Description Logics

Franz Baader
Theoretical Computer Science
TU Dresden
Germany

1. Motivation and introduction to Description Logics
2. Tableau-based reasoning procedures
3. Automata-based reasoning procedures
4. Complexity of reasoning in Description Logics
5. Reasoning in inexpressive Description Logics

© Franz Baader

Goal

find DLs for which reasoning is tractable,
i.e, which have polynomial-time decision procedures

- Tractable DLs cannot allow for all Boolean operators:
  satisfiability in propositional logic is already NP-complete
- Conjunction ($\sqcap$) is indispensable:
  otherwise one cannot require several properties simultaneously
- Negation plus conjunction is propositionally complete:
  full negation must be disallowed
- No DL without roles:
  either value or existential restrictions should be present

Two minimal DLs satisfying these requirements:

- $\mathcal{FL}_0$: conjunction ($\sqcap$) and value restrictions ($\forall r.C$)
- $\mathcal{EL}$: conjunction ($\sqcap$) and existential restrictions ($\exists r.C$)

© Franz Baader

$\mathcal{FL}_0$

conjunction and value restrictions

Satisfiability: is trivial

every $\mathcal{FL}_0$-concept description is satisfiable — *just interpret all concept names as the whole domain*

Subsumption: is the interesting inference problem for $\mathcal{FL}_0$-concept descriptions

$$\forall r.(\forall s.B \sqcap \forall s.\forall r.A) \sqcap \forall r.(A \sqcap B) \quad \sqsubseteq \quad \forall r.(A \sqcap \forall s.(B \sqcap \forall r.A))$$

Structural subsumption algorithm:

1. Normalize the concept descriptions
2. Compare the structure of the normalized descriptions

© Franz Baader

Normalization

of $\mathcal{FL}_0$-concept descriptions
proceeds in several steps

Step 1: push value restrictions over conjunction

$$\forall r.(C \sqcap D) \longrightarrow \forall r.C \sqcap \forall r.D$$ *equivalence preserving*

$$\begin{aligned}\forall r.(\forall s.B \sqcap \forall s.\forall r.A) \sqcap \forall r.(A \sqcap B) &\rightarrow \forall r.\forall s.B \sqcap \forall r.\forall s.\forall r.A \sqcap \forall r.(A \sqcap B)\\ &\rightarrow \forall r.\forall s.B \sqcap \forall r.\forall s.\forall r.A \sqcap \forall r.A \sqcap \forall r.B\end{aligned}$$

$$\begin{aligned}\forall r.(A \sqcap \forall s.(B \sqcap \forall r.A)) &\rightarrow \forall r.(A \sqcap \forall s.B \sqcap \forall s.\forall r.A)\\ &\rightarrow \forall r.A \sqcap \forall r.\forall s.B \sqcap \forall r.\forall s.\forall r.A\end{aligned}$$

Normal form obtained by apply this rule exhaustively:

- conjunction of concept descriptions of the form $\forall r_1.\cdots\forall r_n.A$
  where $n \geq 0$ and $A \in N_C$
- can be computed in polynomial time

© Franz Baader

Normalization

of $\mathcal{FL}_0$-concept descriptions
proceeds in several steps

Step 2: use words over N_R

$$\forall r_1.\cdots\forall r_n.A \longrightarrow \forall r_1\ldots r_n.A$$

where $w = r_1\ldots r_n$ is viewed as a word over the alphabet N_R $\quad n = 0$: $w = \varepsilon$

$$\forall r.\forall s.B \sqcap \forall r.\forall s.\forall r.A \sqcap \forall r.A \sqcap \forall r.B \longrightarrow \forall rs.B \sqcap \forall rsr.A \sqcap \forall r.A \sqcap \forall r.B$$

$$\forall r.A \sqcap \forall r.\forall s.B \sqcap \forall r.\forall s.\forall r.A \longrightarrow \forall r.A \sqcap \forall rs.B \sqcap \forall rsr.A$$

Step 3: collect words in front of the same name in a finite language

$$\forall w_1.A \sqcap \ldots \sqcap \forall w_k.A \longrightarrow \forall\{w_1,\ldots,w_k\}.A$$

$$\forall rs.B \sqcap \forall rsr.A \sqcap \forall r.A \sqcap \forall r.B \longrightarrow \forall\{r,rsr\}.A \sqcap \forall\{r,rs\}.B$$

$$\forall r.A \sqcap \forall rs.B \sqcap \forall rsr.A \longrightarrow \forall\{r,rsr\}.A \sqcap \forall\{rs\}.B$$

Dresden

© Franz Baader

Normal form

of $\mathcal{FL}_0$-concept descriptions

$$\text{NF}(C) = \forall L_1.A_1 \sqcap \ldots \sqcap \forall L_m.A_m$$

where $A_1,\ldots,A_m$ are concept names and
$L_1,\ldots,L_m$ are finite languages over N_R

Characterization of subsumption:

Let $\text{NF}(C) = \forall L_1.A_1 \sqcap \ldots \sqcap \forall L_m.A_m$
and $\text{NF}(D) = \forall K_1.A_1 \sqcap \ldots \sqcap \forall K_m.A_m$.

Then $C \sqsubseteq D$ iff $K_1 \subseteq L_1, \ldots, K_m \subseteq L_m$

can be checked in polynomial time

$$\begin{array}{rcl} \text{NF}(C) & = & \forall\{r,rsr\}.A \sqcap \forall\{r,rs\}.B \\ & & \cup\!| \qquad\quad \cup\!| \\ \text{NF}(D) & = & \forall\{r,rsr\}.A \sqcap \forall\{rs\}.B \end{array} \longrightarrow C \sqsubseteq D$$

Dresden

© Franz Baader

Extension to acyclic TBoxes

Subsumption in $\mathcal{FL}_0$ w.r.t. acyclic TBoxes
corresponds to the
inclusion problem for acyclic finite automata.

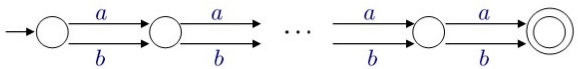
Inclusion problem:

Given two acyclic finite automata $\mathcal{A}, \mathcal{B}$
Question does $L(\mathcal{A}) \subseteq L(\mathcal{B})$ hold

coNP-complete
i.e., non-inclusion is NP-complete

Note:

Acyclic automata define finite sets of words, however, they can do this
exponentially more succinct than the enumeration of all elements

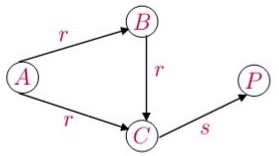


Dresden

© Franz Baader

From acyclic TBoxes to acyclic automata

$$\begin{array}{rcl} A & \equiv & \forall r.B \sqcap \forall r.C \\ B & \equiv & \forall r.C \\ C & \equiv & \forall s.P \end{array}$$



Complications:

$A \equiv \forall r.\forall s.B \sqcap \ldots$	introduce auxiliary states
$A \equiv B \sqcap \ldots$	introduce and then eliminate ε-transitions

Characterization of subsumption:

$A \sqsubseteq_{\mathcal{T}} B$ iff $L(B,P) \subseteq L(A,P)$ for all primitive concepts P

words leading from A to P

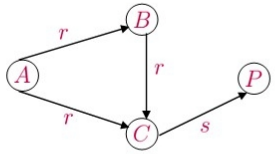
$$L(A,P) = \{rrs, rs\} \supseteq L(B,P) = \{rs\} \longrightarrow A \sqsubseteq B$$

Dresden

© Franz Baader

From acyclic TBoxes to acyclic automata

$$\begin{array}{lcl} A & \equiv & \forall r.B \sqcap \forall r.C \\ B & \equiv & \forall r.C \\ C & \equiv & \forall s.P \end{array}$$



Complications:

- $A \equiv \forall r.\forall s.B \sqcap \ldots$ introduce auxiliary states
- $A \equiv B \sqcap \ldots$ introduce and then eliminate ε-transitions

Characterization of subsumption:

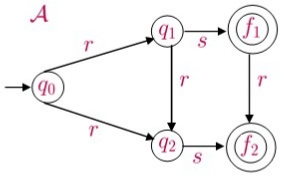
$A \sqsubseteq_{\mathcal{T}} B$ iff $L(B,P) \subseteq L(A,P)$ for all primitive concepts P

The subsumption problem in $\mathcal{FL}_0$ w.r.t. acyclic TBoxes is in coNP.

© Franz Baader

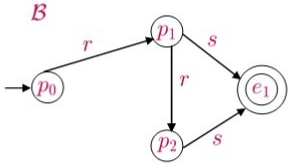
From acyclic automata to acyclic TBoxes

$\mathcal{A}$



$$\mathcal{T}_{\mathcal{A}}: \quad \begin{array}{lcl} Q_0 & \equiv & \forall r.Q_1 \sqcap \forall r.Q_2 \\ Q_1 & \equiv & \forall r.Q_2 \sqcap \forall s.F_1 \\ Q_2 & \equiv & \forall s.F_2 \\ F_1 & \equiv & \forall r.F_2 \sqcap P_{fin} \\ F_2 & \equiv & P_{fin} \end{array}$$

$\mathcal{B}$



$$\mathcal{T}_{\mathcal{B}}: \quad \begin{array}{lcl} P_0 & \equiv & \forall r.P_1 \\ P_1 & \equiv & \forall r.P_2 \sqcap \forall s.E_1 \\ P_2 & \equiv & \forall s.E_1 \\ E_1 & \equiv & P_{fin} \end{array}$$

Characterization of inclusion:

$L(\mathcal{B}) \subseteq L(\mathcal{A})$ iff $Q_0 \sqsubseteq_{\mathcal{T}_{\mathcal{A}} \cup \mathcal{T}_{\mathcal{B}}} P_0$ ⟶ subsumption coNP-hard

© Franz Baader

Complexity of subsumption

	$\mathcal{FL}_0$	$\mathcal{EL}$	
no TBox	polynomial	polynomial	1.
acyclic TBox	coNP-complete	polynomial	
cyclic TBox	PSpace-complete	polynomial	
general TBox	ExpTime-complete	polynomial	2.

© Franz Baader

$\mathcal{EL}$

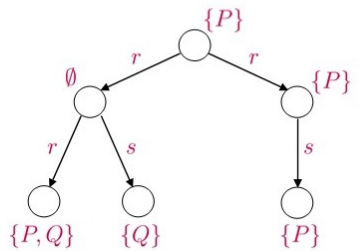
conjunction, existential restrictions, and top concept $\top$

$\mathcal{EL}$-concept descriptions can be represented as description trees:

- nodes labeled with sets of concept names
- edges labeled with role names

$C = P \sqcap \exists r.(\exists r.(P \sqcap Q) \sqcap \exists s.Q) \sqcap \exists r.(P \sqcap \exists s.P)$

T_C:



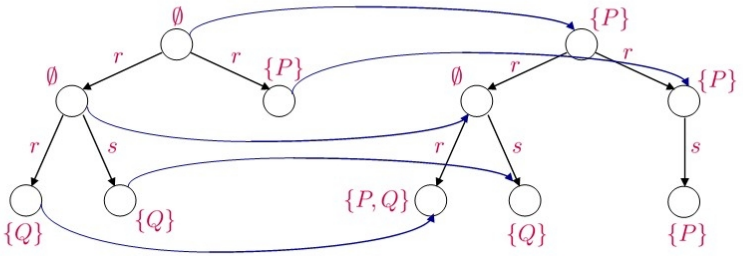
© Franz Baader

Subsumption

corresponds to existence of homomorphism

Let $T_1 = (V_1, E_1, \ell_1)$ and $T_2 = (V_2, E_2, \ell_2)$ be description trees.
The mapping $h : V_1 \to V_2$ is a homomorphism if

- it maps the root of T_1 to the root of T_2;
- $\ell(v) \subseteq \ell(h(v))$ for all $v \in V_1$;
- $(v_1, r, v_2) \in E_1$ implies $(h(v_1), r, h(v_2)) \in E_2$.



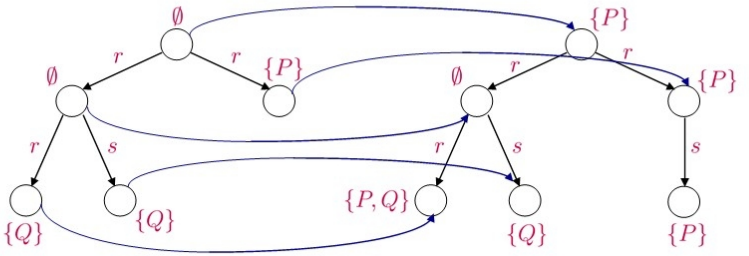
© Franz Baader

Subsumption

corresponds to existence of homomorphism

$C \sqsubseteq D$ iff there is a homomorphism from T_D to T_C.

$\exists r.(\exists r.Q \sqcap \exists s.Q) \sqcap \exists r.P \quad \sqsupseteq \quad P \sqcap \exists r.(\exists r.(P \sqcap Q) \sqcap \exists s.Q) \sqcap \exists r.(P \sqcap \exists s.P)$



© Franz Baader

Existence of homomorphism

For trees, this can be decided in polynomial time.

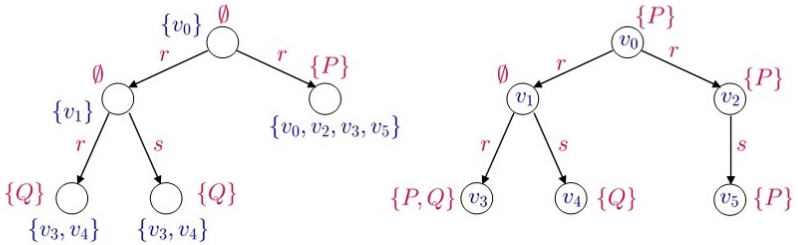
NP-complete for graphs

To decide whether there exists a homomorphism $T_1 \to T_2$,
we compute for every $u \in V_1$ the set

$$H(u) := \{v \in V_2 \mid \text{there is a homomorphism from } T_1|_u \text{ to } T_2|_v\}$$

in a bottom-up manner.

subtree of T_1 with root u



© Franz Baader

Subsumption

in the presence of GCIs

We describe an algorithm that classifies a general TBox $\mathcal{T}$,
i.e., computes all subsumption relationships between the concept names of $\mathcal{T}$.

1. Normalize the general TBox.
2. Translate normalized TBox into a graph.
3. Complete the graph.
4. Read off all subsumption relationships from the completed graph.

© Franz Baader

Normalization

of a set of GCIs

Goal: obtain an equivalent set where all GCIs are of the form

$$A_1 \sqcap A_2 \sqsubseteq B \qquad A \sqsubseteq \exists r.B \qquad \exists r.A \sqsubseteq B$$

where $A, A_1, A_2, B \in N_C \cup \{\top\}$

$\exists s.A \sqcap \exists r.\exists s.A \sqsubseteq A \sqcap \exists r.\top \quad \rightsquigarrow \quad \exists s.A \sqsubseteq B_1, \; B_1 \sqcap \exists r.\exists s.A \sqsubseteq A \sqcap \exists r.\top$

$\rightsquigarrow \quad \exists s.A \sqsubseteq B_1, \; \exists r.\exists s.A \sqsubseteq B_2, \; B_1 \sqcap B_2 \sqsubseteq A \sqcap \exists r.\top$

$\rightsquigarrow \quad \exists s.A \sqsubseteq B_1, \; \exists s.A \sqsubseteq B_3, \; \exists r.B_3 \sqsubseteq B_2, \; B_1 \sqcap B_2 \sqsubseteq A \sqcap \exists r.\top$

$\rightsquigarrow \quad \exists s.A \sqsubseteq B_1, \; \exists s.A \sqsubseteq B_3, \; \exists r.B_3 \sqsubseteq B_2, \; B_1 \sqcap B_2 \sqsubseteq A, \; B_1 \sqcap B_2 \sqsubseteq \exists r.\top$

Dresden

© Franz Baader

Normalization

of a set of GCIs

Goal: obtain an equivalent set where all GCIs are of the form

$$A_1 \sqcap A_2 \sqsubseteq B \qquad A \sqsubseteq \exists r.B \qquad \exists r.A \sqsubseteq B$$

where $A, A_1, A_2, B \in N_C \cup \{\top\}$

Normal form can be computed

- by applying simple equivalence preserving transformation rules
- in linear time if an appropriate strategy is used

Dresden

© Franz Baader

Graph

of a normalized set of GCIs

The classification graph is of the form $(V, V \times V, S, R)$ where

- V is the set of concept names occurring in $\mathcal{T}$; *including* $\top$
- S labels nodes with sets of concept names;
- R labels edges with sets of role names.

Initialization:

- $S(A) := \{A, \top\}$;
- $R(A, B) := \emptyset$.

Invariant:

- $B \in S(A)$ implies $A \sqsubseteq_{\mathcal{T}} B$;
- $r \in R(A, B)$ implies $A \sqsubseteq_{\mathcal{T}} \exists r.B$.

Dresden

© Franz Baader

Completion rules

extend the labels in the graph

R1 $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}$ and $A_1, A_2 \in S(A) \rightsquigarrow$ add B to $S(A)$

R2 $A_1 \sqsubseteq \exists r.B \in \mathcal{T}$ and $A_1 \in S(A) \rightsquigarrow$ add r to $R(A, B)$

R3 $\exists r.B_1 \sqsubseteq A_1 \in \mathcal{T}$ and $B_1 \in S(B)$, $r \in S(A, B) \rightsquigarrow$ add A_1 to $S(A)$

Rule application preserves the invariant:

$B \in S(A)$ implies $A \sqsubseteq_{\mathcal{T}} B$

$r \in R(A, B)$ implies $A \sqsubseteq_{\mathcal{T}} \exists r.B$

R3 $\exists r.B_1 \sqsubseteq A_1 \in \mathcal{T}$, $B_1 \in S(B)$, $r \in S(A, B)$ $\longrightarrow$ $\exists r.B_1 \sqsubseteq_{\mathcal{T}} A_1$, $\exists r.B \sqsubseteq_{\mathcal{T}} \exists r.B_1$, $A \sqsubseteq_{\mathcal{T}} \exists r.B$ $\longrightarrow$ $A \sqsubseteq_{\mathcal{T}} A_1$

Dresden

© Franz Baader

Completion rules

yield a polynomial-time decision procedure

1. Rule application terminates after a polynomial number of steps.
2. If no more rules are applicable, then
$$A \sqsubseteq_{\mathcal{T}} B \quad \text{iff} \quad B \in S(A)$$

1. Termination

- the number of nodes is linear and the number of edges quadratic in the size of $\mathcal{T}$;
- the size of the label sets is linear in the size of $\mathcal{T}$;
- applying one rule needs polynomial time.

at most cubic number of rule applications

Dresden

© Franz Baader

Completion rules

yield a polynomial-time decision procedure

1. Rule application terminates after a polynomial number of steps.
2. If no more rules are applicable, then
$$A \sqsubseteq_{\mathcal{T}} B \quad \text{iff} \quad B \in S(A)$$

2. $A \sqsubseteq_{\mathcal{T}} B \quad \text{iff} \quad B \in S(A)$

"$\Leftarrow$" follows from the fact that the invariants are preserved.

"$\Rightarrow$" Assume that $B \notin S(A)$.

The following is a model of $\mathcal{T}$ that violates the subsumption relationship:

- $\Delta^{\mathcal{I}} := V$;
- $B'^{\mathcal{I}} := \{A' \mid B' \in S(A')\}$ for all concept names B';
- $r^{\mathcal{I}} := \{(A', B') \mid r \in R(A', B')\}$ for all role names r.

Dresden

© Franz Baader

Goal

find DLs for which reasoning is tractable,
i.e, which have polynomial-time decision procedures

$\mathcal{EL}$ is such a DL:

- subsumption in $\mathcal{EL}$ is polynomial
  even in the presence of GCIs;
- this even holds for some extensions of $\mathcal{EL}$;
- $\mathcal{EL}$ is used in several biomedical ontologies (e.g., SNOMED).

Dresden

© Franz Baader