Introduction to Description Logics

Anni-Yasmin Turhan
Technische Universität Dresden
Institute for Theoretical Computer Science

TU
Dresden

Knowledge Representation

General goal of knowledge representation:

> "Develop formalisms for providing high-level descriptions of the world
> that can be effectively used to build intelligent applications."

- **formalisms**:
  formal syntax and formal and unambiguous semantics
- **high-level descriptions**:
  which aspects should be represented, which left out?
- **intelligent applications**:
  are able to infer new knowledge from given knowledge
- **effectively used**:
  reasoning techniques should allow “usable” implementation

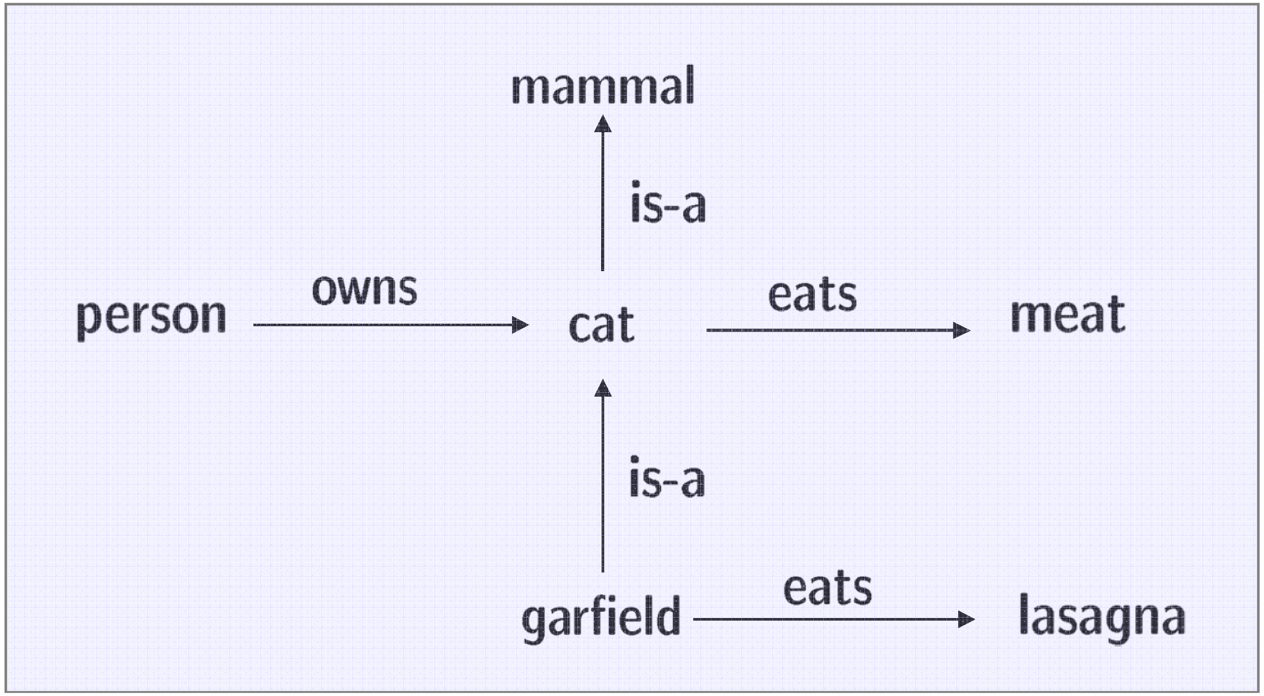
Early knowledge representation systems

How to represent terminological knowledge?

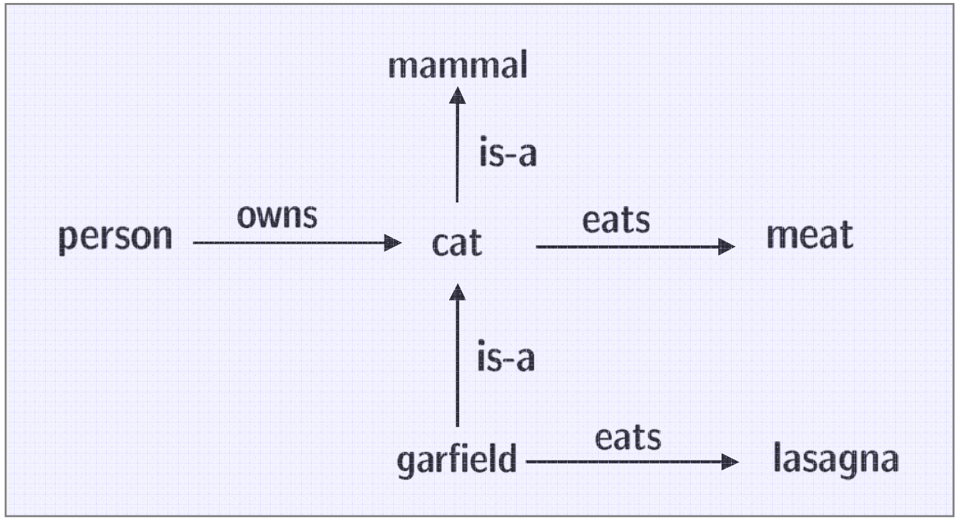
Semantic Networks

- representation by graph-based formalism
- models entities and their relations

For example:



Semantic networks: Drawbacks



Unclear semantics

- What does a node mean?
- What does a link in the graph mean?
  - 'is-a' has different meanings!
  - 'eats': One thing that cats eat is meat?
    All things that cats eat is meat?

Problems: missing semantics (reasoning!), complex pictures

➽ Ad-hoc methods for automated reasoning.

➽ Result of automated reasoning is system dependent!

Remedy: Use a logical formalism for KR rather than pictures

On phases of DL research

Early phase — eighties

- structural reasoning procedures
  (bring concepts to a normal form and then compare their structure)
- sound, but incomplete reasoning systems
- complete reasoning regarded as not feasible (since intractable)

Second phase — nineties

- investigation of sound and complete reasoning procedures
  Tableaux method
- complexity results and reasoning procedures for
  increasingly expressive DLs
- optimized implementations of reasoning procedures
  e.g. FaCT system ('98), RACER system ('99)

On phases of DL research

Third phase

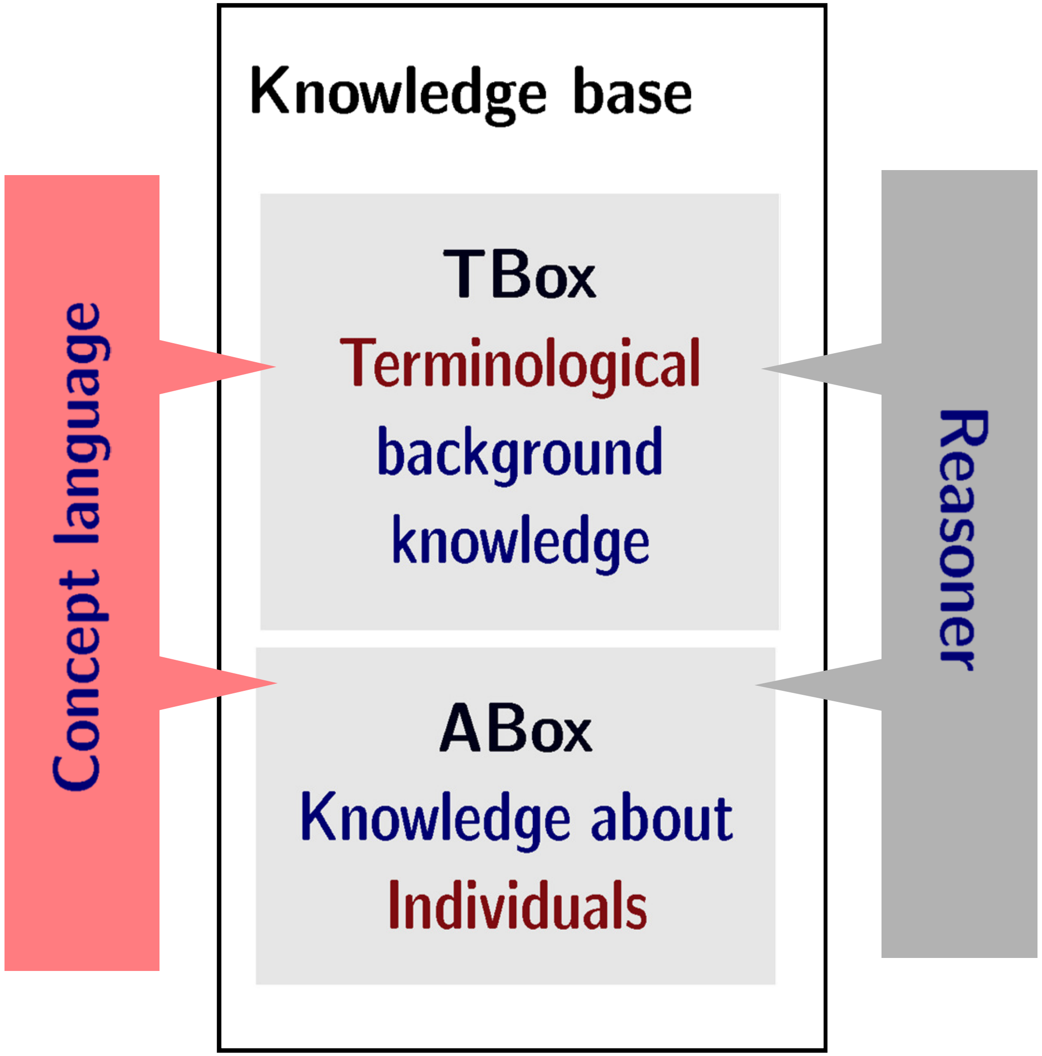
- investigation of reasoning procedures for highly expressive DLs
- investigation of new inferences
- development of ontology editors
- standardization efforts: DAML+OIL, OWL 1.0

Fourth phase – last 6 years

- continuation of investigating increasingly expressive DLs (e.g. $\mathcal{SROIQ}$)
- investigation of DLs with limited expressivity, but good computational properties for a particular inference

  "light weight DLs"
- W3C recommendation: OWL 2 (and 3 profiles)

Overview DL systems



Defining Concepts with DLs

The core part of any DL is the concept language

$$\text{Mammal} \sqcap \exists \text{has-cover}.\text{Fur} \sqcap \forall \text{eats}.\text{Meat}$$

- 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:

$$\text{Cat} \equiv \text{Mammal} \sqcap \exists \text{has-cover}.\text{Fur} \sqcap \forall \text{eats}.\text{Meat}$$

The description logic $\mathcal{ALC}$: syntax

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

role names $r, s, \ldots$ (binary predicates)

$\mathcal{ALC}$ concept constructors:

$\neg C$	(negation)	
$C \sqcap D$	(conjunction)	
$C \sqcup D$	(disjunction)	
$\exists r.C$	(existential restriction)	$\mathcal{EL}$
$\forall r.C$	(value restriction)	

Special concepts:

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

For example: $\neg(A \sqcup \exists r.(\forall s.B \sqcap \neg A))$

Mammal $\sqcap$ $\exists$has-cover.Fur $\sqcap$ $\forall$eats.Meat

Example: $\mathcal{ALC}$-concept descriptions

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

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

Parent:

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

Grandparent:

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

Uncle of happy children:

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

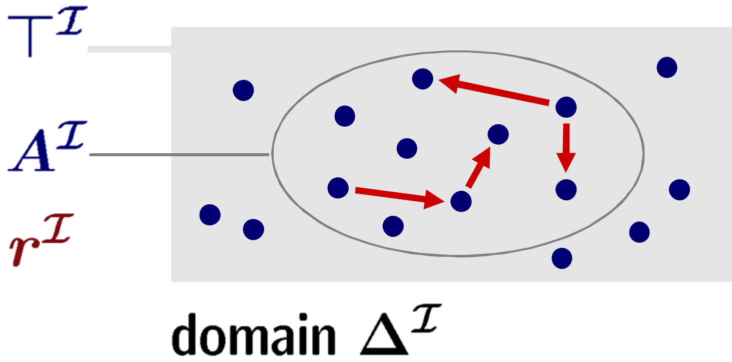
Semantics of named concepts

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:

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

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

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

$$(\exists r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists e : e \in \Delta^{\mathcal{I}} \text{ with } (d,e) \in r^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}}\}$$

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

Reasoning tasks for concepts

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

1. Concept satisfiability

   C is satisfiable if there exists a model of C.

   If unsatisfiable, the concept contains a contradiction.

2. Concept subsumption written $C \sqsubseteq D$

   Does $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ hold for all $\mathcal{I}$?

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

3. Concept equivalence written $C \equiv D$

   Does $C^{\mathcal{I}} = D^{\mathcal{I}}$ hold for all $\mathcal{I}$?

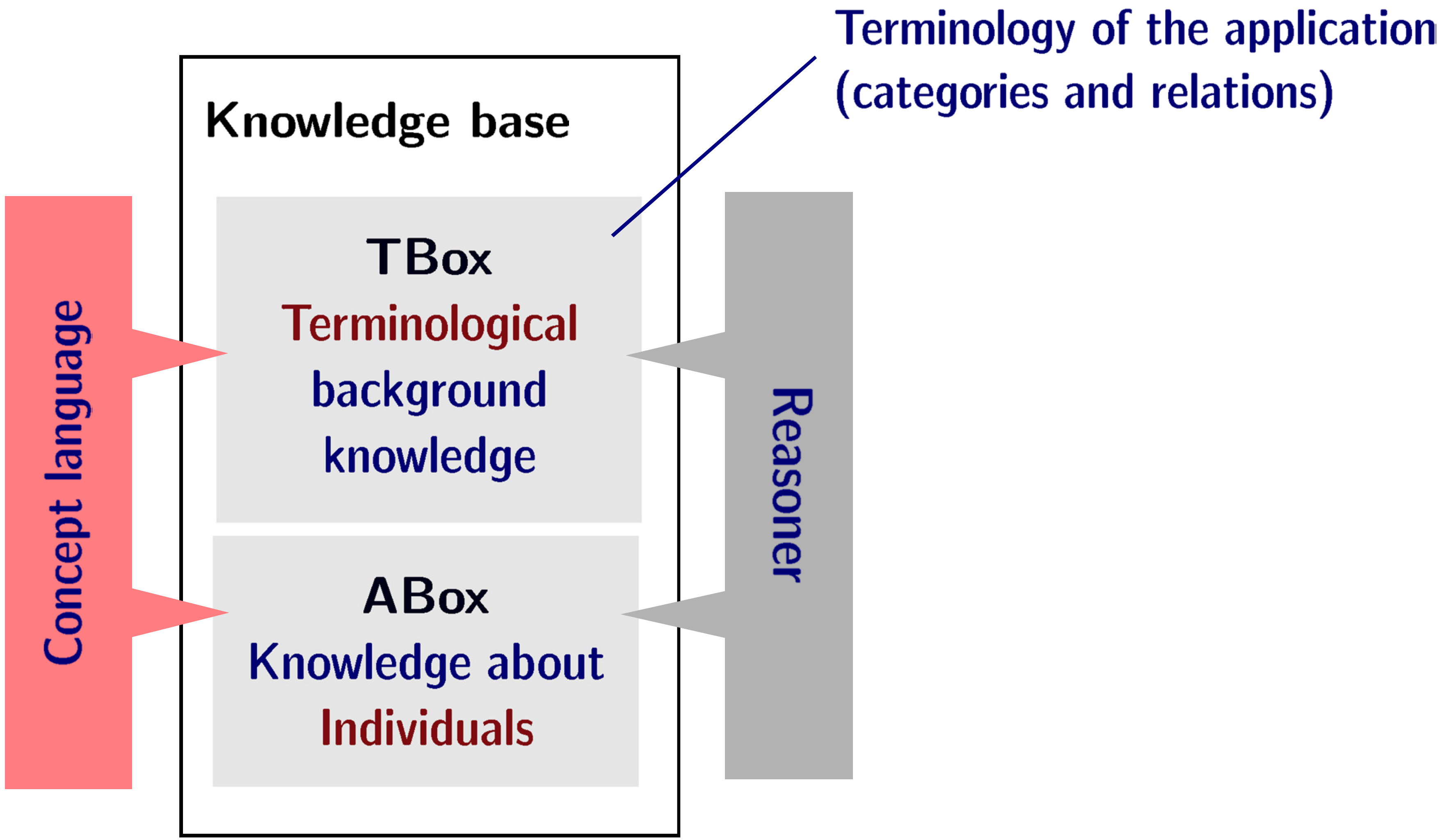
   If $C \equiv D$, then D and C 'say the same'.

Examples

- $\forall$owner.Rich $\sqcap$ $\forall$owner.Famous $\sqsubseteq$ $\forall$owner.(Rich $\sqcap$ Famous)
- $\exists$owner.Rich $\sqcap$ $\exists$owner.Famous $\not\sqsubseteq$ $\exists$owner.(Rich $\sqcap$ Famous)
- $C \sqsubseteq \top$ for all C.
- $\bot \sqsubseteq C$ 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: syntax and semantics

Kinds of concept axioms:

- Primitive concept definition: $A \sqsubseteq D$ $\quad\quad A \in N_C$
- Concept definition: $A \equiv D$ $\quad\quad A \in N_C$
- General concept inclusion (GCI): $C \sqsubseteq D$

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

- General concept equivalence: $C \equiv D$

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

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

$\mathcal{I}$ is a model of a TBox $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all $C \sqsubseteq D \in \mathcal{T}$.

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

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

➻ Unfoldable TBoxes can be conceived as macro definitions.

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 $\mathcal{T}$?

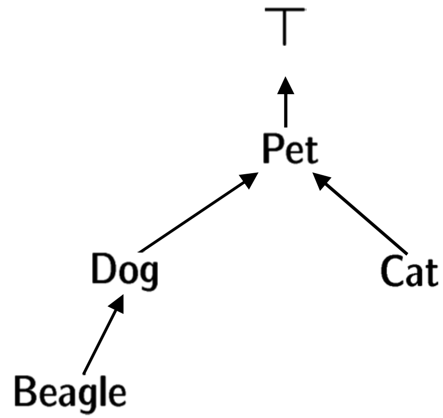
2. **Concept subsumption w.r.t. TBoxes** $(C \sqsubseteq_{\mathcal{T}} D)$

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

3. **Classification of the TBoxes**

   Computation of all subsumption relationships between all named concepts in $\mathcal{T}$.

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



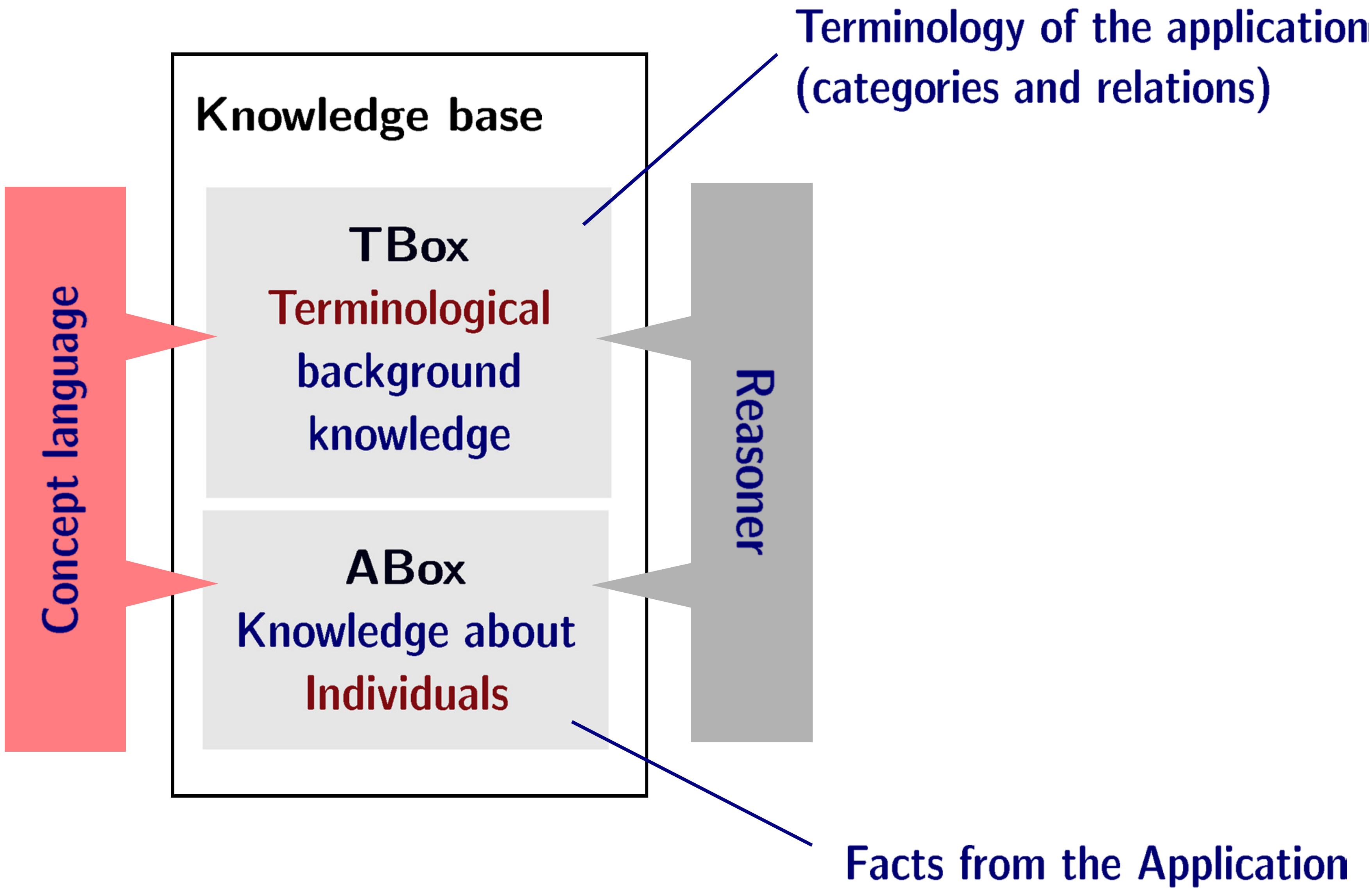
Example for TBox reasoning

TBox

$$
\left\{
\begin{aligned}
\mathsf{Mammal} &\sqsubseteq \mathsf{Animal} \qquad\qquad \mathsf{Salad} \sqsubseteq \mathsf{Plant} \\
\mathsf{Vegetarian} &\equiv \mathsf{Animal} \sqcap \forall \mathsf{eats.Plant} \\
\mathsf{Cat} &\equiv \mathsf{Mammal} \sqcap \exists \mathsf{has\text{-}cover.Fur} \sqcap \forall \mathsf{eats.Meat} \\
\mathsf{VegetarianCat} &\equiv \mathsf{Cat} \sqcap \forall \mathsf{eats.Plants} \ \sqcap \exists \mathsf{eats.Salad} \\
&\cancel{\mathsf{Meat} \sqcap \mathsf{Plant} \sqsubseteq \bot} \\
\mathsf{Salad} &\sqsubseteq \mathsf{Meat}
\end{aligned}
\right\}
$$

1. TBox is satisfiable.
2. VegetarianCat is unsatisfiable w.r.t. TBox.
3. VegetarianCat $\sqsubseteq$ Vegetarian w.r.t. all of theTBoxes.

DL systems are more than a concept language



ABoxes: syntax & semantics

ABox assertions in DL systems are:

- Concept assertions: $C(a)$
- Role assertions: $r(a, b)$

Extend interpretations to individuals:
$a \in N_I$, $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$

Semantics of assertions:

- Concept Assertions: $\mathcal{I}$ satisfies $C(a) \iff a^{\mathcal{I}} \in C^{\mathcal{I}}$
- Role Assertions: $\mathcal{I}$ satisfies $r(a, b) \iff (a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$

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

$\mathcal{I}$ 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.

(unlike models!)

ABox $\mathcal{A}$

Mammal(garfield)

Lasagna(l23)

eats(garfield, l23)

$\forall$eats.Beef(garfield)

Fur(f17)

has-cover(garfield, f17)

likes-most(garfield, garfield)

Assertional Reasoning Services

Reasoning tasks for ABoxes:

1. **ABox consistency**

   Given: $\mathcal{A}$ and $\mathcal{T}$. Do they have a common model?

2. **Instance checking**

   Given: $\mathcal{A}$, $\mathcal{T}$, individual a, and concept C

   Does $a^{\mathcal{I}} \in C^{\mathcal{I}}$ hold in all models of $\mathcal{A}$ and $\mathcal{T}$?

3. **ABox realization**

   Given $\mathcal{A}$ and $\mathcal{T}$.

   Compute for each individual a in $\mathcal{A}$:

   the named concepts in $\mathcal{T}$ of which a is an instance of.

Example for ABox Reasoning

ABox is a partial description of the world.

ABox

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

$\mathsf{Fur}(\mathsf{f17})$

$\mathsf{Lasagna}(\mathsf{l23})$

$\mathsf{has\text{-}cover}(\mathsf{garfield}, \mathsf{f17})$

$\cancel{\mathsf{eats}(\mathsf{garfield}, \mathsf{l23})}$

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

$\forall \mathsf{eats}.\mathsf{Beef}(\mathsf{garfield})$

TBox

$$\mathsf{Cat} \equiv \mathsf{Mammal} \sqcap \exists \mathsf{has\text{-}cover}.\mathsf{Fur} \sqcap \forall \mathsf{eats}.\mathsf{Meat}$$

$$\mathsf{Meat} \equiv \mathsf{Beef} \sqcup \mathsf{Chicken}$$

$$\mathsf{Lasagna} \sqcap \mathsf{Beef} \sqsubseteq \bot$$

1. ABox is inconsistent w.r.t. TBox.
2. garfield is an instance of Cat

Relation of DLs to other logics

Description Logics and First-order Logic

Basic correspondence:

concept names A	$\Longleftrightarrow$	unary predicates P_A
role names r	$\Longleftrightarrow$	binary predicates P_r
concepts	$\Longleftrightarrow$	formulas with one free variable
individuals	$\Longleftrightarrow$	constants c_a

Translation of concept descriptions into First-order Logic

$$
\begin{aligned}
\varphi^x(A) &= P_A(x) \\
\varphi^x(\neg C) &= \neg\varphi^x(C) \\
\varphi^x(C \sqcap D) &= \varphi^x(C) \land \varphi^x(D) \\
\varphi^x(C \sqcup D) &= \varphi^x(C) \lor \varphi^x(D) \\
\varphi^x(\exists r.C) &= \exists y.P_r(x,y) \land \varphi^y(C) \\
\varphi^x(\forall r.C) &= \forall y.P_r(x,y) \to \varphi^y(C)
\end{aligned}
$$

φ^y: x and y exchanged

Note: - two variables suffice (no "=", no constants, no function symbols)

- not all DLs are purely first-order (transitive closure, etc.)

Translation of TBoxes and ABoxes into FOL

TBoxes:

Let C be a concept and $\mathcal{T}$ a (general or unfoldable) TBox.

$$\varphi(\mathcal{T}) = \forall x. \bigwedge_{D \sqsubseteq E \in \mathcal{T}} \varphi^x(D) \rightarrow \varphi^x(E)$$

ABoxes:

individual names a $\Longleftrightarrow$ constants c_a

$$\varphi(C(a)) = \varphi^x(C)[c_a]$$

$$\varphi(r(a,b)) = P_r(c_a, c_b)$$

$$\varphi(\mathcal{A}) = \bigwedge_{\beta \in \mathcal{A}} \varphi(\beta)$$

DLs beyond $\mathcal{ALC}$

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 $(\leq\ n\ r\ C),\ (\geq\ n\ r\ C)$

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

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

Example:

Car $\sqcap$ $(\geq 5$ has-seat$)$ $\sqcap$ $(\leq 5$ has-seat$)$
$\sqcap$ $(\geq 1$ has-seat Drivers-seat$)$ $\sqcap$ $(\leq 1$ 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.

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

- Individuals are distinct.

(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 operators

$\boldsymbol{r}^+$ transitive role $(\boldsymbol{r}^+)^{\mathcal{I}} = \{(\boldsymbol{x}, \boldsymbol{z}) \mid$
$(\boldsymbol{x}, \boldsymbol{y}) \in \boldsymbol{r}^{\mathcal{I}}, (\boldsymbol{y}, \boldsymbol{z}) \in \boldsymbol{r}^{\mathcal{I}} \Rightarrow (\boldsymbol{x}, \boldsymbol{z}) \in \boldsymbol{r}^{\mathcal{I}}\}$

e.g. has-ancestor

$\boldsymbol{r}^-$ inverse role $(\boldsymbol{r}^-)^{\mathcal{I}} = \{(\boldsymbol{y}, \boldsymbol{x}) \mid (\boldsymbol{x}, \boldsymbol{y}) \in \boldsymbol{r}^{\mathcal{I}}\}$

e.g. (has-parent)$^-$ = has-child

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
- $\mathcal{R}$: complex Role inclusions

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

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: $\mathcal{SROIQ}$
  - 2 profiles that correspond to light-weight DLs

The $\mathcal{EL}$ family

Prominent members:

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

$\mathcal{EL}^+$ extends $\mathcal{EL}$ by: complex role inclusions: $r \circ s \sqsubseteq t$.

$\mathcal{EL}^{++}$ extends $\mathcal{EL}^+$ by:

- $\bot$
- nominals
- corresponds to OWL 2 EL profile
- allows for efficient reasoning

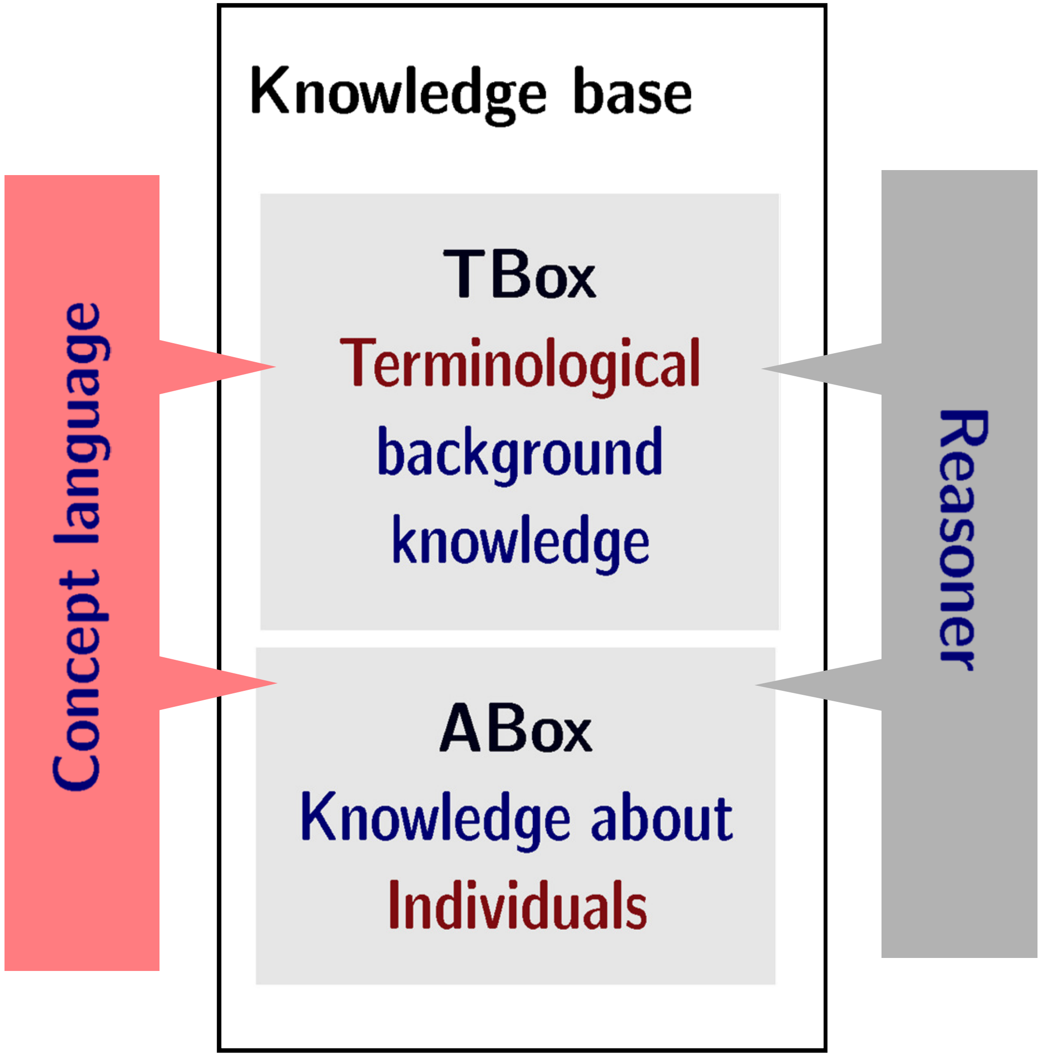
Typically, used with general TBoxes!

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
- allow only for fairly light-weight TBoxes, but can express the basic constructs of ER or UML diagramms

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

Overview DL systems



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. they should be:

- terminating, — You get always an answer.
- sound, — Every positive answer is correct.
- complete. — Every negative answer is correct.

➽ Prerequisit for safe and reliable applications!

Reduction of inferences

Many standard reasoning services can be reduced to satisfiability.
(If negation is present in the DL!)

Use the reduction and implement one reasoning method!

- Equivalence $\Longleftrightarrow$ Satisfiability

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

- Subsumption $\Longleftrightarrow$ Satisfiability

  $C \sqsubseteq_{\mathcal{T}} D$ iff $C \sqcap \neg D$ unsatisfiable w.r.t. $\mathcal{T}$

  $C \not\sqsubseteq_{\mathcal{T}} \bot$ if C is satisfiable w.r.t. $\mathcal{T}$ unsatisfiable w.r.t. $\mathcal{T}$

Reduction of inferences

Many standard reasoning services can be reduced to satisfiability.
(If negation is present in the DL!)

Use the reduction and implement one reasoning method!

- Instance checking $\iff$ ABox consistency

  a is instance of C w.r.t. $(\mathcal{T}, \mathcal{A})$ iff $(\mathcal{T},\ \mathcal{A} \cup \{\neg C(a)\})$ is inconsistent

- Satisfiability $\iff$ ABox consistency

  C is satisfiable w.r.t. $\mathcal{T}$ iff $(\mathcal{T}, \{C(a)\})$ is consistent

Use the reduction

Reformulate a ...	as an ABox consistency check
satisfiability test: sat(C)?	Consistent: $(\mathcal{T}, \{C(a)\})$?

Implement consistency test!

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

We consider: satisfiability of a concept w.r.t. a TBox.

Main steps:

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

Idea: get rid of the unfoldable TBox in a preprocessing step.

Naive approach for expansion:

Let C 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!

$$\mathcal{T} = \{ \quad \begin{aligned} A_0 &\equiv \forall r.A_1 \sqcap \forall s.A_1 \\ A_1 &\equiv \forall r.A_2 \sqcap \forall s.A_2 \\ &\vdots \\ A_{k-1} &\equiv \forall r.A_k \sqcap \forall s.A_k \end{aligned} \quad \}$$

Negation Normal Form

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

Transformation rules:

$$
\begin{array}{rcl}
\neg\neg C & \rightsquigarrow & C \\[1em]
\neg(C \sqcap D) & \rightsquigarrow & \neg C \sqcup \neg D \\
\neg(C \sqcup D) & \rightsquigarrow & \neg C \sqcap \neg D \\[1em]
\neg(\exists r.C) & \rightsquigarrow & \forall r.\neg C \\
\neg(\forall r.C) & \rightsquigarrow & \exists r.\neg C
\end{array}
$$

Tableau Algorithm: Idea

Try to construct a model for the input concept C_0 as follows:
(C_0: expanded and in NNF)

- Represent potential models by proof ABoxes
- To decide satisfiability of C_0,
  start with one initial proof ABox $\mathcal{A}_0$
- Repeatedly apply tableau rules
  and check for obvious contradictions
- Return 'satisfiable' iff a complete and contradiction-free
  proof ABox was found

  (I.e. if all proof ABoxes contain a contradiction,
  return 'not satisfiable')

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 (no more rules applicable)

or

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

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 $C(y) \notin \mathcal{A}$	$\mathcal{A}' := \mathcal{A} \cup \{C(y)\}$

Algorithm is a decision procedure

Lemma

1. If the algorithm returns “satisfiable”,
   then the input concept has a model.
2. If the algorithm returns “not satisfiable”,
   then the input concept has no model.
3. The algorithm terminates on any input

Corollary

$\mathcal{ALC}$-concept satisfiability and subsumption are decidable

Soundness and completeness

Soundness of the procedure:

is shown by local correctness of each tableau rule.

Local correctness:

Let $\mathcal{S}'$ be obtained from $\mathcal{S}$ by the application of a tableau rule.

Then $\mathcal{S}$ is consistent iff $\mathcal{S}'$ is consistent.

Completeness of the procedure:

Directly follows from the definition of a clash.

Termination—some technical notions

Role depth of concepts $d(C)$:

$$
\begin{aligned}
d(A) &= 0 \qquad\qquad A \in N_C \\
d(\neg C) &= d(C) \\
d(C \sqcap D) &= d(C \sqcup D) = \max\{d(C), d(D)\} \\
d(\exists r.C) &= d(\forall r.C) = d(C) + 1
\end{aligned}
$$

Maximal nesting of quantifiers in a concept description.

Termination—some technical notions

sub-concept descriptions of concepts $sub(C)$:

$C \in sub(C)$

$C = \neg D$, then $D \in sub(C)$

$C = C_1 \sqcap C_2$ or $C = C_1 \sqcup C_2$, then $C_1, C_2 \in sub(C)$

$C = \exists r.D$ or $C = \forall r.D$, then $D \in sub(C)$

sub-concept descriptions of ABoxes $sub(\mathcal{A})$:

$$sub(\mathcal{A}) := \bigcup_{C(a) \in \mathcal{A}} sub(C)$$

Termination

The algorithm terminates since:

1. depth of the proof ABox bounded by $d(C_0)$.
2. for each individual, at most $\#sub(C_0)$ successors are generated
3. each individual has at most $\#sub(C_0)$ concept assertions
4. concepts are never deleted from node labels

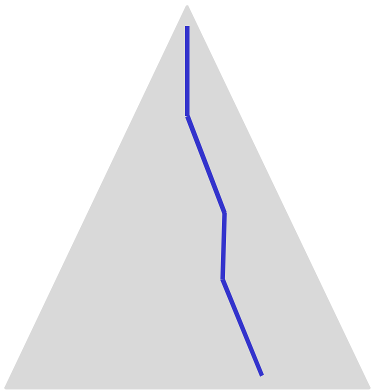
Complexity for reasoning with unfoldable TBoxes

Complexity of unfolding: exponential

Complexity of transformation into NNF: linear

Complexity of application of tableau rules: polynomial space

$\mathcal{A}_0$ $\mathcal{A}_1$ $\cdots$ $\mathcal{A}_{\#\mathsf{sub}(C_0)}$



- all ABoxes need to be considered, but only one at a time
- the whole tree may be generated, but only one path needs to be stored

Tableau algorithm for general TBoxes

- simple expansion does not work in the presence of GCIs:
  - replace a name by which part of the TBox?
  - cyclic axioms: termination?

- Applying the GCIs like rules does not work either!

$$\exists r.(C \sqcap \exists s.D) \sqsubseteq \neg E \sqcup \exists r.D$$

  'Precondition' may never appear at relevant element

- Recall: GCIs hold at every point in the model
  $\longrightarrow$ new tableau rule for GCIs needed

Tableau algorithm for general TBoxes

Tableau rule for 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. Assert C_{GCI} for every individual: new tableau rule

   $\longrightarrow_{\top \sqsubseteq C_{GCI}}$: 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 $C_{GCI} = \neg B \sqcup \exists r.B$

x_0 • $B,\ \neg B \sqcup \exists r.B$

$\exists r.B \quad \longrightarrow_{\exists}$

$\downarrow$

x_1 • $B,\ \neg B \sqcup \exists r.B$

$\exists r.B$

$\downarrow$

x_2 • $B,\ \neg B \sqcup \exists r.B$

$\exists r.B$

$\vdots$

Remedy:

Block of application of $\longrightarrow_{\exists}$

Ancestor blocking

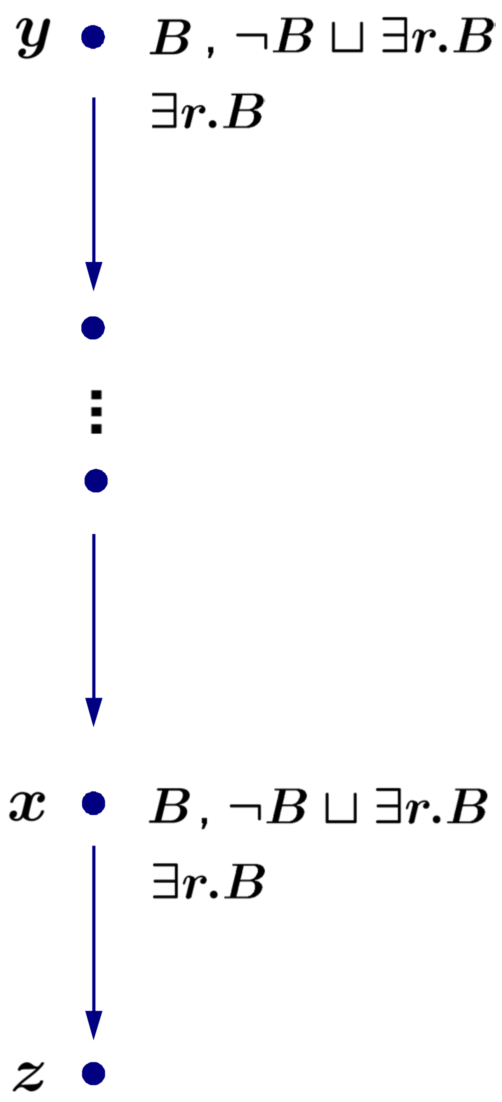
An individual x is directly blocked
by an individual y, iff:

- there is a path from y to x in $\mathcal{A}$
- x was generated by $\longrightarrow_\exists$ after y
  'y is older than x.'
- $\{C \mid C(x) \in \mathcal{A}\} \subseteq \{D \mid D(y) \in \mathcal{A}\}$

An individual x is indirectly blocked if:

- there is a path from y to x in $\mathcal{A}$
- y is directly blocked

An individual x is blocked
if it is blocked or indirectly blocked.



Adaptations to blocking

Replace the exists rule $\longrightarrow_{\exists}$ by a exists rule with blocking $\longrightarrow_{\exists\Box}$:

	Precondition	Replace $\mathcal{A}$ by:
$\longrightarrow_{\exists\Box}$	$(\exists r.C)(x) \in \mathcal{A}$, and x is not (indirectly) blocked but no z in $\mathcal{A}$ s.t. $\{r(x,z), C(z)\} \subseteq \mathcal{A}$	$\mathcal{A}' := \mathcal{A} \cup \{r(x,z), C(z)\}$

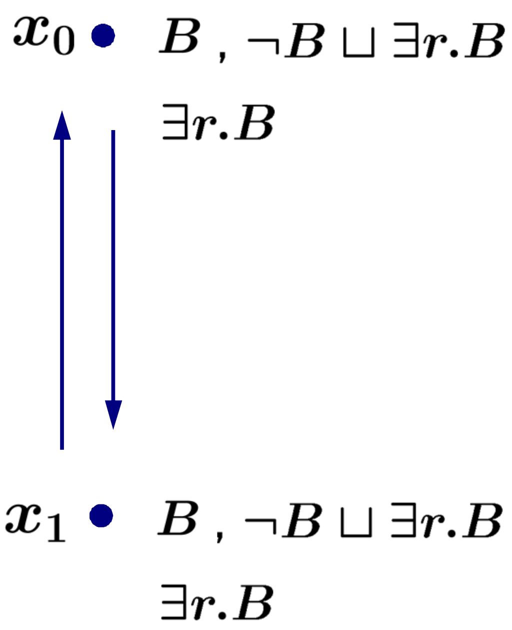
Have we obtained a model?

Some role-successors are missing in the 'blocked' ABox!

Build model w.r.t. blocking:

How to obtain a model for:
$\mathcal{T} = \{B \sqsubseteq \exists r.B\}$?

Introduce 'back links'.



Soundness and completeness

Soundness of the procedure:

is shown by local correctness of each tableau rule.

Local correctness:

Let $\mathcal{S}'$ be obtained from $\mathcal{S}$ by the application of a tableau rule.

Then $\mathcal{S}$ is consistent iff $\mathcal{S}'$ is consistent.

Completeness of the procedure:

Directly follows from the definition of a clash.

Termination

The algorithm terminates since:

1. depth of the proof ABox bounded:
   - #individuals in $\mathcal{A}$: finite
   - #'new' individuals directly reachable from an 'old individual': finite
   - #'new' individuals reachable from a 'new individual': finite (bound by blocking condition)
2. each individual has at most $\#sub(C_{GCI}) + \#sub(\mathcal{A})$ successors
3. each individual has at most $\#sub(C_{GCI}) + \#sub(\mathcal{A})$ concept assertions
4. concepts are never deleted from node labels

Tableau method for DLs

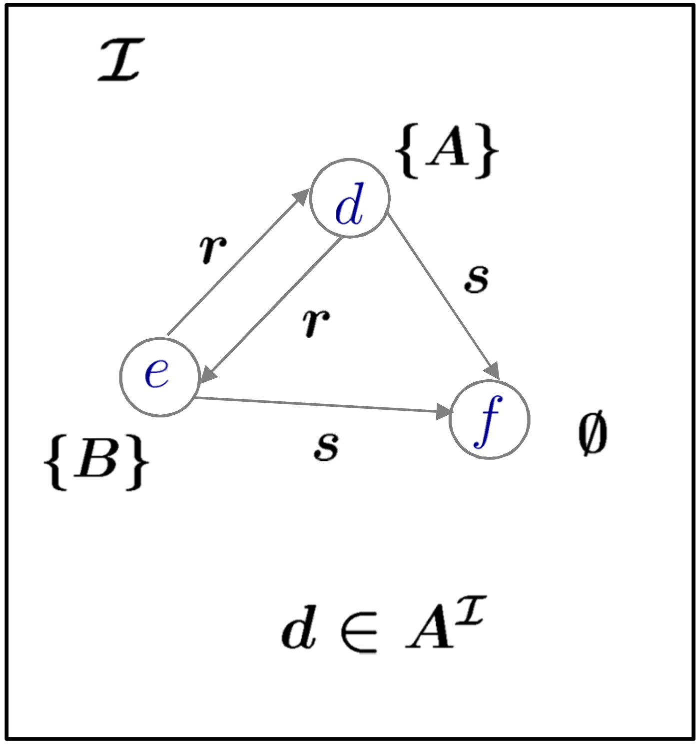
The tableaux algorithm

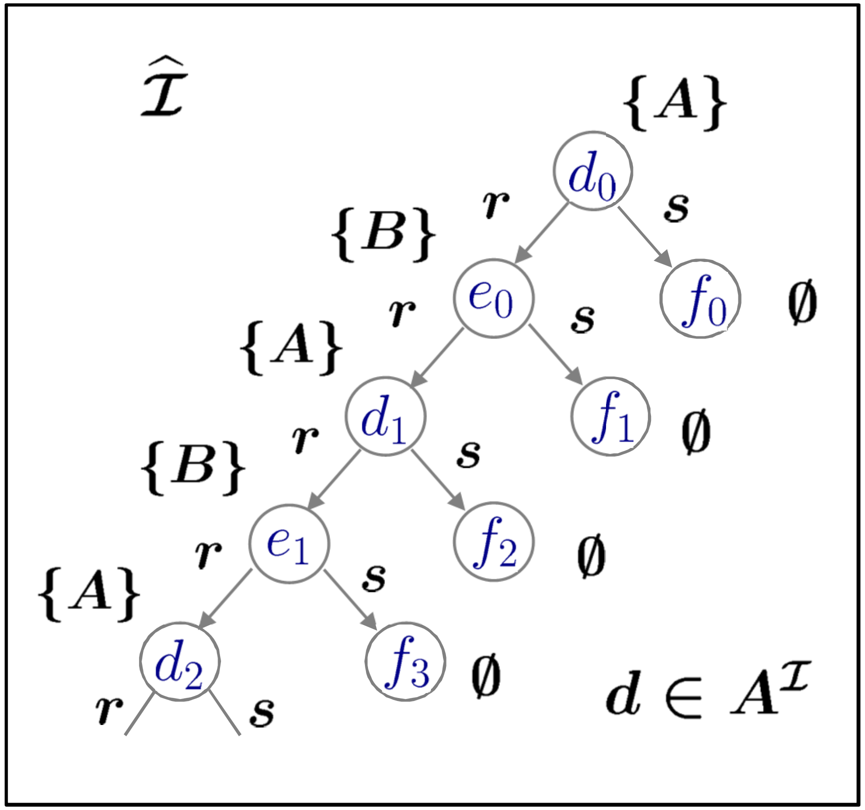
- is implemented in reasoner systems for expressive DLs

  — in particular in the reasoner for OWL 2

- requires optimizations to yield systems
  with acceptable running times

Tree-shaped models (for $\mathcal{ALC}$)





model of:

$A \sqsubseteq \exists r.B$

$B \sqsubseteq \exists r.A$

$A \sqcup B \sqsubseteq \exists s.\top$

Starting with a given node, the graph can be unraveled into a tree without 'changing membership' in concepts.

Tree model property of $\mathcal{ALC}$

Let $\mathcal{T}$ be a TBox and C a concept description.

The interpretation $\mathcal{I}$ is a tree model of C w.r.t. $\mathcal{T}$ if

- $\mathcal{I}$ is a model of $\mathcal{T}$ and
- the graph $(\Delta^{\mathcal{I}}, \bigcup_{r \in N_R} r^{\mathcal{I}})$ is a tree whose root belongs to $C^{\mathcal{I}}$.

Theorem:

$\mathcal{ALC}$ has the tree model property.

i.e., if $\mathcal{T}$: $\mathcal{ALC}$-TBox and C: $\mathcal{ALC}$-concept description such that C is satisfiable w.r.t. $\mathcal{T}$, then C has a tree model w.r.t. $\mathcal{T}$.

No tree model property for $\mathcal{ALCO}$

Theorem:

$\mathcal{ALCO}$ does not have the tree model property.

Proof:

The concept $\{a\}$ does not have a tree model w.r.t. $\{\{a\} \sqsubseteq \exists r.\{a\}\}$.

Finite model property of $\mathcal{ALC}$

Let $\mathcal{T}$ be a TBox and C a concept description.

The interpretation $\mathcal{I}$ is a finite model of C w.r.t. $\mathcal{T}$ iff

- $\mathcal{I}$ is a model of $\mathcal{T}$ and
- $C^{\mathcal{I}} \neq \emptyset$, and $\Delta^{\mathcal{I}}$ is finite.

Theorem:

$\mathcal{ALC}$ has the finite model property.

i.e., if $\mathcal{T}$: $\mathcal{ALC}$-TBox and C: $\mathcal{ALC}$-concept description such that C is satisfiable w.r.t. $\mathcal{T}$, then C has a finite model w.r.t. $\mathcal{T}$.