: $\mathcal{I}$ satisfies $(a, b) : R \iff (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$
  "$\mathcal{I}$ satisfies $(a, b) : R$ iff the extension of a and b is in the extension of R."

- (In)equality assertions: $\mathcal{I}$ satisfies $a \not\equiv b \iff a^{\mathcal{I}} \not\equiv b^{\mathcal{I}}$
  "$\mathcal{I}$ satisfies $a \not\equiv b$ iff the extension of a is (not) in the extension of b."

Semantics of ABoxes:

$\mathcal{I}$ is a model for an ABox $\mathcal{A}$ if $\mathcal{I}$ satisfies all assertions in $\mathcal{A}$.

Example: ABox

ABox is a partial description of the world.

ABox $\mathcal{A}$

$\mathsf{Mammal}(\mathsf{dumbo})$

$\mathsf{Darkgrey}(\mathsf{g23})$

$(\mathsf{dumbo}, \mathsf{g23}) : \mathsf{has\text{-}color}$

$\forall \mathsf{has\text{-}color}.\mathsf{Lightgrey}(\mathsf{dumbo})$

$\mathsf{Trunk}(\mathsf{t14})$

$(\mathsf{dumbo}, \mathsf{t14}) : \mathsf{has\text{-}bodypart}$

$(\mathsf{dumbo}, \mathsf{dumbo}) : \mathsf{likes\text{-}most}$

Assertional Reasoning Services

Reasoning tasks for ABoxes:

1. **ABox consistency**
   Given an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$, do they have a common model?

2. **Instance checking**
   Given an ABox $\mathcal{A}$, a TBox $\mathcal{T}$, an individual name a, and a concept C does $a^{\mathcal{I}} \in C^{\mathcal{I}}$ hold in all models of $\mathcal{A}$ and $\mathcal{T}$?

3. **ABox realization**
   Given an ABox $\mathcal{A}$, and a TBox $\mathcal{T}$,
   compute for each individual:
   of which named concepts $C \in \mathcal{T}$ is it an instance of?

Example for ABox Reasoning

ABox is a partial description of the world.

ABox

$\mathsf{Mammal}(\mathsf{dumbo})$

~~$\mathsf{Darkgrey}(\mathsf{g23})$~~

~~$(\mathsf{dumbo}, \mathsf{g23}) : \mathsf{has\text{-}color}$~~

$\forall \mathsf{has\text{-}color}.\mathsf{Lightgrey}(\mathsf{dumbo})$

$\mathsf{Trunk}(\mathsf{t14})$

$(\mathsf{dumbo}, \mathsf{t14}) : \mathsf{has\text{-}bodypart}$

$(\mathsf{dumbo}, \mathsf{dumbo}) : \mathsf{likes\text{-}most}$

TBox

$$\mathsf{Elephant} \equiv \mathsf{Mammal} \sqcap \exists \mathsf{has\text{-}bodypart}.\mathsf{Trunk} \sqcap \forall \mathsf{has\text{-}color}.\mathsf{Grey}$$

$$\mathsf{Grey} \equiv \mathsf{Lightgrey} \sqcup \mathsf{Darkgrey}$$

$$\bot \equiv \mathsf{Lightgrey} \sqcap \mathsf{Darkgrey}$$

1. ABox is inconsistent w.r.t. TBox.
2. dumbo is an instance of Elephant

Tutorial overview

1. DLs: Basic notions
   1.1. concept descriptions,
   1.2. TBox,
   1.3. ABox

   and their reasoning services
2. Reasoning in $\mathcal{ALC}$
3. OWL 2: beyond $\mathcal{ALC}$
4. OWL 2 EL
   4.1 the $\mathcal{EL}$-family
   4.2. Reasoning method for $\mathcal{EL}$
5. OWL 2 QL
   5.1. DL-Lite family
   5.2. Conjunctive queries
6. Conclusions

Why automated reasoning?

TBox and the ABox capture implicit information.
We want to access this information by making it explicit!

Does my knowledge base . . .

- contain a concept that cannot have instances? (since its definition is contradictory.)
  Check for satisfiabiliy w.r.t. TBox.
- contain an unwanted synonym for a concept? (unwanted / unintended redundancy in my TBox)
  Check for equivalent concepts.
- yield the concept hierarchy I wanted?
  Classify.
- contain individuals not compliant with the specification of the concepts they belong to?
  Check ABox consistency.

Requirements for good reasoning algorithms:

They should be decision procedures,i.e.:

- terminating, — You get always an answer.
- correct, — Every answer is correct.
- complete. — You get all correct answers.

➽ Prerequisit for safe and reliable applications!

Reasoning in $\mathcal{ALC}$

Goal:

Find algorithm for satisfiability and subsumption
(for now: only concept or unfoldable TBoxes
no general TBoxes)

Subsumption can be reduced to (un)satisfiability

$\Longrightarrow$ we can restrict ourselves to satisfiability

Tableau Algorithms

- Used to prove decidability/complexity bounds of DLs
- Most state-of-the-art DL reasoners are based on tableau algorithms

Use the reduction

Reformulate a ...	as an ABox consistency check
satisfiability test: sat(C)?	Consistent: $(\mathcal{T}, \{C(a)\})$?
subsumption test: $C \sqsubseteq_{\mathcal{T}} D$?	Inconsistent: $(\mathcal{T}, \{C \sqcap \neg D(a)\})$?
instance check: $\mathcal{A}, \mathcal{T} \models C(a)$?	Inconsistent: $(\mathcal{T}, \mathcal{A} \cup \{\neg C(a)\})$?

Reasoning method for unfoldable $\mathcal{ALC}$-KBs

1. Use the reduction to reformulate the reasoning problem

2. Expand concepts w.r.t. TBox

3. Normalize concept descriptions

4. Apply tableau rules

Expansion of concept descriptions

$\Longrightarrow$ Reduce satisfiability w.r.t. TBoxes to satisfiability without TBoxes

I.e. get rid of the TBox.

Naive approach for expansion:

Let C_0 be concept, $\mathcal{T}$ unfoldable TBox

1. replace every concept name of a defined concept with the right-hand side of its definitions $A \equiv C$
2. repeat until no more replacements can be made.

Expansion of concept descriptions II

Expansion process terminates due to acyclicity of the concept definitions!

But: exponential blow-up in the worst case!

$$A_0 \equiv \forall R.A_1 \sqcap \forall S.A_1$$
$$A_1 \equiv \forall R.A_2 \sqcap \forall S.A_2$$
$$\vdots$$
$$A_{k-1} \equiv \forall R.A_k \sqcap \forall S.A_k$$

Negation Normal Form

A concept C is in negation normal form (NNF) if negation occurs only in front of concept names.

Transformation rules:

$$\begin{aligned}
\neg\neg C &\rightsquigarrow C \\
\neg(C \sqcap D) &\rightsquigarrow \neg C \sqcup \neg D \\
\neg(C \sqcup D) &\rightsquigarrow \neg C \sqcap \neg D \\
\neg(\exists R.C) &\rightsquigarrow \forall R.\neg C \\
\neg(\forall R.C) &\rightsquigarrow \exists R.\neg C
\end{aligned}$$

Tableau Algorithm: Idea

Try to construct a model for the input concept C_0 as follows:

- Represent models by proof ABoxes
- To decide satisfiability of C_0,
  start with initial proof ABox $\mathcal{A}_{C_0}$
- Repeatedly apply tableau rules
  and check for obvious contradictions
- Return "satisfiable" iff a complete and contradiction-free
  proof ABox was found

Proof ABox

Tableau algorithm works on sets of ABoxes: $\mathcal{S}$

Initially, $\mathcal{S}$ contains proof ABox for concept C_0:

$\mathcal{S} := \{\mathcal{A}_0\}$, with $\mathcal{A}_0 := \{C_0(x_0)\}$

Apply tableau rules to set of proof ABoxes $\mathcal{S}$ until

- a proof ABox is complete or
- there exists an individual x in $\mathcal{A}$ such that $\{B(x), \neg B(x)\} \subseteq \mathcal{A}$ for some concept name B or $\bot(x) \in \mathcal{A}$. (Clash)

Tableau rules for $\mathcal{ALC}$

	Precondition	Replace $\mathcal{A}$ by:
$\longrightarrow_{\sqcap}$	$(C_1 \sqcap C_2)(x) \in \mathcal{A}$ $C_1(x) \notin \mathcal{A}$ or $C_2(x) \notin \mathcal{A}$	$\mathcal{A}' := \mathcal{A} \cup \{C_1(x), C_2(x)\}$
$\longrightarrow_{\sqcup}$	$(C_1 \sqcup C_2)(x) \in \mathcal{A}$ $C_1(x) \notin \mathcal{A}$ and $C_2(x) \notin \mathcal{A}$	$\mathcal{A}' := \mathcal{A} \cup \{(C_1)(x)\}$ $\mathcal{A}'' := \mathcal{A} \cup \{(C_2)(x)\}$
$\longrightarrow_{\exists}$	$(\exists r.C)(x) \in \mathcal{A}$, but no z in $\mathcal{A}$ s.t. $\{r(x,z), C(z)\} \subseteq \mathcal{A}$	$\mathcal{A}' := \mathcal{A} \cup \{r(x,z), C(z)\}$
$\longrightarrow_{\forall}$	$\{(\forall r.C)(x), r(x,y)\} \subseteq \mathcal{A}$, but $C(y) \notin \mathcal{A}$	$\mathcal{A}' := \mathcal{A} \cup \{C(y)\}$

Correctness of the Algorithm

1. The algorithm terminates on any input

2. If the algorithm returns “satisfiable”,
   then the input concept has a model.

3. If the algorithm returns “not satisfiable”,
   then the input concept has no model.

Tableau algorithm for general TBoxes

Treatment of GCIs

1. Code all GCIs into one.

   For $\mathcal{T} = \{ C_1 \sqsubseteq D_1,\ C_2 \sqsubseteq D_2, \ldots,\ C_n \sqsubseteq D_n \}$
   build the GCI $\top \sqsubseteq C_{GCI}$ with
   $C_{GCI} \equiv (\neg C_1 \sqcup D_1) \sqcap (\neg C_2 \sqcup D_2) \sqcap \cdots \sqcap (\neg C_n \sqcup D_n)$

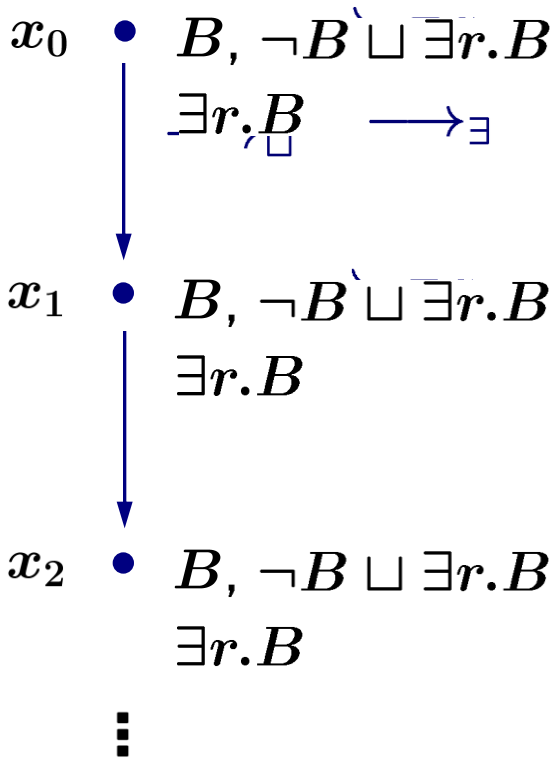
2. New tableau rule: assert C_{GCI} for every individual

   $\longrightarrow_{\top \sqsubseteq C_{GCI}}$: If x in $\mathcal{A}$ and $C_{GCI}(x) \notin \mathcal{A}$,
   then replace $\mathcal{A}$ with $\mathcal{A}' = \mathcal{A} \cup C_{GCI}(x)$

Problem: termination

Consider: $\mathcal{T} = \{B \sqsubseteq \exists r.B\}$

with $C_{GCI} = \neg B \sqcup \exists r.B$



Remedy: Blocking of application of $\longrightarrow_\exists$

An individual x is blocked
by an individual y, iff:

- x was generated by $\longrightarrow_\exists$ after y
- $\{C \mid C(x) \in \mathcal{A}\} \subseteq \{C \mid C(y) \in \mathcal{A}\}$

Tableaux for DLs

For $\mathcal{ALC}$:

- with blocking: termination regained
- Complexity of reasoning w.r.t. general TBoxes: ExpTime-complete

Recall:

- one reasoning method for many reasoning services
- approach requires presence of negation in the DL

Tableau method implemented in most state-of-the-art DL reasoners

For extensions of $\mathcal{ALC}$:

- tableau rules for additional concept constructors

Tutorial overview

1. DLs: Basic notions
   1.1. concept descriptions,
   1.2. TBox,
   1.3. ABox

   and their reasoning services
2. Reasoning in $\mathcal{ALC}$

3. OWL 2: beyond $\mathcal{ALC}$
4. OWL 2 EL
   4.1 the $\mathcal{EL}$-family
   4.2. Reasoning method for $\mathcal{EL}$
5. OWL 2 QL
   5.1. DL-Lite family
   5.2. Conjunctive queries
6. Conclusions

Expressive power vs. computational complexity

In many cases, the expressive power of $\mathcal{ALC}$ does not suffice:

- an elephant has precisely four bodyparts that are legs
- every elephant has a bodypart which is a trunk
  and every trunk is a bodypart of an elephant

Many extensions of $\mathcal{ALC}$ have been developed, for example:

- qualified number restrictions $(\leq n\ r\ C)$ and $(\geq n\ r\ C)$
- inverse roles r^- to be used in existential and universal restriction

But: Increasing expressivity also increases computational complexity

➽ Tradeoff between expressivity and computational complexity!

Beyond $\mathcal{ALC}$: concept constructors

Number restrictions $(\leq n\, r),\ (\geq n\, r)$

$$(\leq n\, r)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \#\{y \mid (x,y) \in r^{\mathcal{I}}\} \leq n\}$$

$$(\geq n\, r)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \#\{y \mid (x,y) \in r^{\mathcal{I}}\} \geq n\}$$

Qualified number restrictions $(\leq\ n\ r\ C),\ (\leq\ n\ r\ C)$

$$(\leq n\ r\ C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \#\{y \mid (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \leq n\}$$

$$(\geq\ n\ r\ C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \#\{y \mid (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geq n\}$$

Example:

$$\text{Car} \sqcap (\geq 5\ \text{has-seat}) \sqcap (\leq 5\,\text{has-seat})$$
$$\sqcap\ (\geq 1\ \text{has-seat Drivers-seat}) \sqcap (\leq 1\ \text{has-seat Drivers-seat})$$

Beyond $\mathcal{ALC}$: Concept constructors II

Sometimes it is useful to refer to individuals in the TBox.

Recall: If they have same description

- concepts are equivalent.
- individuals are distinct.

C $\equiv$ ($\forall$ has-child.$\bot$)
D $\equiv$ ($\leq$ 0 has-child)
$\Longrightarrow$ C $\equiv$ D

(Carla, Luisa): parent, Person(Carla),
(Markus, Luisa): parent, Person(Markus)
$\Longrightarrow$ Carla $\not\equiv$ Markus

Concept constructors using individuals:

- Nominals $\{a\}$ $\quad\{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}$
- One-of $\{a_1, \ldots, a_n\}$ $\quad\{a_1, \ldots, a_n\}^{\mathcal{I}} = \{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$

E.g.: RomanCatholic $\sqsubseteq$ $\exists$ knows.{Pope}

Beyond $\mathcal{ALC}$: Roles

Role declarations

R	atomic role	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ e.g. has-child
F	feature or attribute	$F^{\mathcal{I}} = \{(x,y) \mid (x,y) \in F^{\mathcal{I}} \wedge (x,z) \in F^{\mathcal{I}} \Rightarrow y = z\}$ e.g. has-mother
$R \sqsubseteq S$	role inclusion role hierarchy	$R \sqsubseteq S$ holds in $\mathcal{I} \Leftrightarrow R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$

e.g. has-mother $\sqsubseteq$ has-parent $\sqsubseteq$ has-family-member

has-sibling $\sqsubseteq$ has-family-member

Beyond $\mathcal{ALC}$: Roles II

Role declarations

R^+ transitive Role $(R^+)^{\mathcal{I}} = \{(x,z) \mid (x,y) \in R^{\mathcal{I}}, (y,z) \in R^{\mathcal{I}} \Rightarrow (x,z) \in R^{\mathcal{I}}\}$

e.g. has-ancestor

R^- inverse Role $(R^-)^{\mathcal{I}} = \{(y,x) \mid (x,y) \in R^{\mathcal{I}}\}$

e.g. $(\text{has-parent})^- = \text{has-child}$

$S \circ R \sqsubseteq R$ right identities $(x,y) \in S^{\mathcal{I}} \wedge (y,z) \in R^{\mathcal{I}} \rightarrow (x,z) \in R^{\mathcal{I}}$

e.g. has-part $\circ$ has-function $\sqsubseteq$ has-function

Excursus: names of description logics

Basis-DL: $\mathcal{ALC}$

- $\mathcal{E}$: Existential restrictions
- $\mathcal{N}$: Number restrictions
- $\mathcal{Q}$: Qualified number restrictions
- $\mathcal{O}$: nominals, Objects
- $\mathcal{F}$: Features, functional roles
- $^+$: Transitive roles
- $\mathcal{I}$: Inverse roles
- $\mathcal{H}$: role Hierarchies
- $\mathcal{R}$: right identities

$\mathcal{S}$: Abbreviation for $\mathcal{ALC}^+$

Complexity of subsumption in some DLs

	unfoldable TBox	general TBox
$\mathcal{EL}$ $(\sqcap, \exists)$	P	P
$\mathcal{ALC}$	PSPACE	EXPTIME
$\mathcal{ALCNIO}$	EXPTIME	NEXPTIME
$\mathcal{SHOIQ}$	NEXPTIME	NEXPTIME
$\mathcal{SROIQ}$		2-NEXPTIME

- These are worst case complexities!
  - worst case does not need to appear in practice
  - worst case requires regular structure of the KB

Tableaux reasoner systems for DLs

- Fact++ — $\mathcal{SROIQ}$
- Pellet — $\mathcal{SHOIQ}$
- RacerPro — $\mathcal{SHIQ}$
- Hermit — $\mathcal{SHIQ}$
  employs Hypertableaux
- CB — Horn-$\mathcal{SHIF}$
  consequence-based reasoner — fragment of OWL 2

Most systems offer more reasoning functionality

Complexity of subsumption in some DLs

	unfoldable TBox	general TBox
$\mathcal{EL}$ $(\sqcap, \exists)$	P	P
$\mathcal{ALC}$	PSPACE	EXPTIME
$\mathcal{ALCNIO}$	EXPTIME	NEXPTIME
$\mathcal{SHOIQ}$	NEXPTIME	NEXPTIME
$\mathcal{SROIQ}$		2-NEXPTIME

- OWL DL
- OWL 2

Tutorial overview

1. DLs: Basic notions
   1.1. concept descriptions,
   1.2. TBox,
   1.3. ABox

   and their reasoning services
2. Reasoning in $\mathcal{ALC}$
3. OWL 2: beyond $\mathcal{ALC}$
4. OWL 2 EL
   4.1 the $\mathcal{EL}$-family
   4.2. Reasoning method for $\mathcal{EL}$
5. OWL 2 QL
   5.1. DL-Lite family
   5.2. Conjunctive queries
6. Conclusions

Light-weight DL: $\mathcal{EL}$

$\mathcal{EL}$ $(\sqcap, \exists, \top)$ is used in medical and biological ontologies

$$\mathsf{Inflammation} \sqsubseteq \mathsf{Disease} \sqcap \exists \mathsf{acts\text{-}on}.\mathsf{Tissue}$$

$$\mathsf{HeartDisease} \equiv \mathsf{Disease} \sqcap \exists \mathsf{has\text{-}loc}.\exists \mathsf{comp\text{-}of}.\mathsf{Heart}$$

$$\mathsf{Retina} \sqsubseteq \mathsf{Tissue} \sqcap \exists \mathsf{has\text{-}loc}.\mathsf{Eye}$$

The $\mathcal{EL}$ family

Prominent members:

$\mathcal{EL}$: $\quad \sqcap, \exists, \top$

$\mathcal{EL}^+$ extends $\mathcal{EL}$ by:
- role chain inclusions: $r \circ s \sqsubseteq t$.
  e.g. has-mother $\circ$ has-sister $\sqsubseteq$ has-aunt.)

$\mathcal{EL}^{++}$ extends $\mathcal{EL}^+$ by:
- $\bot$
- nominals
- corresponds to OWL 2 EL profile

Typically, used with general TBoxes!

Subsumption in $\mathcal{EL}$

Steps of the subsumption algorithm for $\mathcal{EL}$:

1. Normalize TBox
2. Translate normalized TBox into "Completion Sets"
3. Saturate sets using completion rules
4. Read off subsumption relationships from completeion sets

TBox Normalization I

Assumptions:

- consider only the subsumption of concept names w.r.t. TBoxes

  $C \sqsubseteq_{\mathcal{T}} D$ iff $A \sqsubseteq_{\mathcal{T}'} B$, where $\mathcal{T}' = \mathcal{T} \cup \{A \equiv C, B \equiv D\}$

- TBox statements always have the form $C \sqsubseteq D$

  $C \equiv D$ equivalent to $C \sqsubseteq D, D \sqsubseteq C$

Normalized TBox only has statements of the following form:

$$A \sqsubseteq B$$
$$A \sqcap A' \sqsubseteq B$$
$$A \sqsubseteq \exists R.B$$
$$\exists R.A \sqsubseteq B$$

A, A', B: concept names or $\top$

TBox Normalization II

Normalization can be achieved by introducing new concept names:

Let $C \sqsubseteq D \in \mathcal{T}$ with C, D complex concepts

Top-most operator in D (C treated analogously):

- $D = \exists R.E$

  $C \sqsubseteq D \quad \rightsquigarrow \quad A \sqsubseteq E,\ E \sqsubseteq A,\ C \sqsubseteq \exists R.A$

- $D = E \sqcap F$

  introduce "new names" for E and F

Additional case:

$C \sqsubseteq D \sqcap E$ replaced by $C \sqsubseteq D$ and $C \sqsubseteq E$

Completion Sets

Compute a completion set $S(A)$ for every concept name A in $\mathcal{T}$ (and every existential restriction)

Intuition:

- $S(A)$ contains all concept names B with $A \sqsubseteq_{\mathcal{T}} B$
- $S(A, r)$ contains all concept names B with $A \sqsubseteq \exists r.B$

Initially:

- $S(A) = \{A, \top\}$ for each concept name A
- $S(A, r) = \emptyset$ for each concept name A and role name r

Completion rules

Then saturate completion sets using the rules:

CR1 $A' \sqsubseteq B \in \mathcal{T}$, $A' \in S(A)$, $B \notin S(A)$ $\rightsquigarrow$ add B to $S(A)$

CR2 $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}$, $A_1, A_2 \in S(A)$, $B \notin S(A)$ $\rightsquigarrow$ add B to $S(A)$

CR3 $A' \sqsubseteq \exists r.B \in \mathcal{T}$, $A' \in S(A)$, $B \notin S(A, r)$ $\rightsquigarrow$ add B to $S(A, r)$

CR4 $\exists r.C' \sqsubseteq B \in \mathcal{T}$, $C' \in S(C)$, $C \in S(A, r)$, $B \notin S(A)$ $\rightsquigarrow$ add B to $S(A)$

Completion Sets II

Lemma. 1. $B \in S(A)$ iff $A \sqsubseteq_{\mathcal{T}} B$.

2. subsumption sets can be computed in polynomial time.

ad 2.

- there are at most $|\mathcal{T}|$ completion sets to be computed;
- each completion set contains at most $|\mathcal{T}|$ elements

$\Longrightarrow$ at most $|\mathcal{T}|^2$ rule applications

- searching for an applicable rule needs time at most $|\mathcal{T}|^2$

$\Longrightarrow$ algorithm runs in time $|\mathcal{T}|^4$

Completion Sets for ABox Realization

Compute a completion set $S(A)$ for every

- concept name A in $\mathcal{T}$ (and every existential restriction),
- every individual (and every role successor)

Intuition:

- $S(A)$ contains all concept names B with $A \sqsubseteq_{\mathcal{T}} B$
- $S(A, r)$ contains all concept names B with $A \sqsubseteq \exists r.B$
- $S(a)$ contains all concept names B with a instance of B
- $S(a, r)$ contains all concept names B with a instance of $\exists r.B$ (and individuals b with $r(a, b) \in \mathcal{A}$)

Completion-based reasoners for $\mathcal{EL}$

CEL (Classifier for $\mathcal{EL}$)

- supported DL: $\mathcal{EL}+$
- for "good natured" TBoxes: incremental classification
- explanation of subsumption relations (axiom pinpointing)

jCEL (Java classifier for $\mathcal{EL}$)

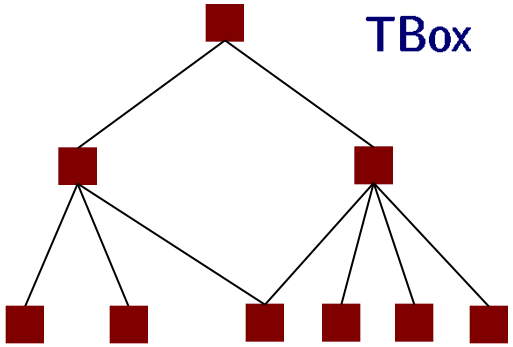
- supported DL: $\mathcal{ELHI}f_{\mathcal{R}}$
- TBox classification + ABox realization

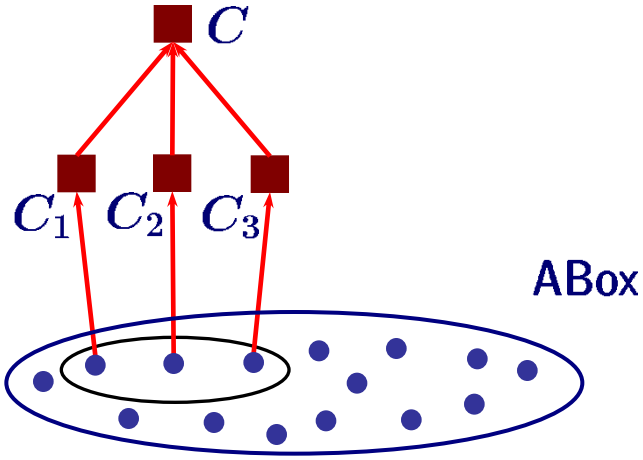
Bottom-up construction of KBs

Idea:

Allow the knowledge engineer to define concepts by giving examples:

1. examples given by ABox individuals
2. for each example, compute the most specific concept describing the individual
3. extract their commonalities by computing the least common subsumer





Generalizations in $\mathcal{EL}$

LCS does not always exist in the presence of general TBoxes:

$$A \equiv \exists R.A$$
$$B \equiv \exists R.B$$

LCS(A, B) w.r.t. $\mathcal{T}$: ???

Approach without TBoxes:

1. Represent $\mathcal{EL}$-concepts by $\mathcal{EL}$-trees
2. Compute the LCS as the product of $\mathcal{EL}$-trees.

Approximative solution for general $\mathcal{EL}$-TBoxes:

1. Compute the LCS up to a fixed role-depth.
2. Use the completion sets to generate the $\mathcal{EL}$-trees

Tutorial overview

1. DLs: Basic notions
   1.1. concept descriptions,
   1.2. TBox,
   1.3. ABox

   and their reasoning services
2. Reasoning in $\mathcal{ALC}$
3. OWL 2: beyond $\mathcal{ALC}$
4. OWL 2 EL
   4.1 the $\mathcal{EL}$-family
   4.2. Reasoning method for $\mathcal{EL}$
5. OWL 2 QL
   5.1. DL-Lite family
   5.2. Conjunctive queries
6. Conclusions

Light-weight DL: DL-Lite family

DL-Lite$_{core}$

- concept constructors: $A, \exists r.\top, \exists r^-.\top$
- roles: inverse
- TBox axioms: $C \sqsubseteq D$

DL-Lite$_{\mathcal{F}}$ extends DL-Lite$_{core}$ by:

- (inverse) functional roles

DL-Lite$_{\mathcal{R}}$ extends DL-Lite$_{core}$ by:

- role hierarchy
- role disjointness axioms

. . . for roles and their inverse

Core of the OWL 2 QL profile

DL-Lite family

- designed for ontology-based data access
- tailored towards applications that need to handle huge amounts of data
- allow efficient querying of ABoxes (w.r.t. a fairly light-weight TBox)

➳ required to store ABox in relational data base system and use relational DB engine for querying

Conjunctive Queries

Definition:

Conjunctive Query $q(X)$ w.r.t. a KB $= (\mathcal{T}, \mathcal{A})$ has the form:

$$q(X) \longleftarrow \exists Y.conj(X, Y).$$

where

X : is a tuple of answer set variables

Y : is a tuple of free variables

$conj$: is a set of expressions of the form:
$C(z_i)$ or $r(z_i, z_j)$, where C:
concept description, r: role and $\{z_i, z_j\} \subseteq X \cup Y$.

➳ Limited form of FOL queries

Conjunctive Queries

Query answering:

the answers to a query q w.r.t. a KB $\mathcal{K}$ is
the set of tuples of individuals from $\mathcal{K}$ $(a_1, \ldots, a_n)$ such that

$$(a_1^{\mathcal{I}}, \ldots, a_n) \in q\mathcal{I}$$

for all models of $\mathcal{K}$.

"Compute tuples from $\mathcal{K}$ for which the query q always evaluates to true."

Recall:

- "traditional" ABox queries are concept-based
  They allow only for tree-shaped queries.

  Example.: Query for all Persons, who are their best friend.

  ➳ use of variables!

- Conjunctive queries are a more flexible query language!

Approach for query answering in DL-Lite

1.) Use TBox to reformulate query q into a FOL-query $q_{\mathcal{T}}$, discard the TBox

"Compile TBox information into the query."

2.) View ABox $\mathcal{A}$ as a relational database $\mathcal{I}_{\mathcal{A}}$, where
- $A^{\mathcal{I}_{\mathcal{A}}} = \{a \mid A(a) \in \mathcal{A}\}$
- $r^{\mathcal{I}_{\mathcal{A}}} = \{(a, b) \mid r(a, b) \in \mathcal{A}\}$

3.) Evaluate $q_{\mathcal{T}}$ in the DB $\mathcal{I}_{\mathcal{A}}$ using a relational query engine.

Excursus: complexity

Complexity classes: definied w.r.t. size of the input.
Ontology-based applications:
Sometimes the TBox is small compared to the ABox!

Query complexity:

complexity w.r.t. number of conjunct of the query.

Data complexity:

complexity w.r.t. size of the ABox.

Combined complexity:

:= complexity w.r.t. size of the TBox and the ABox.

Complexity for query answering for conjunctive queries

Expressive DLs:

- $\mathcal{SHIQ}$
  combined complexity: 2ExpTime-complete
- $\mathcal{ALCI}$
  combined complexity: 2ExpTime-complete

light-weight DLs:

- $\mathcal{EL}$:
  NP-hard für combined complexity
- $\mathcal{DL}$-Lite:
  PTime in size of TBox
  LogSpace for data complexity

DL Lite family in practice

- allows for efficient ontology-based data access
- DL Lite$_{\mathcal{R}}$: maximal DL with LogSpace data complexity
- query answering for DL Lite$_{\mathcal{R}}$ is implemented in QuOnto
- DL Lite$_{\mathcal{R}}$: core of OWL 2 QL

Conclusions

- DLs offer a variety of ontology languages to model application domains
- DLs have formal semantics which is used to define reasoning services
- DL systems offer powerful reasoning services
- OWL 2 offers 3 DL-based ontology languages
- OWL 2 offers 2 lightweight ontology languages for which reasoning scales well
- Semantic Web applications are needed to demand / inspire new ontology services