Ontology and Database Systems: Knowledge Representation and Ontologies

Part 6: Reasoning in the $\mathcal{ALC}$ family

Diego Calvanese

Faculty of Computer Science
European Master in Computational Logic

A.Y. 2015/2016

Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science

Part 6

Reasoning in the $\mathcal{ALC}$ family

Outline of Part 6

Outline of Part 6

Outline of Part 6

Recall the definition of $\mathcal{ALC}$ – Concept language

Construct	**Syntax**	**Example**	**Semantics**
atomic concept	A	Doctor	$A^\mathcal{I} \subseteq \Delta^\mathcal{I}$
atomic role	P	hasChild	$P^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$
conjunction	$C_1 \sqcap C_2$	Hum $\sqcap$ Male	$C_1^\mathcal{I} \cap C_2^\mathcal{I}$
value restriction	$\forall R.C$	$\forall$hasChild.Male	$\{o \mid \forall o'. (o,o') \in R^\mathcal{I} \to o' \in C^\mathcal{I}\}$
negation	$\neg C$	$\neg\forall$hasChild.Male	$\Delta^\mathcal{I} \setminus C^\mathcal{I}$

(C_1, C_2 denote arbitrary concepts and R an arbitrary role)

We make also use of the following abbreviations:

Construct	Stands for	
$\bot$	$A \sqcap \neg A$	(for some atomic concept A)
$\top$	$\neg\bot$	
$C_1 \sqcup C_2$	$\neg(\neg C_1 \sqcap \neg C_2)$	
$\exists R.C$	$\neg\forall R.\neg C$	

unibz

$\mathcal{ALC}$ ontology (or knowledge base)

Def.: $\mathcal{ALC}$ ontology

Is a pair $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox:

- The TBox is a set of **inclusion assertions** on $\mathcal{ALC}$ concepts: $C_1 \sqsubseteq C_2$
- The ABox is a set of **membership assertions** on individuals:
  - Membership assertions for concepts: $A(c)$
  - Membership assertions for roles: $P(c_1, c_2)$

Note: We use $C_1 \equiv C_2$ as an abbreviation for $C_1 \sqsubseteq C_2,\ C_2 \sqsubseteq C_1$.

Example

TBox:
$$\begin{array}{rcl}
\text{Father} & \equiv & \text{Human} \sqcap \text{Male} \sqcap \exists\text{hasChild} \\
\text{HappyFather} & \sqsubseteq & \text{Father} \sqcap \forall\text{hasChild.}(\text{Doctor} \sqcup \text{Lawyer} \sqcup \text{HappyPerson}) \\
\text{HappyAnc} & \sqsubseteq & \forall\text{descendant.HappyFather} \\
\text{Teacher} & \sqsubseteq & \neg\text{Doctor} \sqcap \neg\text{Lawyer}
\end{array}$$

ABox: $\text{Teacher}(\texttt{mary}),\quad \text{hasFather}(\texttt{mary}, \texttt{john}),\quad \text{HappyAnc}(\texttt{john})$

From $\mathcal{ALC}$ to First Order Logic

We have seen that $\mathcal{ALC}$ is a well-behaved fragment of function-free First Order Logic with unary and binary predicates only ($\mathsf{FOL}_{\mathsf{bin}}$).

To translate an $\mathcal{ALC}$ TBox to $\mathsf{FOL}_{\mathsf{bin}}$ we proceed as follows:

1. Introduce: a unary predicate $A(x)$ for each atomic concept A
   a binary predicate $P(x,y)$ for each atomic role P
2. Translate complex concepts as follows, using translation functions t_x, one for each variable x:

$$\begin{aligned} t_x(A) &= A(x) & t_x(C \sqcap D) &= t_x(C) \wedge t_x(D) \\ t_x(\neg C) &= \neg t_x(C) & t_x(C \sqcup D) &= t_x(C) \vee t_x(D) \\ t_x(\exists P.C) &= \exists y.\, P(x,y) \wedge t_y(C) & & \\ t_x(\forall P.C) &= \forall y.\, P(x,y) \rightarrow t_y(C) & & \text{(with } y \text{ a new variable)} \end{aligned}$$

3. Translate a TBox $\mathcal{T} = \bigcup_i \{\, C_i \sqsubseteq D_i \,\}$ as the FOL theory:

$$\Gamma_{\mathcal{T}} = \bigcup_i \{\, \forall x.\, t_x(C_i) \rightarrow t_x(D_i) \,\}$$

4. Translate an ABox $\mathcal{A} = \bigcup_i \{\, A_i(c_i) \,\} \cup \bigcup_j \{\, P_j(c'_j, c''_j) \,\}$ as the FOL th.:

$$\Gamma_{\mathcal{A}} = \bigcup_i \{\, A_i(c_i) \,\} \cup \bigcup_j \{\, P_j(c'_j, c''_j) \,\}$$

unibz

From $\mathcal{ALC}$ to First Order Logic – Reasoning

Via the translation to FOL_{bin}, there is a direct correspondence between DL reasoning services and FOL reasoning services:

C is satisfiable	iff	its translation $t_x(C)$ is satisfiable
C is satisfiable w.r.t. $\mathcal{T}$	iff	$\Gamma_{\mathcal{T}} \cup \{ \exists x. t_x(C) \}$ is satisfiable
$\mathcal{T} \models_{\mathcal{ALC}} C \sqsubseteq D$	iff	$\Gamma_{\mathcal{T}} \models_{FOL} \forall x. (t_x(C) \rightarrow t_x(D))$
$C \sqsubseteq D$	iff	$\models_{FOL} t_x(C) \rightarrow t_x(D)$
$\top \sqsubseteq C$	iff	$\models_{FOL} t_x(C)$

(We use $\models_{FOL} \varphi$ to denote that φ is a valid FOL formula.)

From First Order Logic to $\mathcal{ALC}$?

Question

Is it possible to define a transformation $\tau(\cdot)$ from $\mathsf{FOL}_{\mathsf{bin}}$ formulas to $\mathcal{ALC}$ concepts and roles such that the following is true?

$$\models_{FOL} \varphi \qquad \text{implies} \qquad \top \sqsubseteq \tau(\varphi)$$

- If yes, we should specify the transformation $\tau(\cdot)$.
- If not, we should provide a formal proof that $\tau(\cdot)$ does not exist.

unibz

Outline of Part 6

Distinguishability of interpretations

Def.: Distinguishing between models

If $\mathcal{I}$ and $\mathcal{J}$ are two interpretations of a logic $\mathcal{L}$, then we say that $\mathcal{I}$ and $\mathcal{J}$ are **distinguishable in $\mathcal{L}$** if there is a formula φ of the language of $\mathcal{L}$ such that

$$\mathcal{I} \models_{\mathcal{L}} \varphi \qquad \text{and} \qquad \mathcal{J} \not\models_{\mathcal{L}} \varphi$$

Proving non equivalence:

To show that two logics $\mathcal{L}_1$ and $\mathcal{L}_2$ with the same class of interpretations are **not equivalent**, it is enough to show that there are two interpretations $\mathcal{I}$ and $\mathcal{J}$ that are distinguishable in $\mathcal{L}_1$ and not distinguishable in $\mathcal{L}_2$.

Bisimulation

The notion of **bisimulation** in description logics is intended to capture equivalence of objects and their properties.

Def.: Bisimulation

A **bisimulation** $\sim_{\mathcal{B}}$ between two $\mathcal{ALC}$ interpretations $\mathcal{I}$ and $\mathcal{J}$ is a relation in $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ such that, for every pair of objects $o_1 \in \Delta^{\mathcal{I}}$ and $o_2 \in \Delta^{\mathcal{J}}$, if $o_1 \sim_{\mathcal{B}} o_2$ then the following hold:

- for every atomic concept A: $o_1 \in A^{\mathcal{I}}$ if and only if $o_2 \in A^{\mathcal{J}}$ (**local condition**);
- for every atomic role P:
  - for each o_1' with $(o_1, o_1') \in P^{\mathcal{I}}$, there is an o_2' with $(o_2, o_2') \in P^{\mathcal{J}}$ such that $o_1' \sim_{\mathcal{B}} o_2'$ (**forth property**);
  - for each o_2' with $(o_2, o_2') \in P^{\mathcal{J}}$, there is an o_1' with $(o_1, o_1') \in P^{\mathcal{I}}$ such that $o_1' \sim_{\mathcal{B}} o_2'$ (**back property**).

$(\mathcal{I}, o_1) \sim (\mathcal{J}, o_2)$ means that there is a bisimulation $\sim_{\mathcal{B}}$ between $\mathcal{I}$ and $\mathcal{J}$ such that $o_1 \sim_{\mathcal{B}} o_2$.

Bisimulation and $\mathcal{ALC}$

Lemma

$\mathcal{ALC}$ cannot distinguish o_1 in interpretation $\mathcal{I}$ and o_2 in interpretation $\mathcal{J}$ when $(\mathcal{I}, o_1) \sim (\mathcal{J}, o_2)$.
In other words, if $(\mathcal{I}, o_1) \sim (\mathcal{J}, o_2)$, then for every $\mathcal{ALC}$ concept C we have that

$$o_1 \in C^{\mathcal{I}} \qquad \text{if and only if} \qquad o_2 \in C^{\mathcal{J}}$$

Proof.

By induction on the structure of concepts. [Exercise] □

Outline of Part 6

Disjoint union model property of $\mathcal{ALC}$

Def.: Disjoint union model

For two interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ and $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$, the **disjoint union of $\mathcal{I}$ and $\mathcal{J}$** is the interpretation:

$$\mathcal{I} \uplus \mathcal{J} = (\Delta^{\mathcal{I}\uplus\mathcal{J}}, \cdot^{\mathcal{I}\uplus\mathcal{J}})$$

where

- $\Delta^{\mathcal{I}\uplus\mathcal{J}} = \Delta^{\mathcal{I}} \uplus \Delta^{\mathcal{J}}$;
- $A^{\mathcal{I}\uplus\mathcal{J}} = A^{\mathcal{I}} \uplus A^{\mathcal{J}}$, for every atomic concept A;
- $P^{\mathcal{I}\uplus\mathcal{J}} = P^{\mathcal{I}} \uplus P^{\mathcal{J}}$, for every atomic role P.

Exercise

Prove via the bisimulation lemma that, for each pair of $\mathcal{ALC}$ concepts C and D:

$$\text{if } \mathcal{I} \models C \sqsubseteq D \text{ and } \mathcal{J} \models C \sqsubseteq D \quad \text{then} \quad \mathcal{I} \uplus J \models C \sqsubseteq D.$$

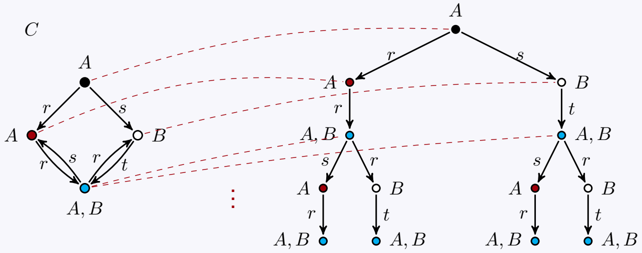
Tree model property of DLs

Theorem

An $\mathcal{ALC}$ concept C is satisfiable w.r.t. a TBox $\mathcal{T}$ if and only if there is a **tree-shaped model** $\mathcal{I}$ of $\mathcal{T}$ and an object o such that $o \in C^{\mathcal{I}}$.

Proof.

The "if" direction is obvious. For the "only-if" direction, we exploit the fact that an interpretation and its unraveling into a tree are bisimilar.



□

Expressive power of $\mathcal{ALC}$

Exercise

Prove, using tree model property, that the FOL$_{\text{bin}}$ formula $\forall x.P(x,x)$ cannot be translated into $\mathcal{ALC}$. In other words, prove that there is no $\mathcal{ALC}$ TBox $\mathcal{T}$ such that

$$\mathcal{I} \models_{\mathcal{ALC}} \mathcal{T} \qquad \text{if and only if} \qquad \mathcal{I} \models_{FOL} \forall x.P(x,x)$$

A consequence of the above fact, and of the fact that $\mathcal{ALC}$ can be expressed in FOL$_{\text{bin}}$ is that:

Expressive power of $\mathcal{ALC}$

$\mathcal{ALC}$ is **strictly less expressive** than FOL$_{\text{bin}}$.

From FOL$_{\text{bin}}$ to $\mathcal{ALC}$

Def.: Bisimulation invariance

A FOL unary formula $\varphi(x)$ is **invariant for bisimulation** if for all interpretations $\mathcal{I}$ and $\mathcal{J}$, and all objects o_1 and o_2 such that $(\mathcal{I}, o_1) \sim (\mathcal{J}, o_2)$

$$\mathcal{I}, [x \to o_1] \models \varphi(x) \qquad \text{if and only if} \qquad \mathcal{J}, [x \to o_2] \models \varphi(x)$$

Theorem ([Benthem 1976, 1983])

The following are equivalent for all unary FOL$_{\text{bin}}$ $\varphi(x)$:

- $\varphi(x)$ is invariant for bisimulation.
- $\varphi(x)$ is equivalent to the standard translation of an $\mathcal{ALC}$ concept.

Outline of Part 6

Outline of Part 6

Tableau algorithms for DLs

Tableau-based techniques

They try to decide the satisfiability of a formula (or theory) by using **rules** to construct (a representation of) a model.

- They have been used in FOL and modal logics for many years.
- For DLs, they have been extensively explored since the late 1990s [Baader and Sattler 2001].
- They are considered well suited for implementation.
- In fact, many of the most successful DL reasoners implement tableau techniques or variations of them.
  E.g.: RACER, FaCT++, Pellet, Hermit, etc.

A tableau algorithm for $\mathcal{ALC}$ concepts – Overview

We describe an algorithm that decides concept satisfiability in $\mathcal{ALC}$.

For an input $\mathcal{ALC}$ concept C_0, it tries to build a graph representation of a model $\mathcal{I}$ of C_0:

- It works with **labeled, tree-shaped graphs**:
  - the nodes are labeled with concepts, and
  - the edges are labeled with roles.
- At each moment, the algorithm stores a **set $\mathcal{G}$ of labeled graphs**.
- It starts with the set $\mathcal{G}_0$ containing one graph with just one node labeled C_0.
- It uses **tableau rules** corresponding to the constructors, **to infer a new set $\mathcal{G}'$** of graphs **from the previous set $\mathcal{G}$**.
- Intuitively, each new graph makes explicit some constraint resulting from C_0 that was still implicit in the previous step.

A tableau algorithm for $\mathcal{ALC}$ concepts – Overview (cont'd)

- Each **rule**, when applied to a graph G in the current set $\mathcal{G}$, may:
  - add new nodes to G, or
  - add new labels to the existing nodes of G.
- The rules are **non-deterministic**, in general, i.e., they may be applied in more than one way, resulting in different possible graphs.
- If a graph contains a **clash**, i.e., an explicit contradiction, it is dropped and not expanded further.
- When no rule can be applied anymore to a graph, the graph is called **complete**.
- The algorithm continues
  - until some graph G in the current set is complete and clash-free, or
  - until all graphs contain a clash.
- A complete and clash-free graph G represents a model $\mathcal{I}$ of C_0.

unibz

Negation Normal Form

Def.: Negation normal form

A concept C is in **negation normal form (NNF)** if the '$\neg$' operator is applied only to atomic concepts. Moreover, C does not contain $\top$ or $\bot$.

- Every concept C can be transformed into a concept $\textsc{nnf}(C)$ in NNF, by **pushing inside the '$\neg$' operator**, using the following equivalences:

$$\begin{array}{rcl} \neg(C \sqcap D) & \equiv & \neg C \sqcup \neg D \\ \neg(C \sqcup D) & \equiv & \neg C \sqcap \neg D \\ \neg(\neg C) & \equiv & C \end{array} \qquad \begin{array}{rcl} \neg\forall P.C & \equiv & \exists P.\neg C \\ \neg\exists P.C & \equiv & \forall P.\neg C \end{array}$$

- The translation process terminates in linear time.
- C and $\textsc{nnf}(C)$ are **equivalent**, i.e., $C^{\mathcal{I}} = \textsc{nnf}(C)^{\mathcal{I}}$, for every interpretation $\mathcal{I}$.

Completion graphs

Consider a given concept C_0 in NNF.

We denote by $\textsc{sub}(C_0)$ the set consisting of all subconcepts of C_0 and their negations (in NNF).

Def.: Completion graph

A **completion graph** for C_0 is a labeled graph $\langle V, E, \mathcal{L}\rangle$, where

- V is a finite set of nodes,
- $E \subseteq V \times V$ is the set of edges, and
- $\mathcal{L}$ is a **labeling function** that maps:
  - each node $v \in V$ to a set of concepts $\mathcal{L}(v) \subseteq \textsc{sub}(C0)$, and
  - each edge (v, v') to a role $\mathcal{L}(v, v')$.

Def.: Initial completion graph

The **initial completion graph** G_0 for C_0 is the graph that contains only one node v_0, no edges, and has $\mathcal{L}(v_0) = \{C_0\}$.

Complete and clash-free completion graph

The idea of the algorithm is to start from the initial completion graph, and to apply **expansion rules** until some graph is reached to which no more rules are applicable, and which does not contain an explicit contradiction (or clash).

Def.: Clash, clash free completion graph

- A completion graph $G = \langle V, E, \mathcal{L} \rangle$ contains a **clash** if for some $v \in V$ and some concept C, we have that $\{C, \neg C\} \subseteq \mathcal{L}(v)$.
- A completion graph is called **clash-free** if it contains no clash.

Def.: Complete completion graph

A completion graph is **complete** if no expansion rule can be applied to it.

Expansion rules for $\mathcal{ALC}$ concepts

Expansion rules for $\mathcal{ALC}$ concept satisfiability		
$\sqcap$-rule	if	$C_1 \sqcap C_2 \in \mathcal{L}(v)$ and $\{C_1, C_2\} \not\subseteq \mathcal{L}(v)$
	then	$\mathcal{L}(v) := \mathcal{L}(v) \cup \{C_1, C_2\}$
$\sqcup$-rule	if	$C_1 \sqcup C_2 \in \mathcal{L}(v)$ and $\{C_1, C_2\} \cap \mathcal{L}(v) = \emptyset$
	then	$\mathcal{L}(v) := \mathcal{L}(v) \cup \{D\}$ for some $D \in \{C_1, C_2\}$
$\exists$-rule	if	$\exists P.C \in \mathcal{L}(v)$, and there is no w such that $\mathcal{L}(v, w) = P$ and $C \in \mathcal{L}(w)$
	then	create a new node w and an edge (v, w), and set $\mathcal{L}(v, w) := P$ and $\mathcal{L}(w) := \{C\}$
$\forall$-rule	if	$\forall P.C \in \mathcal{L}(v)$, and there is some w such that $\mathcal{L}(v, w) = P$ and $C \notin \mathcal{L}(w)$
	then	$\mathcal{L}(w) := \mathcal{L}(w) \cup \{C\}$

unibz

The tableau algorithm for $\mathcal{ALC}$ concept satisfiability

1. Let $\mathcal{G}_0 = \{G_0\}$ be the set that contains only the initial completion graph G_0 for C_0.
2. For $i \geq 0$, obtain the set $\mathcal{G}_{i+1}$ of all **clash-free graphs** that can be obtained by applying an expansion rule to some $G \in \mathcal{G}_i$.
3. If for some $i \geq 0$ we have that:
   - there is a **complete** $G \in \mathcal{G}_i$, then the algorithm answers **yes**;
   - $\mathcal{G}_i = \emptyset$, then the algorithm answers **no**.

Next we show that this yields a sound and complete algorithm for deciding concept satisfiability.

Theorem

The above procedure terminates, and it answers yes iff C_0 is satisfiable.

Tableau for concept satisfiability – Example

Consider concept $C_0 = \underbrace{(A_1 \sqcap \overbrace{\exists P.(A_2 \sqcup A_3)}^{C_3})}_{C_1} \sqcap \underbrace{\forall P.\neg A_2}_{C_2}$

1. $G_0 = \langle\{x_0\}, \emptyset, \mathcal{L}_0\rangle$, with $\mathcal{L}_0(x_0) = \{C_0\}$
2. $G_1 = \langle\{x_0\}, \emptyset, \mathcal{L}_1\rangle$, with $\mathcal{L}_1(x_0) = \{C_0, C_1, C_2\}$ (by $\sqcap$-rule)
3. $G_2 = \langle\{x_0\}, \emptyset, \mathcal{L}_2\rangle$, with $\mathcal{L}_2(x_0) = \{C_0, C_1, C_2, A_1, C_3\}$ (by $\sqcap$-rule)
4. $G_3 = \langle\{x_0, x_1\}, \{(x_0, x_1)\}, \mathcal{L}_3\rangle$, with $\mathcal{L}_3(x_0) = \{C_0, C_1, C_2, A_1, C_3\}$,
   $\mathcal{L}_3(x_1) = \{A_2 \sqcup A_3\}$, $\mathcal{L}_3(x_0, x_1) = P$ (by $\exists$-rule)
5. $G_4 = \langle\{x_0, x_1\}, \{(x_0, x_1)\}, \mathcal{L}_4\rangle$, with $\mathcal{L}_4(x_0) = \{C_0, C_1, C_2, A_1, C_3\}$,
   $\mathcal{L}_4(x_1) = \{A_2 \sqcup A_3, \neg A_2\}$, $\mathcal{L}_4(x_0, x_1) = P$ (by $\forall$-rule)
6. $G_5 = \langle\{x_0, x_1\}, \{(x_0, x_1)\}, \mathcal{L}_5\rangle$, with $\mathcal{L}_5(x_0) = \{C_0, C_1, C_2, A_1, C_3\}$,
   $\mathcal{L}_5(x_1) = \{A_2 \sqcup A_3, \neg A_2, A_2\}$, $\mathcal{L}_5(x_0, x_1) = P$ $\rightsquigarrow$ **clash**
   $G_6 = \langle\{x_0, x_1\}, \{(x_0, x_1)\}, \mathcal{L}_6\rangle$, with $\mathcal{L}_6(x_0) = \{C_0, C_1, C_2, A_1, C_3\}$,
   $\mathcal{L}_6(x_1) = \{A_2 \sqcup A_3, \neg A_2, A_3\}$, $\mathcal{L}_6(x_0, x_1) = P$ (by $\sqcup$-rule)
   $\rightsquigarrow$ **complete and clash-free**

Termination of the tableau algorithm

Lemma

The tableau algorithm for $\mathcal{ALC}$ **terminates**.

Proof sketch.

- Each completion graph G is a finite tree:
  - its depth is linearly bounded by $|C_0|$ (in fact, by the quantifier depth);
  - its breadth is linearly bounded by $|C_0|$ (in fact, by the number of existentials).
- All concepts added to the labels are subconcepts of C_0, and all roles added to the edge labels occur in C_0. Hence the labels are finite.
- The graphs grow 'monotonically': there is no deleting and regenerating of nodes or labels
- Every completion graph G obtained from G_0 will eventually be expanded into some G' that either *(a)* contains a clash, or *(b)* is complete. Hence, the algorithm will eventually answer yes or no. □

Completion graphs as representations of model

A complete and clash-free completion graph represents an interpretation.

Def.: Induced interpretation

Let $G = \langle V, E, \mathcal{L} \rangle$ be a completion graph.

We define the interpretation $\mathcal{I}_G = (\Delta^{\mathcal{I}_G}, \cdot^{\mathcal{I}_G})$ **induced by** G as follows:

- The domain $\Delta^{\mathcal{I}_G}$ is the set V of nodes of G.
- The interpretation function $\cdot^{\mathcal{I}_G}$ is given by the labels:
  - For each atomic concept A, we have $A^{\mathcal{I}_G} = \{v \mid A \in \mathcal{L}(v)\}$.
  - For each (atomic) role P, we have $P^{\mathcal{I}_G} = \{(v, w) \mid P = \mathcal{L}(v, w)\}$.

Then we can prove by induction on the concept structure [Exercise]:

Lemma

Let G be **complete and clash-free** completion graph. Then, for every node v and every $\mathcal{ALC}$ concept C

$$C \in \mathcal{L}(v) \quad \text{if and only if} \quad v \in C^{\mathcal{I}_G}.$$

Soundness of the tableau algorithm

With the previous lemma, it is easy to show that the algorithm is **sound**.

Lemma (L1)

Let G be a complete and clash-free completion graph for C_0 constructed by the tableau algorithm. Then $\mathcal{I}_G \models C_0$.

Proof.

By construction of G, we know that $C_0 \in \mathcal{L}(v_0)$. Hence, by the previous lemma, we have that $v_0 \in C_0^{\mathcal{I}_G}$, and thus $\mathcal{I}_G \models C_0$, as desired. □

Corollary (Soundness)

If the tableau algorithm builds a complete and clash-free completion graph for and $\mathcal{ALC}$ concept C_0, then C_0 is satisfiable.

Simulating models in completion graphs

Towards showing completeness of the tableau algorithm, we need to relate interpretations to completion graphs.

Def.: Intepretation simulating a completion graph

We say that an interpretation $\mathcal{I}$ **simulates** a completion graph $G = \langle V, E, \mathcal{L} \rangle$ if there exists a mapping $\pi : V \to \Delta^{\mathcal{I}}$ such that:

- for each node $v \in V$, if $C \in \mathcal{L}(v)$ then $\pi(v) \in C^{\mathcal{I}}$.
- for each edge $(v, w) \in E$, if $P = \mathcal{L}(v, w)$ then $(\pi(v), \pi(w)) \in P^{\mathcal{I}}$.

Note: A completion graph simulated by an interpretation is always clash-free.

Completeness of the tableau algorithm

Lemma (L2)

If $\mathcal{I} \models C_0$, then there exists some $i \geq 0$ and some complete and clash-free $G \in \mathcal{G}_i$ such that $\mathcal{I}$ simulates G.

Proof sketch.

Roughly, we show that:

1. $\mathcal{I}$ simulates G_0.
2. If some $G \in \mathcal{G}_i$ is simulated by $\mathcal{I}$ and G is not complete, then there is some $G \in \mathcal{G}_{i+1}$ that is also simulated by $\mathcal{I}$.

   Informal intuition: $\mathcal{I}$ shows us how to apply the expansion rules (in particular, the $\sqcup$-rule) in such a way that the simulation is preserved.

The claim then follows since rule application eventually leads to a complete graph. □

Corollary (Completeness)

If C_0 is satisfiable, then the algorithm builds a complete and clash-free G for C_0.

Tree shaped interpretation

It is not hard to see that the completion graphs generated by the tableau algorithm, and the interpretations they induce, have a very specific shape.

Def.: Tree shaped interpretation

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is **tree-shaped** if the graph $\langle V, E\rangle$ with $V = \Delta^{\mathcal{I}}$ and $E = \{(d, d') \mid (d, d') \in P^{\mathcal{I}}$ for some role $P\}$ is a tree.

A simple inspection of the expansion rules reveals that each $\mathcal{I}_G$ induced from a constructed completion graph G is tree-shaped.

Formally, we can show:

Lemma (L3)

If $\mathcal{I}_G$ is an interpretation induced by a completion graph G obtained with the algorithm above, then it is tree shaped.

Tree model property

We have seen that:

- If C_0 has a model, then there is a complete and clash-free completion graph for C_0 (which simulates that model) [Lemma L3].
- If there is a complete and clash-free completion graph for C_0, then there is a tree-shaped model of C_0 (induced by that graph) [Lemmas L1, L2].

Hence, putting this together, we get that if C_0 has a model, then it has a tree shaped model.

Theorem (Tree model property)

Every satisfiable $\mathcal{ALC}$ concept has a tree shaped model.

Note: this is an alternative proof to the one based on bisimulations.

Tree model property – Computational impact

The tree model property is very important and useful:

- We only need to look at tree shaped structures when reasoning about $\mathcal{ALC}$ concepts.
- Trees are computationally 'friendly'.
- We can apply techniques for trees to obtain algorithms and complexity bounds.

However, as we have seen using bisimulations, this property also exposes a **limitation in the expressive power** of $\mathcal{ALC}$ concepts:

- Intuitively, they cannot distinguish (non)tree-shapedness.
- They cannot describe, for example, structures with cycles.

Note: The tree model property provides a further justification why DLs are a decidable fragment of FOL.

Satisfiability of $\mathcal{ALC}$ concepts – Exercises

Exercise

Check the satisfiability of the following concepts:

1. $\neg(\forall R.A \sqcup \exists R.(\neg A \sqcap \neg B))$
2. $\exists R.(\forall S.C) \sqcap \forall R.(\exists S.\neg C)$
3. $\exists S.C \sqcap \exists S.D \sqcap \forall S.(\neg C \sqcup \neg D)$
4. $\exists S.(C \sqcap D) \sqcap (\forall S.\neg C \sqcup \exists S.\neg D)$
5. $C \sqcap \exists R.A \sqcap \exists R.B \sqcap \neg\exists R.(A \sqcap B)$

Exercise

Check if the following subsumption is valid:

$$\neg\forall R.A \sqcap \forall R.((\forall R.B) \sqcup A) \ \sqsubseteq\ \forall R.\neg(\exists R.A) \sqcap \exists R.(\exists R.B)$$

unibz

Some significant cases of $\mathcal{ALC}$ subsumption – Exercises

Which of the following statements is true? Explain your answer.

1. $\forall R.(A \sqcap B) \sqsubseteq \forall R.A \sqcap \forall R.B$ $\quad\surd$
2. $\forall R.A \sqcap \forall R.B \sqsubseteq \forall R.(A \sqcap B)$ $\quad\surd$
3. $\forall R.A \sqcup \forall R.B \sqsubseteq \forall R.(A \sqcup B)$ $\quad\surd$
4. $\forall R.(A \sqcup B) \sqsubseteq \forall R.A \sqcup \forall R.B$ $\quad R^{\mathcal{I}} = \{(x,y),(x,z)\}, A^{\mathcal{I}} = \{y\},\ B^{\mathcal{I}} = \{z\}$
5. $\exists R.(A \sqcap B) \sqsubseteq \exists R.A \sqcap \exists R.B$ $\quad\surd$
6. $\exists R.(A \sqcup B) \sqsubseteq \exists R.A \sqcup \exists R.B$ $\quad\surd$
7. $\exists R.A \sqcup \exists R.B \sqsubseteq \exists R.(A \sqcup B)$ $\quad\surd$
8. $\exists R.A \sqcap \exists R.B \sqsubseteq \exists R.(A \sqcap B)$ $\quad R^{\mathcal{I}} = \{(x,y),(x,z)\},\ A^{\mathcal{I}} = \{y\},\ B^{\mathcal{I}} = \{z\}$

Outline of Part 6

Complexity of reasoning in $\mathcal{ALC}$

Exercise

Consider the concept C_n defined inductively as follows;

$$\begin{aligned} C_1 &= \exists P.A \sqcap \exists P.\neg A \\ C_{i+1} &= \exists P.A \sqcap \exists P.\neg A \sqcap \forall P.C_i, \qquad \text{for } i \in \{1,\ldots,n\} \end{aligned}$$

Check the form of the interpretation induced by the completion graph obtained by starting from $C_n(x_0)$.

Solution

Given the input concept C_n, the satisfiability algorithm generates a complete and clash free completion graph that is a binary tree of depth n, and thus induces an interpretation with $2^{n+1}-1$ individuals.

So, in principle, the complexity of checking satisfiability of an $\mathcal{ALC}$ concept might require exponential space.

However, we show that this can be avoided.

Upper bound for concept satisfiability in $\mathcal{ALC}$

Theorem [Schmidt-Schauss and Smolka 1991]

Satisfiability of $\mathcal{ALC}$ concepts is in PSPACE.

Proof sketch.

We show that if an $\mathcal{ALC}$ concept is satisfiable, we can construct a model using only polynomial space.

- Since $\text{PSPACE} = \text{NPSPACE}$, we consider a non-deterministic algorithm that for each application of the $\sqcup$-rule, chooses the "correct" graph.
- Then, the tree model property of $\mathcal{ALC}$ implies that the different branches of the tree model to be constructed by the algorithm can be explored separately, **in a depth-first manner**, as follows:
  1. Apply **exhaustively** both the $\sqcap$-rule and (non-deterministically) the $\sqcup$-rule, and check for clashes.
  2. **Choose a node** x and apply the $\exists$-rule to generate all necessary direct successors of x.
  3. Apply the $\forall$-rule to propagate the labels to the newly generated successors.
  4. Handle the successors in the same way, one after the other. □

Outline of Part 6

unibz

Sources of complexity for reasoning over concepts

We analyze now the intrinsic complexity of reasoning over concept expressions for various sublanguages of $\mathcal{ALC}$.

Two sources of complexity:

- Union ($\mathcal{U}$) (and Booleans in general) require solving propositional satisfiability $\rightsquigarrow$ complexity of type NP.
- Interaction between $\exists R.C$ ($\mathcal{E}$) and $\forall R.C$ $\rightsquigarrow$ complexity of type coNP.

When they are combined, the complexity jumps to PSpace.

This provides the basis for the hardness results in the following table:

Complexity of concept satisfiability: [Donini, Hollunder, et al. 1992; Donini, Lenzerini, et al. 1997]

$\mathcal{AL}$, $\mathcal{ALN}$	PTime
$\mathcal{ALU}$, $\mathcal{ALUN}$	NP-complete
$\mathcal{ALE}$	coNP-complete
$\mathcal{ALC}$, $\mathcal{ALCN}$, $\mathcal{ALCI}$, $\mathcal{ALCQI}$	PSpace-complete

Concept satisfiability in $\mathcal{ALU}$ is NP-hard

We reduce **satisfiability of Boolean formulae in CNF** to **concept satisfiability in $\mathcal{ALU}$** .

For a Boolean formula F in CNF, let $\rho(F)$ be the $\mathcal{ALU}$ concept obtained by:

- considering Boolean variables as atomic concepts, and
- replacing in F each $\wedge$ with $\sqcap$, and each $\vee$ with $\sqcup$.

Theorem

F is satisfiable iff $\rho(F)$ is satisfiable.

Proof.

Let $F = C_1 \wedge \cdots \wedge C_n$ be a Boolean formula in CNF over Boolean variables $A_1, \ldots, A_k$.

Then F is satisfiable if and only if one can choose in every clause C_i a literal L_i such that $\{L_1, \ldots, L_n\}$ does not contain A_j and $\neg A_j$ for some variable A_j.

Concept satisfiability in $\mathcal{ALU}$ is NP-hard (Cont'd)

Proof ("Only If" Part).

Suppose F is satisfiable. Then there exist $L_1, \ldots, L_n$ as specified above.
Let $\mathcal{I}$ be the interpretation with $\Delta^{\mathcal{I}} = \{1\}$, and such that

$$A^{\mathcal{I}} = \begin{cases} \{1\}, & \text{if } A = L_i \text{ for some } i \\ \emptyset, & \text{otherwise} \end{cases} \qquad P^{\mathcal{I}} = \emptyset, \text{ for every role } P.$$

Then $L_i^{\mathcal{I}} = \{1\}$, for $i \in \{1 \ldots, n\}$. Hence $(\rho(F))^{\mathcal{I}} = \{1\}$, so $\rho(F)$ is satisfiable.

Proof ("If" Part).

Suppose $\rho(F)$ is a satisfiable concept.

Then there exists an interpretation $\mathcal{I}$ and an $a \in \Delta^{\mathcal{I}}$ such that $a \in (\rho(F))^{\mathcal{I}}$.

Hence every clause C_i contains a literal L_i such that $a \in L_i^{\mathcal{I}}$.

Thus $\{L_1, \ldots, L_n\}$ does not contain A_j and $\neg A_j$ for some variable A_j, which implies that F is satisfiable. □

Concept satisfiability in $\mathcal{ALE}$ is coNP-hard

Def.: Exact Cover

Let $U = \{u_1, \ldots, u_n\}$ be a finite set, and let $\mathcal{M} = \{M_1, \ldots, M_m\}$ be a family of subsets of U.
An **exact cover** for $(U, \mathcal{M})$ are sets $M_{i_1}, \ldots, M_{i_\ell}$ of $\mathcal{M}$ that:

- are pairwise disjoint, i.e., $M_{i_h} \cap M_{i_k} = \emptyset$, for $h \neq k$, and
- cover U, i.e., $M_{i_1} \cup \cdots \cup M_{i_\ell} = U$.

The **Exact Cover problem** consists in checking whether there exists an exact cover for a given $(U, \mathcal{M})$.

The Exact Cover problem is NP-complete.

We reduce **Exact Cover** to **concept unsatisfiability in $\mathcal{ALE}$**.

unibz

Reducing Exact Cover to concept unsatisfiability in $\mathcal{ALE}$

Given $U = \{u^1, \ldots, u^n\}$ and $\mathcal{M} = \{M_1, \ldots, M_m\}$, we consider the concept

$$C_{\mathcal{M}} \;=\; C_1 \sqcap \cdots \sqcap C_m \sqcap D$$

where: $C_i = \text{Æ}_i^1 P.\text{Æ}_i^2 P.\cdots \text{Æ}_i^n P.\text{Æ}_i^1 P.\text{Æ}_i^2 P.\cdots \text{Æ}_i^n P.\top$

$$\text{with } \text{Æ}_i^j = \begin{cases} \exists, & \text{if } u^j \in M_i \\ \forall, & \text{if } u^j \notin M_i \end{cases}$$

$$D = \underbrace{\forall P.\cdots\forall P.}_{2n}\bot$$

Notice that the quantifier prefix is duplicated, i.e., for every element $u^j \in U$ there are two quantifiers in each C_i, one at level j and one at level $n + j$.

Theorem

There is an exact cover for $(U, \mathcal{M})$ iff $C_{\mathcal{M}}$ is unsatisfiable.

Reducing Exact Cover to $\mathcal{ALE}$ concept unsat. – Example

Let $U = \{u^1, u^2, u^3\}$, and $\mathcal{M} = \{M_1, M_2, M_3\}$, where

$$M_1 = \{u^1, u^2\}, \qquad M_2 = \{u^2, u^3\}, \qquad M_3 = \{u^3\}$$

The corresponding $\mathcal{ALE}$-concept is $C_\mathcal{M} = C_1 \sqcap C_2 \sqcap C_3 \sqcap D$, where

$$\begin{array}{lcccl}
 & & & & u^1 \; u^2 \; u^3 \; u^1 \; u^2 \; u^3 \\
\hline
M_1 = \{u^1, u^2\} & \rightsquigarrow & C_1 & = & \exists P.\exists P.\forall P.\exists P.\exists P.\forall P.\top \\
M_2 = \{u^2, u^3\} & \rightsquigarrow & C_2 & = & \forall P.\exists P.\exists P.\forall P.\exists P.\exists P.\top \\
M_3 = \{u^3\} & \rightsquigarrow & C_3 & = & \forall P.\forall P.\exists P.\forall P.\forall P.\exists P.\top \\
 & & D & = & \forall P.\forall P.\forall P.\forall P.\forall P.\forall P.\bot
\end{array}$$

- Intuitively, the existentials in the C_is force the existence of a P-path of length $2n$, iff $(U, \mathcal{M})$ has an exact cover.
- If the existence of such a path is enforced, the presence in $C_\mathcal{M}$ of D causes a clash, otherwise $C_\mathcal{M}$ is satisfiable.
- Notice that for the reduction to work correctly, the quantifier prefix needs to be of length $2n$ rather than n. Consider e.g., the instance of exact cover $(U, \{M_1, M_2\})$, where U, M_1, and M_2 are as above.

Concept satisfiability in $\mathcal{ALC}$ is PSPACE-hard

Def.: Quantified Boolean Formulae

A quantified Boolean formula (QBF) has the form

$$(Æ_1 X_1)(Æ_2 X_2)\cdots(Æ_n X_n)\, F(X_1,\ldots,X_n)$$

where each $Æ_i$ is either $\forall$ or $\exists$, and $F(X_1,\ldots,X_n)$ is a Boolean formula (in CNF) with Boolean variables $X_1,\ldots,X_n$.
Such formula is **valid** if
  for every assignment to X_1 / there exists an assignment to X_1 such that
    for every assignment to X_2 / there exists an assignment to X_2 such that
      ...
        $F(X_1,\ldots,X_n)$ evaluates to true.

The **Quantified Boolean Formulae problem** consists in checking whether a given QBF is valid.

The Quantified Boolean Formulae problem is PSPACE-complete.

We reduce **QBF** to **concept satisfiability in $\mathcal{ALC}$**.

unibz

Reducing QBF to concept satisfiability in $\mathcal{ALC}$

Consider the QBF $Q = (\text{Æ}_1 X_1)(\text{Æ}_2 X_2)\cdots(\text{Æ}_n X_n)\, F$, where $F = G^1 \land \cdots \land G^m$ is a Boolean formula in CNF. We construct the concept

$$C_Q \;=\; D_1 \sqcap C_1^1 \sqcap \cdots \sqcap C_1^m$$

where in C_Q all concepts are formed over atomic concept A and atomic role P.

- The concept D_1 encodes the quantifier prefix, and is defined inductively:

$$D_i \;=\; \begin{cases} \exists P.A \sqcap \exists P.\neg A \sqcap \forall P.D_{i+1}, & \text{if } \text{Æ}_i = \forall \\ \exists P.\top \sqcap \forall P.D_{i+1}, & \text{if } \text{Æ}_i = \exists \end{cases} \quad \text{for } i \in \{1,\ldots,n\}$$

  and $D_{n+1} = \top$.
- Each concept C_1^ℓ encodes a clause G^ℓ, and is defined inductively:

$$C_i^\ell \;=\; \begin{cases} \forall P.(A \sqcup C_{i+1}^\ell), & \text{if } X_i \text{ appears in } G^\ell \\ \forall P.(\neg A \sqcup C_{i+1}^\ell), & \text{if } \neg X_i \text{ appears in } G^\ell \\ \forall P.C_{i+1}^\ell, & \text{if } X_i \text{ does not appear in } G^\ell \end{cases} \quad \text{for } i \in \{1,\ldots,n\}$$

  and $C_{n+1}^\ell = \bot$.

Reducing QBF to $\mathcal{ALC}$ concept satisfiability – Example

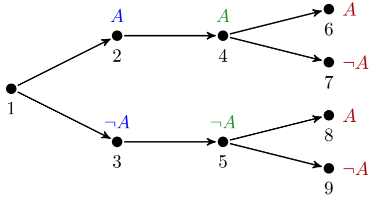
Let $Q = (\forall X)(\exists Y)(\forall Z)\,(\overbrace{(\neg X \lor Y)}^{G^1} \land \overbrace{(X \lor \neg Y)}^{G^2} \land \overbrace{(\neg X \lor Y \lor \neg Z)}^{G^3})$.

Then $C_Q \;=\; D \sqcap C^1 \sqcap C^2 \sqcap C^3$, where

$$
\begin{array}{rcl}
D & = & \exists P.A \sqcap \exists P.\neg A \sqcap \forall P.(\exists P.\top \sqcap \forall P.(\exists P.A \sqcap \exists P.\neg A \sqcap \forall P.\top)) \\
C^1 & = & \forall P.(\neg A \sqcup \forall P.(A \sqcup \forall P.(\bot))) \quad \leftarrow \quad G^1 = \neg X \lor Y \\
C^2 & = & \forall P.(A \sqcup \forall P.(\neg A \sqcup \forall P.(\bot))) \quad \leftarrow \quad G^2 = X \lor \neg Y \\
C^3 & = & \forall P.(\neg A \sqcup \forall P.(A \sqcup \forall P.(\neg A \sqcup \bot))) \quad \leftarrow \quad G^3 = \neg X \lor Y \lor \neg Z
\end{array}
$$

Interpretation generated by D:

Model of C_Q:



Complexity of concept satisfiability and subsumption

- The previous reductions give us lower bounds for concept satisfiability.
- Since C is satisfiable iff $C \sqsubseteq \bot$, and all three languages can express $\bot$, this gives also complementary lower bounds for concept subsumption.
- The tableaux algorithms for $\mathcal{ALC}$, can be refined to work more efficiently for the cases of $\mathcal{ALU}$ and $\mathcal{ALE}$ concept satisfiability and subsumption [Schmidt-Schauss and Smolka 1991; Donini, Hollunder, et al. 1992].

Theorem

Concept satisfiability is:

- NP-complete in $\mathcal{ALU}$,
- coNP-complete in $\mathcal{ALE}$,
- PSPACE-complete in $\mathcal{ALC}$.

Theorem

Concept subsumption is:

- coNP-complete in $\mathcal{ALU}$,
- NP-complete in $\mathcal{ALE}$,
- PSPACE-complete in $\mathcal{ALC}$.

Outline of Part 6

TBox reasoning and ontology reasoning

- **TBox Satisfiability:** $\mathcal{T}$ is satisfiable, if it admits at least one model.
- **Concept Satisfiability w.r.t. a TBox:** C is satisfiable w.r.t. $\mathcal{T}$, if there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^\mathcal{I}$ is not empty, i.e., $\mathcal{T} \not\models C \equiv \bot$.
- **Subsumption:** C_1 is subsumed by C_2 w.r.t. $\mathcal{T}$, if for every model $\mathcal{I}$ of $\mathcal{T}$ we have $C_1^\mathcal{I} \subseteq C_2^\mathcal{I}$, i.e., $\mathcal{T} \models C_1 \sqsubseteq C_2$.
- **Equivalence:** C_1 and C_2 are equivalent w.r.t. $\mathcal{T}$ if for every model $\mathcal{I}$ of $\mathcal{T}$ we have $C_1^\mathcal{I} = C_2^\mathcal{I}$, i.e., $\mathcal{T} \models C_1 \equiv C_2$.
- **Ontology Satisfiability:** $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, if it admits at least one model.

We can reduce all reasoning tasks to concept satisfiability wrt a TBox, and then further to ontology satisfiability. [Exercise]

Outline of Part 6

Acyclic TBox

Def.: Concept definition

A **definition** of an atomic concept A is an assertion of the form $A \equiv C$, where C is an arbitrary concept expression in which A does not occur.

Def.: Cyclic concept definitions

A set of concept definitions is **cyclic** if it is of the form

$$A_1 \equiv C_1[A_2], \quad A_2 \equiv C_2[A_3], \ldots, \quad A_n \equiv C_n[A_1]$$

where $C[A]$ means that A occurs in the concept expression C.

Def.: Acyclic TBox

A TBox is **acyclic** if it is a set of concept definitions that neither contains multiple definitions of the same concept, nor a set of cyclic definitions.

Unfolding w.r.t. an acyclic TBox

Satisfiability of a concept C w.r.t. an acyclic TBox $\mathcal{T}$ can be reduced to pure concept satisfiability by **unfolding C w.r.t. $\mathcal{T}$**:

1. We start from the concept C to check for satisfiability.
2. Whenever $\mathcal{T}$ contains a definition $A \equiv C'$, and A occurs in C, then in C we substitute A with C'.
3. We continue until no more substitutions are possible.

Theorem

Let $\mathit{Unfold}_{\mathcal{T}}(C)$ be the result of unfolding C w.r.t $\mathcal{T}$.
Then C is satisfiable w.r.t. $\mathcal{T}$ iff $\mathit{Unfold}_{\mathcal{T}}(C)$ is satisfiable.

Proof.

By induction on the number of unfolding steps. [Exercise] □

unibz

Complexity of unfolding w.r.t. acyclic TBoxes

Unfolding a concept w.r.t. an acyclic TBox might lead to an **exponential** blow-up.

For each n, let $\mathcal{T}_n$ be the acyclic TBox:

$$\begin{array}{rcl} A_0 & \equiv & \forall P.A_1 \sqcap \forall R.A_1 \\ A_1 & \equiv & \forall P.A_2 \sqcap \forall R.A_2 \\ & \vdots & \\ A_{n-1} & \equiv & \forall P.A_n \sqcap \forall R.A_n \end{array}$$

It is easy to see that $\mathit{Unfold}_{\mathcal{T}_n}(A_0)$ grows exponentially with n.

Concept satisfiability w.r.t. acyclic TBoxex

We adopt a smarter strategy: **unfolding on demand**

Expansion rules for satisfiability of acyclic $\mathcal{ALC}$ TBoxes

$\sqcap$-rule	if	$C_1 \sqcap C_2 \in \mathcal{L}(v)$ and $\{C_1, C_2\} \not\subseteq \mathcal{L}(v)$
	then	$\mathcal{L}(v) := \mathcal{L}(v) \cup \{C_1, C_2\}$
$\sqcup$-rule	if	$C_1 \sqcup C_2 \in \mathcal{L}(v)$ and $\{C_1, C_2\} \cap \mathcal{L}(v) = \emptyset$
	then	$\mathcal{L}(v) := \mathcal{L}(v) \cup \{D\}$ for some $D \in \{C_1, C_2\}$
$\exists$-rule	if	$\exists P.C \in \mathcal{L}(v)$, and there is no w such that $\mathcal{L}(v,w) = P$ and $C \in \mathcal{L}(w)$
	then	create a new node w and an arc (v,w), and set $\mathcal{L}(v,w) := P$ and $\mathcal{L}(w) := \{C\}$
$\forall$-rule	if	$\forall P.C \in \mathcal{L}(v)$, and there is some w such that $\mathcal{L}(v,w) = P$ and $C \notin \mathcal{L}(w)$
	then	$\mathcal{L}(w) := \mathcal{L}(w) \cup \{C\}$
$\mathcal{T}$-rule	if	$A \in \mathcal{L}(v)$, $A \equiv C \in \mathcal{T}$, and $\textsc{nnf}(C) \notin \mathcal{L}(v)$
	then	$\mathcal{L}(v) := \mathcal{L}(v) \cup \{\textsc{nnf}(C)\}$
$\mathcal{T}$-rule	if	$\neg A \in \mathcal{L}(v)$, $A \equiv C \in \mathcal{T}$, and $\textsc{nnf}(\neg C) \notin \mathcal{L}(v)$
	then	$\mathcal{L}(v) := \mathcal{L}(v) \cup \{\textsc{nnf}(\neg C)\}$

Concept satisfiability w.r.t. acyclic TBoxex – Complexity

Theorem

In $\mathcal{ALC}$, concept satisfiability w.r.t. acyclic TBoxes is **PSPACE-complete**.

Proof.

For the upper bound, we can make use of the tableau algorithm, adopting the same strategy for rule application as the one for plain concept satisfiabily.

For the lower bound, it suffices to observe that PSPACE-hardness already holds for plain concept satisfiablity. □

Outline of Part 6

A tableau algorithm for $\mathcal{ALC}$ ontologies

We extend our algorithm to decide ontology satisfiability.

The algorithm is essentially the same: we start from the initial completion graph, and apply expansion rules until some complete and clash-free graph is reached.

But there are a few differences:

- The initial graph is more complex: it is a representation of the ABox.
- Arc labels in completion graphs are **sets of roles** instead of just one role (to allow for pairs of individuals to be connected by multiple roles).
- The labeled graphs we obtain are not trees, but forests.
- The expansion rules need slight extension/adaptation.

Initial completion graph

Consider an $\mathcal{ALC}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$.

Def.: Initial completion graph

The initial completion graph $G_0 = \langle V, E, \mathcal{L} \rangle$ for $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ is defined as follows:

- V contains one node $\hat{c}$ for each individual c occurring in $\mathcal{A}$.
- Each $\hat{c}$ has the label $\mathcal{L}(\hat{c}) = \{A \mid A(c) \in \mathcal{A}\}$.
- There is an edge $(\hat{c}, \hat{d})$ with role P in its label iff $P(c, d) \in \mathcal{A}$.

unibz

Expansion rule for arbitrary TBox axioms

When the TBox may contain cycles, unfolding cannot be used, since in general it would not terminate.

Instead, we modify the tableau by relying on the following observations:

- $C \sqsubseteq D$ is equivalent to $\top \sqsubseteq \neg C \sqcup D$.
  Hence, $\bigcup_i \{C_i \sqsubseteq D_i\}$ is equivalent to a single inclusion $\top \sqsubseteq \bigsqcap_i (\neg C_i \sqcup D_i)$.
- Let
$$C_{\mathcal{T}} = \bigsqcap_{C \sqsubseteq D \in \mathcal{T}} \mathrm{NNF}(\neg C \sqcup D)$$
  Then for every completion graph G generated by the tableau and for every node v of G, we have to add $C_{\mathcal{T}}$ to the label of v.
- We can obtain this effect by adding a suitable completion rule:

Expansion rules for satisfiability of $\mathcal{ALC}$ ontologies		
...		
$\mathcal{T}$-rule	if	$C_{\mathcal{T}} \notin \mathcal{L}(v)$
	then	$\mathcal{L}(v) := \mathcal{L}(v) \cup \{C_{\mathcal{T}}\}$

Expansion rule for arbitrary TBox axioms – Example

Exercise

Check whether the ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, where $\mathcal{T} = \{A \sqsubseteq \exists R.A\}$ and $\mathcal{A} = \{A(x_0)\}$.

Solution (We denote with $\rightarrow_X$ the application of the X-rule.)

We have that $C_\mathcal{T} = \neg A \sqcup \exists R.A$.

$$\begin{array}{lll}
A(x_0) & \rightarrow_\mathcal{T} & A(x_0),\ C_\mathcal{T}(x_0) \\
 & \rightarrow_\sqcup & A(x_0),\ C_\mathcal{T}(x_0),\ (\exists R.A)(x_0) \\
 & \rightarrow_\exists & A(x_0),\ \ldots,\ R(x_0,x_1),\ A(x_1) \\
 & \rightarrow_\mathcal{T} & A(x_0),\ \ldots,\ R(x_0,x_1),\ A(x_1),\ C_\mathcal{T}(x_1) \\
 & \rightarrow_\sqcup & A(x_0),\ \ldots,\ R(x_0,x_1),\ A(x_1),\ C_\mathcal{T}(x_1),\ \exists R.A(x_1) \\
 & \rightarrow_\exists & A(x_0),\ \ldots,\ R(x_0,x_1),\ A(x_1),\ \ldots,\ R(x_1,x_2),\ A(x_2) \\
 & \rightarrow_\mathcal{T} & \cdots
\end{array}$$

($C(v)$ denotes that the completion graph has a node v labeled with C (similarly for roles).)

Termination is no longer guaranteed!

Due to the application of the $\mathcal{T}$-rule, the nesting of the concepts does not decrease with each rule-application step.

Blocking

To guarantee termination, we need to understand when it is not necessary anymore to create new objects.

Idea: to regain termination, avoid generating new successors for nodes that "behave similarly" to some ancestor (cycle-detection).

Def.: **Blocking**

Let $G = \langle V, E, \mathcal{L} \rangle$ be a completion graph.

- We say that $v \in V$ is **directly blocked** if it is reachable from a node $w \in V$ with $\mathcal{L}(v) \subseteq \mathcal{L}(w)$.
- If w is the closest such node to v, we say that v is **blocked by** w.
- A node is **blocked** if it is directly blocked or one of its ancestors is blocked.

We restrict the application of the $\exists$-rule to nodes that are **not blocked**.

Expansion rules for $\mathcal{ALC}$ ontologies with arbitrary TBoxes

Expansion rules for satisfiability of $\mathcal{ALC}$ ontologies

$\sqcap$-rule	if	$C_1 \sqcap C_2 \in \mathcal{L}(v)$ and $\{C_1, C_2\} \not\subseteq \mathcal{L}(v)$
	then	$\mathcal{L}(v) := \mathcal{L}(v) \cup \{C_1, C_2\}$
$\sqcup$-rule	if	$C_1 \sqcup C_2 \in \mathcal{L}(v)$ and $\{C_1, C_2\} \cap \mathcal{L}(v) = \emptyset$
	then	$\mathcal{L}(v) := \mathcal{L}(v) \cup \{D\}$ for some $D \in \{C_1, C_2\}$
$\exists$-rule	if	$\exists P.C \in \mathcal{L}(v)$, v **is not blocked**, and there is no w such that $P \in \mathcal{L}(v, w)$ and $C \in \mathcal{L}(w)$
	then	create a new node w and an arc (v, w), and set $\mathcal{L}(v, w) := \{P\}$ and $\mathcal{L}(w) := \{C\}$
$\forall$-rule	if	$\forall P.C \in \mathcal{L}(v)$, and there is some w such that $P \in \mathcal{L}(v, w)$ and $C \notin \mathcal{L}(w)$
	then	$\mathcal{L}(w) := \mathcal{L}(w) \cup \{C\}$
$\mathcal{T}$-rule	if	$C_{\mathcal{T}} \notin \mathcal{L}(v)$
	then	$\mathcal{L}(v) := \mathcal{L}(v) \cup \{C_{\mathcal{T}}\}$

Blocking – Exercise

Exercise

Check whether the ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, where $\mathcal{T} = \{A \sqsubseteq \exists R.A\}$ and $\mathcal{A} = \{A(x_0)\}$.

Solution (We denote with $\rightarrow_X$ an application of the X-rule.)

We have that $C_\mathcal{T} = \neg A \sqcup \exists R.A$.

$$\begin{array}{lll}
A(x_0) & \rightarrow_\mathcal{T} & A(x_0),\ C_\mathcal{T}(x_0) \\
& \rightarrow_\sqcup & A(x_0),\ C_\mathcal{T}(x_0),\ (\exists R.A)(x_0) \\
& \rightarrow_\exists & A(x_0),\ C_\mathcal{T}(x_0),\ (\exists R.A)(x_0),\ R(x_0,x_1),\ A(x_1) \\
& \rightarrow_\mathcal{T} & A(x_0),\ C_\mathcal{T}(x_0),\ (\exists R.A)(x_0),\ R(x_0,x_1),\ A(x_1),\ C_\mathcal{T}(x_1) \\
& \rightarrow_\sqcup & A(x_0),\ C_\mathcal{T}(x_0),\ (\exists R.A)(x_0),\ R(x_0,x_1),\ A(x_1),\ C_\mathcal{T}(x_1),\ (\exists R.A)(x_1)
\end{array}$$

Now x_1 is blocked by x_0 since $\mathcal{L}(x_1) = \mathcal{L}(x_0) = \{A, C_\mathcal{T}, \exists R.A\}$ (hence $\mathcal{L}(x_1) \subseteq \mathcal{L}(x_0)$).

Tableau algorithm for $\mathcal{ALC}$ ontology satisfiability

The rest of the algorithm is exactly as for $\mathcal{ALC}$ concepts.

We are going to show that this extended tableau algorithm is a decision procedure for ontology satisfiability.

Theorem

For an $\mathcal{ALC}$ ontology $\mathcal{O}$, the algorithm terminates, and it answers yes if and only if $\mathcal{O}$ is satisfiable.

Tableau algorithm for $\mathcal{ALC}$– Termination

Lemma

The tableau algorithm for $\mathcal{ALC}$ ontology satisfiability terminates.

Proof sketch.

Let be $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ALC}$ ontology.
Each completion graph G is a forest:

- It has one root node for each individual in $\mathcal{A}$.
- Blocking ensures that the depth of each branch is finite (bounded by an exponential in $|\mathcal{O}|$).
- The branching degree of each node is still linearly bounded by $|\mathcal{O}|$ (in fact, by the number of existentials in $\mathcal{O}$).
- Hence the generated graphs are always finite.

And as before:

- All concepts added to the labels occur in $\mathcal{A}$ or in $C_{\mathcal{T}}$.
- G is constructed without deleting or regenerating nodes or labels. □

Tableau algorithm for $\mathcal{ALC}$– Soundness

Similarly as before, a complete and clash-free G induces a model $\mathcal{I}_G$ of $\mathcal{O}$, but the induced model is slightly different

Def.: Interpretation induced by a completion graph

Let $G = \langle V, E, \mathcal{L} \rangle$ be a completion graph for $\mathcal{O}$. We define the interpretation $\mathcal{I}_G = (\Delta^{\mathcal{I}_G}, \cdot^{\mathcal{I}_G})$ as follows:

- $\Delta^{\mathcal{I}_G} = \{v \mid v \in V$ and v is not blocked$\}$
- Each Individual c is interpreted as the corresponding initial node, i.e., $c^{\mathcal{I}} = \hat{c}$.
- $A^{\mathcal{I}_G} = \{v \mid v \in \Delta^{\mathcal{I}_G}$ and $A \in \mathcal{L}(v)\}$, for each concept name A.
- $P^{\mathcal{I}_G} = \{(v,w) \mid \{v,w\} \subseteq \Delta^{\mathcal{I}_G}$ and $P \in \mathcal{L}(x,y)\} \cup$
  $\{(v,w) \mid v \in \Delta^{\mathcal{I}_G},\ P \in \mathcal{L}(v,w'),$ and w' is blocked by $w\}$,
  for each role name P.

Tableau algorithm for $\mathcal{ALC}$– Soundness (cont'd)

Lemma

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ALC}$ ontology, and let G be a complete and clash-free completion graph for $\mathcal{O}$ constructed by the tableau algorithm. Then $\mathcal{I}_G \models \mathcal{O}$.

Proof.

1. As we did before, we show that if G is complete and clash-free, then for every node v and for every concept C, $C \in \mathcal{L}(v)$ implies $v \in C^{\mathcal{I}_G}$.
2. By construction of the initial graph, we know that:
   - $C \in \mathcal{L}(\hat{a})$, for each $C(a) \in \mathcal{A}$, and
   - $P \in \mathcal{L}(\hat{a}, \hat{b})$, for each $P(a, b) \in \mathcal{A}$.

   So, by construction of $\mathcal{I}_G$ and item 1, $\mathcal{I}_G \models \mathcal{A}$.
3. Since $C_{\mathcal{T}} \in \mathcal{L}(v)$ for every node v, we have $C_{\mathcal{T}}^{\mathcal{I}_G} = \Delta^{\mathcal{I}_G}$, hence $\mathcal{I}_G \models \mathcal{T}$.
4. $\mathcal{I}_G \models \mathcal{A}$ and $\mathcal{I}_G \models \mathcal{T}$ imply that $\mathcal{I}_G \models \mathcal{O}$. □

Tableau algorithm for $\mathcal{ALC}$– Soundness (cont'd)

Corollary (Soundness)

If the tableau algorithm builds a complete and clash-free completion graph for an $\mathcal{ALC}$ ontology $\mathcal{O}$, then $\mathcal{O}$ is satisfiable.

Tableau algorithm for $\mathcal{ALC}$– Completeness

Lemma (Completeness)

If an $\mathcal{ALC}$ ontology $\mathcal{O}$ is satisfiable, then the tableau algorithm builds a complete and clash-free completion graph for $\mathcal{O}$.

Proof sketch.

As before, we show that every model $\mathcal{I}$ of $\mathcal{O}$ simulates a complete and clash free completion graph that is constructed by the algorithm.

The notion of simulation π is similar to the case of concept expressions, but it only needs to map the non-blocked nodes, and additionally we require $\pi(\hat{a}) = a^{\mathcal{I}}$ for each initial node $\hat{a}$. □

Finite model property

From the complete and clash-free completion graph constructed by the tableau algorithm for a satisfiable ontology, and from the corresponding interpretation, we can derive some notable model theoretic properties.

Theorem

A satisfiable $\mathcal{ALC}$ ontology has a finite model.

Proof.

The model constructed via tableau is finite.
Completeness of the tableau procedure implies that if an ontology is satisfiable, then the algorithm will find a model, which is indeed finite. $\square$

unibz

Forest model property

- Completion graphs for an ontology are not necessarily tree-shaped, since arbitrary relations between individuals may hold.
- However, a completion graph is composed of a set of trees rooted at the (possibly interconnected) objects representing the individuals.

Def.: **Forest-shaped interpretation**

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is **forest-shaped** if the graph $\langle V, E \rangle$ with

- $V = \Delta^{\mathcal{I}}$, and
- $E = \{(d, d) \mid (d, d) \in P^{\mathcal{I}}$ for some role R and
  $d, d \notin \{a^{\mathcal{I}} \mid a$ an individual$\}\}$

is a set of (disconnected) trees.

Forest model property (cont'd)

The model $\mathcal{I}_G$ obtained from a completion graph is not forest-shaped in general (the blocked nodes create cycles), but it can be shown that the following property holds.

Theorem (Forest model property)

Every satisfiable $\mathcal{ALC}$ ontology has a **forest-shaped model**.

Note:

- Unlike the case of $\mathcal{ALC}$ concepts, trees may now be **infinite**!.
- This property is practically as good (and as restrictive) as the tree-model property of $\mathcal{ALC}$ concepts.
- Many DLs have similar tree/forest model properties, but in some cases we need to adapt slightly the definition of tree/forest shaped models.

Complexity of ontology satisfiability

The computational complexity of the **tableau algorithm is not optimal**:

- The forest can be very big:
  - branches in the forest can have exponential depth before blocking occurs;
  - the whole forest can be double exponentially large.
- Hence, the overall algorithm runs **no longer in** PSpace , and in the worst case needs non-deterministic double exponential time (in 2NExpTime).
- With some adaptations and modified blocking strategies, one can make forests to be of size at most single exponential.
- This provides a non-deterministic exponential upper bound.
  In other words, the (improved) tableau algorithm shows that reasoning over $\mathcal{ALC}$ ontologies is **in** NExpTime.
- We will see that reasoning over $\mathcal{ALC}$ ontologies is “only” ExpTime-hard.
- To obtain worst-case optimal decision procedures we need different techniques.

Complexity of reasoning over DL ontologies

Summing up, reasoning over DL ontologies is much more complex than reasoning over concept expressions.

Bad news:

- without restrictions on the form of TBox assertions, reasoning over DL ontologies is already ExpTime-**hard**, even for very simple DLs (see, e.g., [Donini 2003]).

Good news:

- We can add a lot of expressivity (i.e., essentially all DL constructs seen so far), while still staying within the ExpTime upper bound [Pratt 1979; Schild 1991; Calvanese and De Giacomo 2003].
- There are DL reasoners that perform reasonably well in practice for such DLs (e.g, Racer, Pellet, Fact++, . . .) [Möller and Haarslev 2003].

Outline of Part 6

Lower bounds for reasoning over $\mathcal{ALC}$ ontologies

Theorem

The following problems are ExpTime-hard in $\mathcal{ALC}$:

- concept subsumption w.r.t. TBoxes;
- concept satisfiability w.r.t. TBoxes;
- ontology satisfiability.

Recall that $\mathcal{ALC}$ is closed under concept negation and that:

- $\mathcal{T} \models C_1 \sqsubseteq C_2$ iff $C_1 \sqcap \neg C_2$ is unsatisfiable w.r.t. $\mathcal{T}$.
- C is satisfiable w.r.t. $\mathcal{T}$ iff $\langle \mathcal{T} \cup \{A_n \sqsubseteq C\},\ \{A_n(a_0)\}\rangle$ is satisfiable, where A_n is a fresh concept name.

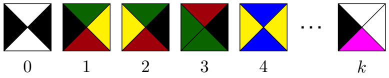
Hence it suffices to prove the hardness result for subsumption w.r.t. TBoxes.

We look at a proof based on encoding the **two player corridor tiling problem**.

Two player corridor tiling game

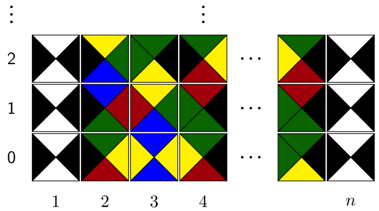
A **tiling system** $\mathbf{T}$ consists of a finite set of square tile types with horizontal and vertical adjacency conditions.

- The adjacency conditions are sometimes represented by coloring the four edges of the tiles (assuming that the tiles cannot be flipped or rotated).



- Adjacent tiles must have the same color on touching sides.

A **corridor tiling** is a tiling of a corridor of width n with tiles of $\mathbf{T}$ respecting the adjacency conditions.



Two player corridor tiling game

A two player corridor tiling game is played by two players, ∀lice and ∃lias:

- ∀lice and ∃lias alternatively place a tile, row by row, from left to right, respecting adjacency conditions.
- ∃lias wins if
  - he can place a special “winning tile” in the second position of a row, or
  - ∀lice cannot place a tile when it is her turn to move.
- In other words, ∀lice wins (i.e., ∃lias loses) if
  - ∃lias cannot place a tile when it is his turn to move, or
  - the game goes on forever.

Two player corridor tiling problem

Def.: Two player corridor tiling problem

Instance:

- A tiling system, expressed as $\mathbf{T} = (k, H, V)$, where
  - $0, 1, \ldots, k$ are the tile types, with k being the winning tile.
  - $H \subseteq [0..k] \times [0..k]$ is the horizontal adjacency relation.
  - $V \subseteq [0..k] \times [0..k]$ is the vertical adjacency relation.
- An initial row of tiles $t_1 t_2 \cdots t_n$ of length n.

Question: Does ∃lias have a winning strategy?
I.e., for every move ∀lice makes, is there a move ∃lias can counter with, in such a way that he wins?

Theorem

Two player corridor tiling is ExpTime-complete.

Encoding of two player corridor tiling in $\mathcal{ALC}$

We show now how to reduce the two player corridor tiling problem to subsumption w.r.t. an $\mathcal{ALC}$ TBox.

- The intention is to represent each placed tile by an object. The object carries the information about the last n moves made.
- We use an atomic role $next$ to connect objects representing successive tiles. We connect an object at the end of a row, to the one at the beginning of the next row.
- We use the following atomic concepts:
  - C_i, for $i \in [1..n]$, denoting that the *column* of the tile represented by an object is i.
  - L_i^t, for each $i \in [1..n]$ and each $t \in [0..k]$, denoting that the *last* tile placed in column i has been tile t.
  - A, denoting that it is $\forall$*lice*'s turn to place the current tile.
  - W, denoting that $\exists$lias *wins*.

We use these concepts and roles to construct an $\mathcal{ALC}$ TBox $\mathcal{T}_{\mathbf{T}}$ that encodes a tiling problem.

Encoding of two player corridor tiling in $\mathcal{ALC}$ (2)

We introduce in $\mathcal{T}_{\mathbf{T}}$ the following concept inclusions to ensure that tilings are correctly represented.

- To encode that each tile is placed in exactly one column in the corridor:

$$\begin{array}{rcll} \top & \sqsubseteq & C_1 \sqcup \cdots \sqcup C_n & \\ C_i & \sqsubseteq & \neg C_j & \text{for } i,j \in [1..n], \quad i \neq j \end{array}$$

- To encode that the tiles are placed in the correct left-to-right order:

$$\begin{array}{rcll} C_i & \sqsubseteq & \forall next.C_{i+1} & \text{for } i \in [1..n{-}1] \\ C_n & \sqsubseteq & \forall next.C_1 & \end{array}$$

- To encode that each column has exactly one tile last placed into it:

$$\begin{array}{rcll} \top & \sqsubseteq & L_i^0 \sqcup \cdots \sqcup L_i^k & \text{for } i \in [1..n] \\ L_i^t & \sqsubseteq & \neg L_i^{t'} & \text{for } i \in [1..n], \quad t,t' \in [0..k], \quad t \neq t' \end{array}$$

Encoding of two player corridor tiling in $\mathcal{ALC}$ (3)

We introduce in $\mathcal{T}_{\mathbf{T}}$ the following concept inclusions to encode the adjacency conditions, by making use of the information carried by the objects.

- To encode the vertical adjacency relation V:

$$C_i \sqcap L_i^t \quad \sqsubseteq \quad \forall next. \bigsqcup_{t' \mid (t,t') \in V} L_i^{t'} \qquad \text{for } i \in [1..n], \quad t \in [0..k]$$

- To encode the horizontal adjacency relation H:

$$C_i \sqcap L_{i-1}^t \quad \sqsubseteq \quad \forall next. \bigsqcup_{t' \mid (t,t') \in H} L_i^{t'} \qquad \text{for } i \in [2..n], \quad t \in [0..k]$$

- To encode that in columns where no move is made nothing changes:

$$\begin{array}{rcll} \neg C_i \sqcap L_i^t & \sqsubseteq & \forall next. L_i^t & \text{for } i \in [1..n], \quad t \in [0..k] \\ \neg C_i \sqcap \neg L_i^t & \sqsubseteq & \forall next. \neg L_i^t & \text{for } i \in [1..n], \quad t \in [0..k] \end{array}$$

Encoding of two player corridor tiling in $\mathcal{ALC}$ (4)

We introduce in $\mathcal{T}_{\mathbf{T}}$ the following concept inclusions to encode the game.

- To encode the existence of all possible moves in the game tree, provided ∃lias hasn't already won:

$$
\begin{array}{rcll}
\neg L_2^k \sqcap C_1 \sqcap L_1^t & \sqsubseteq & \displaystyle\bigsqcap_{t' \,\mid\, (t,t')\in V} \exists next.L_1^{t'}, & \text{for } t \in [0..k] \\
\neg L_2^k \sqcap C_i \sqcap L_i^t \sqcap L_{i-1}^{t'} & \sqsubseteq & \displaystyle\bigsqcap_{t'' \,\mid\, (t,t'')\in V \,\wedge\, (t',t'')\in H} \exists next.L_i^{t''}, & \\
 & & \text{for } i \in [2..n], \quad t,t' \in [0..k] &
\end{array}
$$

- To encode the alternation of moves:

$$
\begin{array}{rcl}
A & \sqsubseteq & \forall next.\neg A \\
\neg A & \sqsubseteq & \forall next.A
\end{array}
$$

- To encode the winning of ∃lias:

$$
W \equiv (A \sqcap L_2^k) \sqcup (A \sqcap \forall next.W) \sqcup (\neg A \sqcap \exists next.W)
$$

ExpTime-hardness of reasoning over $\mathcal{ALC}$ ontologies

Observations:

- if ∃lias cannot move when it is his turn, then W is false for the object representing that tile.
- if ∀lice can force the game to go on forever, then there will be models of $\mathcal{T}_{\mathbf{T}}$ in which W is false.

Theorem

∃lias has a winning strategy for tiling system $\mathbf{T}$ with initial row $t_1 \cdots t_n$
iff
$$\mathcal{T}_{\mathbf{T}} \models A \sqcap C_1 \sqcap L_1^{t_1} \sqcap \cdots \sqcap L_n^{t_n} \sqsubseteq W$$

Since the size of $\mathcal{T}_{\mathbf{T}}$ is polynomial in $\mathbf{T}$ and n, this shows that concept subsumption w.r.t. to $\mathcal{ALC}$ TBoxes is ExpTime-hard (and hence ExpTime-complete).

unibz

Hardness proofs using tilings

Tiling problems are a very useful tool for showing complexity results in description logics, modal logics, and fragments of FOL.

In DLs, they have been used to:

- Show NExpTime-hardness (e.g., for $\mathcal{ALCIOF}$ and extensions):

Bounded tilings

Deciding the existence of a tiling for

- an $n \times n$ grid (or torus) is NP-complete.
- a corridor of width n is PSpace-complete.
- a $2^n \times 2^n$ grid (or torus) is NExpTime-complete.

- Show undecidability (e.g., for DLs with transitive roles in the number restrictions, role value maps, etc.):

Unbounded tilings

Deciding the existence of a tiling for an unbounded grid is undecidable.

Tiling systems and Turing Machines

Tiling problems are very closely related to **Turing Machines** (TMs).

- A row of tiles corresponds to a configuration of the TM, i.e., to the tape content, head position, and state.
- Successive rows correspond to the evolution over time of the TM configuration.
- The horizontal and vertical adjacency relations essentially encode the transition function of the TM.
- The initial row of tiles corresponds to the input word, initially written on the tape.
- The winning tile corresponds to the final state.

Alternating Turing Machines

The tiling we used in our reduction is related to Alternating Turing Machines.

Def.: Alternating Turing Machine (ATM)

An ATM has the form $M = (\Sigma, \Gamma, Q_\forall, Q_\exists, q_0, \delta, q_f, \text{b̸})$, where

- As for an ordinary Turing Machine:
  - Σ is the input alphabet, and Γ the tape alphabet;
  - q_0 is the initial state, and q_f the final state;
  - $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathit{right}, \mathit{left}\}$ is the transition function, where $Q = Q_\exists \cup Q_\forall$.
- $Q_\exists$ is the set of existential states, for which the ATM moves non-deterministically to some successive configuration.
- $Q_\forall$ is the set of universal states, for which the ATM moves to all successive configurations, i.e., it branches off multiple computations.

An ATM **accepts** an input string $w \in \Sigma^*$ if, when started in q_0 with w on the tape, all branched off computations lead to an accepting configuration, i.e., one where the ATM is in q_f.

Two-player tilings and Alternating Turing Machines

A two-player corridor tiling is a simple ‘disguise’ for a PSPACE **ATM** (i.e., and ATM that runs in polynomial space), for which we want to decide acceptance of an input word.

- The initial row of tiles represents the word initially written on the tape.
- Each row of n-tiles corresponds to the tape content, and the width n accounts for the polynomial space used by the ATM.
- The two players $\exists$lias and $\forall$lice correspond to existential and universal states, respectively.
- The alternation between the players in the game corresponds to the alternation between existential and universal moves of the ATM.
- However, there are differences between a two-player tiling and an ATM in the way alternation is handled:
  - In the two-player tiling, the two players strictly alternate at each placed tile.
  - In the ATM, there is no strict alternation between existential and universal states (although one could impose such strict alternation without loss of generality); moreover, one transition corresponds to placing an entire row of tiles, as opposed to a single tile.

ExpTime-hardness of reasoning over $\mathcal{AL}$ ontologies

The lower bound for reasoning over $\mathcal{ALC}$ TBoxes and ontologies can be strengthened to weaker DLs.

Theorem

Concept satisfiability and subsumption w.r.t. $\mathcal{AL}$ TBoxes, and satisfiability of $\mathcal{AL}$ ontologies are ExpTime-hard.

Recall that:

- C is satisfiable w.r.t. $\mathcal{T}$ iff $\mathcal{T} \not\models C \sqsubseteq \bot$.
- C is satisfiable w.r.t. $\mathcal{T}$ iff the ontology $\langle \mathcal{T}, \{C(a_0)\}\rangle$ is satisfiable.

Hence it suffices to prove the result for concept satisfiability w.r.t. a TBox.

We reduce concept satisfiability w.r.t. $\mathcal{ALC}$ TBoxes to concept satisfiability w.r.t. $\mathcal{AL}$ TBoxes.

Note: This reduction is possible only for reasoning w.r.t. a TBox, while (plain) concept satisfiability or subsumption cannot be reduced from $\mathcal{ALC}$ to $\mathcal{AL}$.

Reducing ontology reasoning from $\mathcal{ALC}$ to $\mathcal{AL}$

We reduce concept satisfiability w.r.t. $\mathcal{ALC}$ TBoxes to concept satisfiability w.r.t. $\mathcal{AL}$ TBoxes in a series of steps:

1. Reduce to satisfiability of **atomic** concepts w.r.t. TBoxes with **primitive inclusion assertions only**.
2. Eliminate nesting of constructs in right hand sides of inclusions by introducing new assertions.
3. Encode away qualified existential restrictions.
4. Encode away disjunction.

From $\mathcal{ALC}$ to $\mathcal{AL}$: 1. Simplify assertions and concepts

We reduce concept satisfiability w.r.t. a TBox $\mathcal{T}$ to satisfiability of an **atomic concept** w.r.t. a TBox $\mathcal{T}_1$ with **primitive inclusion assertions only**.

$$C \text{ is satisfiable w.r.t. } \bigcup_i \{C_i \sqsubseteq D_i\}$$

iff

$$A_\mathcal{T} \sqcap C \text{ is satisfiable w.r.t. } \{\ A_\mathcal{T} \ \sqsubseteq\ \bigsqcap_i (\neg C_i \sqcup D_i)\ \sqcap\ \bigsqcap_P \forall P.A_\mathcal{T}\ \}$$

iff

$$A_C \text{ is satisfiable w.r.t. } \left\{ \begin{array}{lll} A_C & \sqsubseteq & A_\mathcal{T} \sqcap C \\ A_\mathcal{T} & \sqsubseteq & \bigsqcap_i (\neg C_i \sqcup D_i)\ \sqcap\ \bigsqcap_P \forall P.A_\mathcal{T} \end{array} \right\}$$

with $A_\mathcal{T}$ and A_C fresh atomic concepts.

From $\mathcal{ALC}$ to $\mathcal{AL}$: 2. Eliminate nesting of constructs

To eliminate the nesting of constructs in the right-hand side of inclusion assertions in $\mathcal{T}_1$, we proceed as follows:

1. We transform the concepts into negation normal form, by pushing negations inside.
2. We replace assertions as follows:

$$\begin{array}{lcl}
A \sqsubseteq C_1 \sqcap C_2 & \rightsquigarrow & A \sqsubseteq C_1, \quad A \sqsubseteq C_2 \\
A \sqsubseteq C_1 \sqcup C_2 & \rightsquigarrow & A \sqsubseteq A_1 \sqcup A_2, \quad A_1 \sqsubseteq C_1, \quad A_2 \sqsubseteq C_2 \\
A \sqsubseteq \forall P.C & \rightsquigarrow & A \sqsubseteq \forall P.A_1, \quad A_1 \sqsubseteq C \\
A \sqsubseteq \exists P.C & \rightsquigarrow & A \sqsubseteq \exists P.A_1, \quad A_1 \sqsubseteq C
\end{array}$$

   where A_1, A_2 are fresh atomic concepts for each replacement.

The above transformations are satisfiability preserving:

Lemma

Let $\mathcal{T}_2$ be obtained from $\mathcal{T}_1$ by steps (1) and (2) above. Then we have that:

$$A_C \text{ is satisfiable w.r.t. } \mathcal{T}_1 \quad \text{iff} \quad A_C \text{ is satisfiable w.r.t. } \mathcal{T}_2$$

From $\mathcal{ALC}$ to $\mathcal{AL}$: 3. Eliminate qualified exist. restr.

To eliminate qualified existential restrictions from the right-hand side of inclusion assertions in $\mathcal{T}_2$, we proceed as follows:

1. For each $\exists P.A_i$ appearing in $\mathcal{T}_2$, we introduce a fresh atomic role P_{A_i}.
2. We replace assertions as follows:

$$\begin{array}{lcl} A \sqsubseteq \exists P.A_i & \rightsquigarrow & A \sqsubseteq \exists P_{A_i} \sqcap \forall P_{A_i}.A_i \\ A \sqsubseteq \forall P.A' & \rightsquigarrow & A \sqsubseteq \forall P.A' \sqcap \bigsqcap_{P_{A_i}} \forall P_{A_i}.A' \end{array}$$

The above transformations are satisfiability preserving:

Lemma

Let $\mathcal{T}_3$ be obtained from $\mathcal{T}_2$ by steps (1) and (2) above. Then we have that:

A_C is satisfiable w.r.t. $\mathcal{T}_2$ iff A_C is satisfiable w.r.t. $\mathcal{T}_3$

Note: As an intermediate result, we obtain:

Concept satisfiability w.r.t. primitive $\mathcal{ALU}$ TBoxes is ExpTime-hard.

From $\mathcal{ALC}$ to $\mathcal{AL}$: 4. Encode away disjunction

To encode away disjunction in the right-hand side of inclusion assertions in $\mathcal{T}_3$, we replace assertions as follows:

$$A_1 \sqsubseteq A_2 \sqcup A_3 \qquad \leadsto \qquad \neg A_2 \sqcap \neg A_3 \sqsubseteq \neg A_1$$

The two assertions are logically equivalent.

From this, we obtain the desired result:

Concept satisfiability w.r.t. $\mathcal{AL}$ TBoxes is ExpTime-hard.

unibz

Outline of Part 6

Outline of Part 6

unibz

Numeric constraints

- Functionality restrictions $\mathcal{ALCF}$: allow one to impose that a relation is a function:
  - global functionality: $\top \sqsubseteq (\leq 1\, R)$ (equivalent to $(\textbf{funct}\ R)$)
    Example: $\top \sqsubseteq (\leq 1\, \mathsf{hasFather})$
  - local functionality: $A \sqsubseteq (\leq 1\, R)$
    Example: $\mathsf{Car} \sqsubseteq (\leq 1\, \mathsf{hasEngine})$
    (although a ship might have more than one engine)

- Number restrictions $\mathcal{ALCN}$: $(\leq n\, R)$ and $(\geq n\, R)$
  Example: $\mathsf{Person} \sqsubseteq (\leq 2\, \mathsf{hasParent})$

- Qualified Number restrictions $\mathcal{ALCQ}$: $(\leq n\, R.\, C)$ and $(\geq n\, R.\, C)$
  Example:
  $$\begin{aligned}\mathsf{FootballTeam} \sqsubseteq\ & (\geq 1\, \mathsf{hasPlayer}.\, \mathsf{Golly}) \sqcap \\ & (\leq 1\, \mathsf{hasPlayer}.\, \mathsf{Golly}) \sqcap \\ & (\geq 2\, \mathsf{hasPlayer}.\, \mathsf{Defensor}) \sqcap \\ & (\leq 4\, \mathsf{hasPlayer}.\, \mathsf{Defensor})\end{aligned}$$

Role constructs

- Inverse roles $\mathcal{ALCI}$: R^-, interpreted as $(R^-)^\mathcal{I} = \{(y, x) \mid (x, y) \in R^\mathcal{I}\}$
  Example: we can refer to the parent, by using the hasChild role, e.g., $\exists$hasChild$^-$.Doctor.
- Transitive roles: (**trans** R), stating that the relation $R^\mathcal{I}$ is **transitive**, i.e., $\{(x, y), (y, z)\} \subseteq R^\mathcal{I} \rightarrow (x, z) \in R^\mathcal{I}$
  Example: (**trans** hasAncestor)
- Inclusion between roles: $R_1 \sqsubseteq R_2$, used to state that a relation is contained in another relation.
  Example: hasMother $\sqsubseteq$ hasParent

Outline of Part 6

unibz

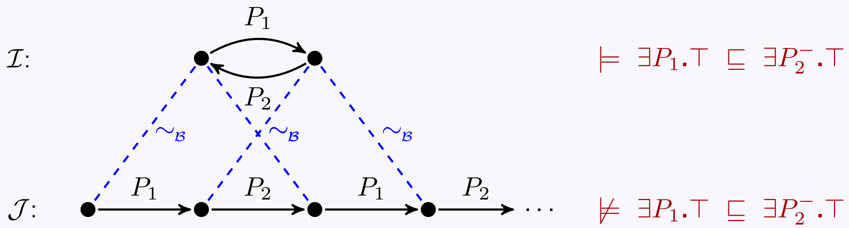
Inverse roles increase the expressive power

Exercise

Prove that the inverse role construct constitutes an effective extension of the expressive power of $\mathcal{ALC}$, i.e., show that $\mathcal{ALC}$ is **strictly less expressive** than $\mathcal{ALCI}$.

Solution

Suggestion: do it via bisimulation. I.e., show that there are two models that are bisimilar but distinguishable in $\mathcal{ALCI}$.



Modeling with inverse roles

Exercise

Try to model the following facts in $\mathcal{ALCI}$.
Notice that not all the statements are modellable in $\mathcal{ALCI}$.

1. Lonely people do not have friends and are not friends of anybody.
2. An intermediate stop is a stop that has a predecessor stop and a successor stop.
3. A person is a child of his father.

Solution

1. $\mathsf{LonelyPerson} \sqsubseteq \mathsf{Person} \sqcap \neg\exists \mathsf{hasFriend}^-.\top \sqcap \neg\exists \mathsf{hasFriend}.\top$
2. $\mathsf{IntermediateStop} \equiv \mathsf{Stop} \sqcap \exists \mathsf{next}.\mathsf{Stop} \sqcap \exists \mathsf{next}^-.\mathsf{Stop}$
3. This cannot be modeled in $\mathcal{ALCI}$.
   Note that $\mathsf{Person} \sqsubseteq \forall \mathsf{hasFather}.(\forall \mathsf{child}.\mathsf{Person})$ is not enough.

Tree model property of $\mathcal{ALCI}$

Theorem (Tree model property)

If C is satisfiable w.r.t. a TBox $\mathcal{T}$, then it is satisfiable w.r.t. $\mathcal{T}$ by a **tree-shaped model** whose root is an instance of C.

Proof (outline).

1. Extend the notion of bisimulation to $\mathcal{ALCI}$.
2. Show that if $(\mathcal{I}, o_1) \sim_{\mathcal{ALCI}} (\mathcal{J}, o_2)$, then $o_1 \in C^{\mathcal{I}}$ iff $o_2 \in C^{\mathcal{J}}$, for every $\mathcal{ALCI}$ concept C.
3. For a non tree-shaped model $\mathcal{I}$ and some element $o_1 \in C^{\mathcal{I}}$, generate a tree-shaped model $\mathcal{J}$ rooted at o_2 and show that $(\mathcal{I}, o_1) \sim_{\mathcal{ALCI}} (\mathcal{J}, o_2)$. □

Bisimulation for $\mathcal{ALCI}$ (tree model property 1)

Def.: $\mathcal{ALCI}$-Bisimulation

An **$\mathcal{ALCI}$-bisimulation** between two $\mathcal{ALCI}$ interpretations $\mathcal{I}$ and $\mathcal{J}$ is a bisimulation $\sim_{\mathcal{B}}$ that satisfies the following additional conditions when $o_1 \sim_{\mathcal{B}} o_2$:

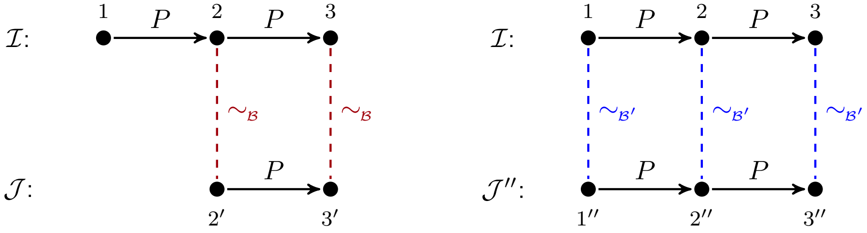
- for each o_1' with $(o_1', o_1) \in P^{\mathcal{I}}$, there is an $o_2' \in \Delta^{\mathcal{J}}$ with $(o_2', o_2) \in P^{\mathcal{J}}$ such that $o_1' \sim_{\mathcal{B}} o_2'$.
- The same property in the opposite direction.

We call these properties the **inverse relation equivalence**.

$(\mathcal{I}, o_1) \sim_{\mathcal{ALCI}} (\mathcal{J}, o_2)$ means that there is an $\mathcal{ALCI}$-bisimulation $\sim_{\mathcal{B}}$ between $\mathcal{I}$ and $\mathcal{J}$ such that $o_1 \sim_{\mathcal{B}} o_2$.

$\mathcal{ALCI}$-bisimulation – Example

Example of bisimulation that is **not** an $\mathcal{ALCI}$-bisimulation, and one that is so.



We have that $(\mathcal{I}, 2) \sim (\mathcal{J}, 2')$ but not $(\mathcal{I}, 2) \sim_{\mathcal{ALCI}} (\mathcal{J}, 2')$.

However, we have that $(\mathcal{I}, 2) \sim_{\mathcal{ALCI}} (\mathcal{J}'', 2'')$.

Invariance under $\mathcal{ALCI}$-bisimulation (tree model prop. 2)

Theorem

If $(\mathcal{I}, o_1) \sim_{\mathcal{ALCI}} (\mathcal{J}, o_2)$, then $o_1 \in C^{\mathcal{I}}$ iff $o_2 \in C^{\mathcal{J}}$, for every $\mathcal{ALCI}$ concept C.

Proof.

By induction on the structure of C.

All the cases are as for $\mathcal{ALC}$, and in addition we have the following case:

- If C is of the form $\exists P^-.C$:

$$\begin{array}{lll} o_1 \in (\exists P^-.C)^{\mathcal{I}} & \text{iff} & o_1' \in C^{\mathcal{I}} \text{ for some } o_1' \text{ with } (o_1', o_1) \in P^{\mathcal{I}} \\ & \text{iff} & o_2' \in C^{\mathcal{J}} \text{ for some } o_2' \text{ with } (o_2', o_2) \in P^{\mathcal{J}} \\ & & \text{and } (\mathcal{I}, o_1') \sim_{\mathcal{ALCI}} (\mathcal{J}, o_2') \\ & \text{iff} & o_2 \in (\exists P^-.C)^{\mathcal{J}} \end{array}$$

□

Transformation into tree-shaped $\mathcal{ALCI}$ models (t.m.p. 3)

Theorem

If $\mathcal{I}$ is a non tree-shaped model, and o is some element of $\Delta^{\mathcal{I}}$, then there is a model $\mathcal{J}$ that is tree-shaped and such that $(\mathcal{I}, o) \sim_{\mathcal{ALCI}} (\mathcal{J}, o)$.

Proof.

We define $\mathcal{J}$ as follows:

- $\Delta^{\mathcal{J}}$ is the **set of paths** $\pi = (o_1, P_1^{(-)}, o_2, \ldots, P_{n-1}^{(-)}, o_n)$ such that $n \geq 1$, $o_1 = o$, and $(o_i, o_{i+1}) \in P_i^{\mathcal{I}}$ or $(o_{i+1}, o_i) \in P_i^{\mathcal{I}}$, for $i \in \{1, \ldots, n-1\}$.
- $A^{\mathcal{J}} = \{\pi o_n \mid o_n \in A^{\mathcal{I}}\}$
- $P^{\mathcal{J}} = \{(\pi o_n \ , \ \pi o_n P o_{n+1}) \mid (o_n, o_{n+1}) \in P^{\mathcal{I}}\} \cup$
  $\{(\pi o_n P^- o_{n+1} \ , \ \pi o_n) \mid (o_{n+1}, o_n) \in P^{\mathcal{I}}\}$

It is easy to show that $\mathcal{J}$ is a tree-shaped model rooted at o.

The $\mathcal{ALCI}$ bisimulation $\sim_{\mathcal{B}}$ between $\mathcal{I}$ and $\mathcal{J}$ is defined as $o_i \sim_{\mathcal{B}} \pi o_i$. □

Outline of Part 6

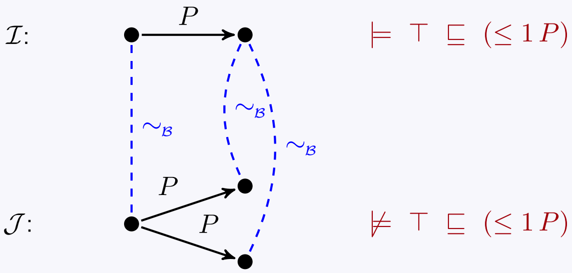
unibz

Number restrictions increase the expressive power

Exercise

Prove that the number restriction construct constitutes an effective extension of the expressive power of $\mathcal{ALC}$, i.e., show that $\mathcal{ALC}$ is **strictly less expressive** than $\mathcal{ALCN}$.

Solution



Qualified number restriction

Exercise

Prove that qualified number restrictions are an effective extension of the expressivity of $\mathcal{ALCN}$, i.e., show that $\mathcal{ALCN}$ is **strictly less expressive** than $\mathcal{ALCQ}$.

Solution (outline)

1. Define a notion of bisimulation that is appropriate for $\mathcal{ALCN}$.
2. Prove that $\mathcal{ALCN}$ is bisimulation invariant for the bisimulation relation defined in item 1.
3. Prove that $\mathcal{ALCN}$ is strictly less expressive than $\mathcal{ALCQ}$.

unibz

Bisimulation for $\mathcal{ALCN}$

Def.: $\mathcal{ALCN}$-bisimulation

An **$\mathcal{ALCN}$-bisimulation** between two $\mathcal{ALCN}$ interpretations $\mathcal{I}$ and $\mathcal{J}$ is a bisimulation $\sim_{\mathcal{B}}$ that satisfies the following additional conditions when $o_1 \sim_{\mathcal{B}} o_2$:

- if $o_1^1, \ldots, o_1^n$ are all the distinct elements in $\Delta^{\mathcal{I}}$ such that $(o_1, o_1^k) \in P^{\mathcal{I}}$, for $k \in \{1, \ldots, n\}$, then there are exactly n elements $o_2^1, \ldots, o_2^n$ in $\Delta^{\mathcal{J}}$ such that $(o_2, o_2^k) \in P^{\mathcal{J}}$, for $k \in \{1, \ldots, n\}$.
- The same property in the opposite direction.

We call these properties the **relation cardinality equivalence**.

$(\mathcal{I}, o_1) \sim_{\mathcal{ALCN}} (\mathcal{J}, o_2)$ means that there is an $\mathcal{ALCN}$-bisimulation $\sim_{\mathcal{B}}$ between $\mathcal{I}$ and $\mathcal{J}$ such that $o_1 \sim_{\mathcal{B}} o_2$.

Invariance under $\mathcal{ALCN}$-bisimulation

Theorem

If $(\mathcal{I}, o_1) \sim_{\mathcal{ALCN}} (\mathcal{J}, o_2)$, then $o_1 \in C^{\mathcal{I}}$ iff $o_2 \in C^{\mathcal{J}}$, for every $\mathcal{ALCN}$ concept C.

Proof.

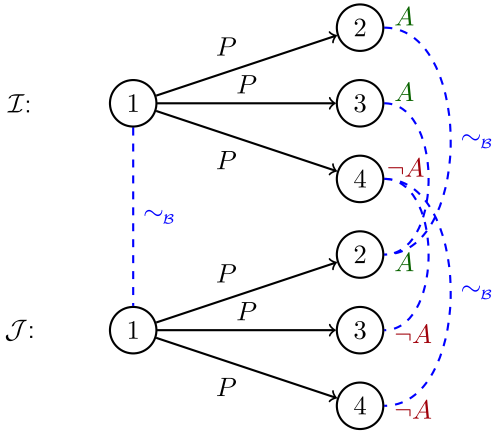
By induction on the structure of C.

All the cases are as for $\mathcal{ALC}$, and in addition we have the following base case:

- If C is of the form $(\leq n\, P)$:
  - If $o_1 \in (\leq n\, P)^{\mathcal{I}}$, then there are $m \leq n$ elements $o_1^1, \ldots, o_1^m$ with $(o_1, o_1^i) \in P^{\mathcal{I}}$.
  - The additional condition on $\mathcal{ALCN}$-bisimulation implies that there are exactly m elements $o_2^1, \ldots, o_2^m$ in $\Delta^{\mathcal{J}}$ such that $(o_2, o_2^i) \in P^{\mathcal{J}}$.
  - This implies that $o_2 \in (\leq n\, P)^{\mathcal{J}}$. $\square$

$\mathcal{ALCN}$ is strictly less expressive than $\mathcal{ALCQ}$

We show that in $\mathcal{ALCQ}$ we can distinguish two models that are $\mathcal{ALCN}$-bisimilar, and hence not distinguishable in $\mathcal{ALCN}$.



$\models \top \sqsubseteq (\leq 1\, P.\neg A)$

$\not\models \top \sqsubseteq (\leq 1\, P.\neg A)$

Outline of Part 6

Encoding $\mathcal{ALCN}$ into $\mathcal{ALCFI}$

We encode away number restrictions by using functionality and inverse roles.
To do so, given an $\mathcal{ALCN}$ concept C and a TBox $\mathcal{T}$, we define:

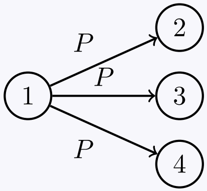
- a set $\mathcal{T}_r$ of $\mathcal{ALCFI}$-axioms, and
- a transformation π from $\mathcal{ALCN}$-concepts to $\mathcal{ALCFI}$-concepts

such that:

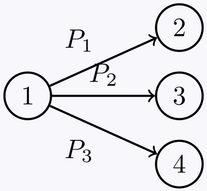
C is satisfiable w.r.t. $\mathcal{T}$ in $\mathcal{ALCN}$ iff
$\pi(C)$ is satisfiable w.r.t. $\pi(\mathcal{T}) \cup \mathcal{T}_r$ in $\mathcal{ALCFI}$

Intuition

Replace role P with $P_1, \ldots, P_n$, which count the number of P-successors.



$1 \models (\leq 3\, P)$
$1 \models \neg(\geq 4\, P)$



$1 \models \exists P_1.\top$
$1 \models \exists P_2.\top$
$1 \models \exists P_3.\top$
$1 \models \neg\exists P_4.\top$

Encoding $\mathcal{ALCN}$ into $\mathcal{ALCFI}$ (cont'd)

We assume C and all concepts in $\mathcal{T}$ to be in NNF, where
$\text{NNF}(\neg(\geq m\,P)) = (\leq m{-}1\,P)$ and $\text{NNF}(\neg(\leq m\,P)) = (\geq m{+}1\,P)$.

Let n_{max} be the maximum number occurring in a number restriction of C or $\mathcal{T}$.

We proceed as follows:

1. For every role P, introduce fresh roles $P_1, \ldots, P_{n_{max}+1}$.
2. For every role P_i, the TBox $\mathcal{T}_r$ contains the following axioms:
   1. $\exists P_{i+1}.\top \sqsubseteq \exists P_i.\top$, for $i \in \{1, \ldots, n_{max}\}$
   2. $\top \sqsubseteq (\leq 1\, P_i)$, for $i \in \{1, \ldots, n_{max}\}$ (NB: $P_{n_{max}+1}$ is not functional)
   3. $\top \sqsubseteq \forall P_i.\forall P_j^-.\bot$, for $i, j \in \{1, \ldots, n_{max}\}$, $i \neq j$.
3. $\pi(C)$ is defined by induction on the structure of C:

$$\begin{array}{rcl@{\qquad}rcl}
\pi(A) & = & A & \pi(C_1 \sqcap C_2) & = & \pi(C_1) \sqcap \pi(C_2) \\
\pi(\neg A) & = & \neg A & \pi(C_1 \sqcup C_2) & = & \pi(C_1) \sqcup \pi(C_2) \\
\pi((\geq m\,P)) & = & \exists P_m.\top & \pi((\leq m\,P)) & = & \forall P_{m+1}.\neg\top \\
\pi(\exists P.C) & = & \multicolumn{4}{l}{\exists P_1.\pi(C) \sqcup \cdots \sqcup \exists P_{n_{max}+1}.\pi(C)} \\
\pi(\forall P.C) & = & \multicolumn{4}{l}{\forall P_1.\pi(C) \sqcap \cdots \sqcap \forall P_{n_{max}+1}.\pi(C)}
\end{array}$$

4. $\pi(\mathcal{T}) = \bigcup_{C \sqsubseteq D \in \mathcal{T}} \{\pi(C) \sqsubseteq \pi(D)\}$

Encoding $\mathcal{ALCN}$ into $\mathcal{ALCFI}$ (cont'd)

We have to prove that if C is satisfiable w.r.t. $\mathcal{T}$, then $\pi(C)$ is satisfiable w.r.t. $\mathcal{T}_r \cup \pi(\mathcal{T})$.

1. If C is satisfiable in $\mathcal{ALCN}$, then it has a tree-shaped model $\mathcal{I}$.
2. Extend $\mathcal{I}$ into $\mathcal{J}$ with the interpretation of $P_1, \ldots, P_{n_{max}+1}$ as follows. For each $o \in \Delta^{\mathcal{I}}$, let $P^{\mathcal{I}}(o) = \{o_1, \ldots, o_m, \ldots\}$ be the set of P-successors of o in $\mathcal{I}$. Then:
   - if $|P^{\mathcal{I}}(o)| < n_{max}$, then add (o, o_i) to $P_i^{\mathcal{J}}$, for $i \in \{1, \ldots, |P^{\mathcal{I}}(o)|\}$.
   - if $|P^{\mathcal{I}}(o)| \geq n_{max}$, then add (o, o_i) to $P_i^{\mathcal{J}}$, for $i \in \{1, \ldots, n_{max}\}$, and also add (o, o_j) to $P_{n_{max}+1}^{\mathcal{J}}$ for $j \geq n_{max} + 1$
3. Prove that $\mathcal{J}$ is a model of $\mathcal{T}_r$.
4. Prove that $\mathcal{J}$ is a model of $\pi(C)$.

Encoding $\mathcal{ALCN}$ into $\mathcal{ALCFI}$ (cont'd)

Finally we have to prove that if $\pi(C)$ is satisfiable w.r.t. $\mathcal{T}_r \cup \pi(\mathcal{T})$, then C is satisfiable wrt $\mathcal{T}$.

1. Let $\mathcal{J}$ be a tree-shaped model of $\mathcal{T}_r \cup \pi(\mathcal{T})$ that satisfies C.
2. Let $\mathcal{I}$ be obtained by extending $\mathcal{J}$ with the interpretation of each role P as follows:
$$P^{\mathcal{I}} = P_1^{\mathcal{I}} \cup \cdots \cup P_{n+1}^{\mathcal{I}}$$
3. Prove by structural induction that $\mathcal{I}$ is a model of $\mathcal{T}$ that satisfies C.

Outline of Part 6

unibz

Role hierarchy: $\mathcal{H}$

Def.: **Role Hierarchy**

A role hierarchy $\mathcal{H}$ is a finite set of **role inclusion assertions**, i.e., expressions of the form

$$R_1 \sqsubseteq R_2$$

for roles R_1 and R_2.
We say that R_1 is a **subrole** of R_2.

Exercise

Explain why the role inclusion $R_1 \sqsubseteq R_2$ cannot be axiomatized by the concept inclusions:

$$\begin{array}{rcl} \exists R_1.\top & \sqsubseteq & \exists R_2.\top \\ \exists R_1^-.\top & \sqsubseteq & \exists R_2^-.\top \end{array}$$

Transitive roles: $\mathcal{S}$

Def.: Transitive roles

(**trans** P) declares a role to be **transitive**. $\mathcal{I} \models$ (**trans** P) if $P^{\mathcal{I}}$ is a transitive relation, i.e., for all $x, y, z \in \Delta^{\mathcal{I}}$, if $\{(x, y), (y, z) \subseteq P^{\mathcal{I}}$, then $(x, z) \in P^{\mathcal{I}}$.

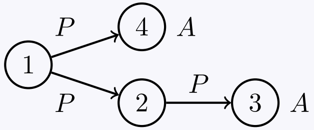
Note: if a role P is transitive, also P^- is transitive. Hence, we can restrict transitivity assertions to atomic roles only without losing expressive power.

Exercise

Explain why transitive roles cannot be axiomatized by the inclusion assertion

$$\exists P.(\exists P.A) \sqsubseteq \exists P.A$$

Solution



This interpretation satisfies the assertion $\exists P.(\exists P.A) \sqsubseteq \exists P.A$, but P is **not transitive**.

Outline of Part 6

TBox internalization

Until now we have distinguished between the following two problems:

- Satisfiability of a concept C, and
- Satisfiability of a concept C w.r.t. a TBox $\mathcal{T}$.

Clearly the first problem is a special case of the second.

For expressive concept languages, satisfiability w.r.t. a TBox can be reduced to concept satisfiability, i.e., the TBox can be internalized:

Def.: **Internalization** of the TBox

For a description logic $\mathcal{L}$, we say that the TBox can be **internalized**, if the following holds:
For every $\mathcal{L}$-TBox $\mathcal{T}$ one can construct an $\mathcal{L}$-concept $C_{\mathcal{T}}$ such that, for every $\mathcal{L}$-concept C, we have that C is satisfiable w.r.t. $\mathcal{T}$ iff $C \sqcap C_{\mathcal{T}}$ is satisfiable.

Note: This is similar to propositional or first order logic, where the problem of checking $\Gamma \models \phi$ (validity under a finite set of axioms Γ) reduces to the problem of checking the validity of a single formula, i.e., $\bigwedge \Gamma \rightarrow \phi$.

TBox internalization for logics including $\mathcal{SH}$

A role hierarchy and transitive roles are sufficient for internalization.

Theorem (TBox internalization for $\mathcal{SH}$)

Let $\mathcal{T} = \{C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n\}$ be a finite set of concept inclusion assertions, and let

$$C_\mathcal{T} = \bigsqcap_{i=1}^{n} \neg C_i \sqcup D_i$$

Let U be a fresh **transitive** role, and let

$$\mathcal{R}_U = \{P \sqsubseteq U \mid P \text{ is a role appearing in } C \text{ or } \mathcal{T}\}$$

Then C is satisfiable w.r.t. $\mathcal{T}$ iff $C \sqcap C_\mathcal{T} \sqcap \forall U.C_\mathcal{T}$ is satisfiable w.r.t. $\mathcal{R}_U$.

One can adopt also other internalization mechanisms:

- exploiting reflexive transitive closure of roles;
- exploiting nominals.

Outline of Part 6

Outline of Part 6

unibz

Expansion rules for $\mathcal{ALCI}$

We need to extend the expansion rules dealing with quantification over roles to the case where the role might be an inverse.

Expansion rules for satisfiability of $\mathcal{ALCI}$ ontologies		
$\sqcap$-rule	if	$C_1 \sqcap C_2 \in \mathcal{L}(v)$ and $\{C_1, C_2\} \nsubseteq \mathcal{L}(v)$
	then	$\mathcal{L}(v) := \mathcal{L}(v) \cup \{C_1, C_2\}$
$\sqcup$-rule	if	$C_1 \sqcup C_2 \in \mathcal{L}(v)$ and $\{C_1, C_2\} \cap \mathcal{L}(v) = \emptyset$
	then	$\mathcal{L}(v) := \mathcal{L}(v) \cup \{D\}$ for some $D \in \{C_1, C_2\}$
$\exists$-rule	if	$\exists P.C \in \mathcal{L}(v)$, v **is not blocked**, and there is no w such that $P \in \mathcal{L}(v, w)$ and $C \in \mathcal{L}(w)$
	then	create a new node w and an edge (v, w), and set $\mathcal{L}(v, w) := \{P\}$ and $\mathcal{L}(w) := \{C\}$
$\exists^-$-rule	if	$\exists P^-.C \in \mathcal{L}(v)$, v **is not blocked**, and there is no w such that $P \in \mathcal{L}(w, v)$ and $C \in \mathcal{L}(w)$
	then	create a new node w and an edge (w, v), and set $\mathcal{L}(w, v) := \{P\}$ and $\mathcal{L}(w) := \{C\}$
$\forall$-rule	if	$\forall P.C \in \mathcal{L}(v)$, and there is some w such that $P \in \mathcal{L}(v, w)$ and $C \notin \mathcal{L}(w)$
	then	$\mathcal{L}(w) := \mathcal{L}(w) \cup \{C\}$
$\forall^-$-rule	if	$\forall P^-.C \in \mathcal{L}(v)$, and there is some w such that $P \in \mathcal{L}(w, v)$ and $C \notin \mathcal{L}(w)$
	then	$\mathcal{L}(w) := \mathcal{L}(w) \cup \{C\}$
$\mathcal{T}$-rule	if	$C_{\mathcal{T}} \notin \mathcal{L}(v)$
	then	$\mathcal{L}(v) := \mathcal{L}(v) \cup \{C_{\mathcal{T}}\}$

In addition, we need to adopt a suitable **blocking strategy**, given that we are dealing with an arbitrary set of inclusion assertions.

Tableau for $\mathcal{ALCI}$ – Example

Example

Satisfiability of $C_0 = A \sqcap \exists P.A \sqcap \forall P^-.\neg A$ w.r.t. the TBox $\mathcal{T} = \{\top \sqsubseteq B\}$.

Solution

$$\begin{tikzcd}
x \arrow[d, "P"] & \mathcal{L}(x) = \{C_0,\ A,\ \exists P.A,\ \forall P^-.\neg A,\ B\} \\
y & \mathcal{L}(y) = \{A,\ B\},\ \text{and } y \text{ is blocked by } x
\end{tikzcd}$$

$$\begin{tikzcd}
A,\ B \\
x \arrow[loop right, "P"] \\
\exists P.A,\quad \forall P^-.\neg A
\end{tikzcd}$$

Problem: x is not an instance of the concept $\forall P^-.\neg A$, hence we have not obtained a model of C.

The reason for the problem is that we have adopted a **too weak blocking strategy**.

Blocking strategy for $\mathcal{ALCI}$

For $\mathcal{ALCI}$, subset-blocking, where the blocking condition is $\mathcal{L}(x) \subseteq \mathcal{L}(y)$, is no longer sufficient. We need to adopt a stronger blocking strategy.

Def.: Equality blocking

A node v is called **directly blocked** if it has an ancestor w with $\mathcal{L}(v) = \mathcal{L}(w)$.

For the previous example

$$\begin{tikzcd} x \arrow[d, "P"] & \mathcal{L}(x) = \{C_0,\ A,\ \exists P.A,\ \forall P^-.\neg A,\ B\} \\ y & \mathcal{L}(y) = \{A,\ B\},\ y \textbf{ is not blocked by } x \end{tikzcd}$$

$$\begin{tikzcd} A,\ B & A,\ B \\ x \arrow[r, "P"] & y \\ \exists P.A,\ \forall P^-.\neg A & \end{tikzcd}$$

Decidability of $\mathcal{ALCI}$

Theorem

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ALCI}$ ontology, where $\mathcal{T}$ is a general TBox. Then:

1. The tableau algorithm terminates when applied to $\mathcal{O}$.
2. The rules can be applied such that they generate a clash-free and complete completion tree iff $\mathcal{O}$ is satisfiable.

Corollary

- Satisfiability of $\mathcal{ALCI}$ ontologies is **decidable**.
- $\mathcal{ALCI}$ has the **finite model property**.

Correctness of tableau algorithm for $\mathcal{ALCI}$

- **Termination:** As for $\mathcal{ALC}$.
- **Soundness:** If the algorithm generates a complete and clash-free completion graph $G = \langle V, E, \mathcal{L} \rangle$ for $\mathcal{O}$, then $\mathcal{O}$ is satisfiable.

  Indeed, the following interpretation $\mathcal{I}_G = (\Delta^{\mathcal{I}_G}, \cdot^{\mathcal{I}_G})$ is a model of $\mathcal{O}$:
  - $\Delta^{\mathcal{I}_G} = \{v \mid v \in V$ and v is not blocked$\}$
  - Each Individual c is interpreted as the corresponding initial node, i.e., $c^{\mathcal{I}} = \hat{c}$.
  - $A^{\mathcal{I}_G} = \{v \mid v \in \Delta^{\mathcal{I}_G}$ and $A \in \mathcal{L}(v)\}$, for each concept name A.
  - $P^{\mathcal{I}_G} = \{(v, w) \mid \{v, w\} \subseteq \Delta^{\mathcal{I}_G}$ and $P \in \mathcal{L}(x, y)\} \cup$
    $\{(v, w) \mid v \in \Delta^{\mathcal{I}_G},\ P \in \mathcal{L}(v, w'),$ and w' is blocked by $w\} \cup$
    $\{(v, w) \mid w \in \Delta^{\mathcal{I}_G},\ P \in \mathcal{L}(v', w),$ and v' is blocked by $v\}$,
    for each role name P.
- **Completeness:** Given a model $\mathcal{I}$ of $\mathcal{O}$, we can use it to steer the application of the non-deterministic rule for $\sqcup$.
  At the end we obtain a complete and clash-free completion graph that generates a model $\mathcal{J}$ that is bisimilar to the initial model $\mathcal{I}$.

Outline of Part 6

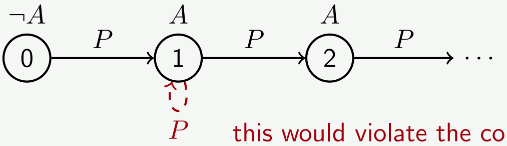
$\mathcal{ALCQI}$ and finite models

$\mathcal{ALCQI}$ with general TBoxes does **not** have the **finite model property**.

Example ($\mathcal{ALCQI}$ ontology satisfiable only in infinite models)

Consider satisfiability of the ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{A} = \{\neg A(x_0)\}$ and $\mathcal{T} = \{\top \sqsubseteq \exists P.A \sqcap (\leq 1\, P^-.\top)\}$.

$\mathcal{O}$ is satisfied only in an infinite model.



this would violate the condition $(\leq 1\, P^-.\top)$

Completion rules for number restrictions – Intuition

Consider a completion graph $G = \langle V, E, \mathcal{L} \rangle$, and a (direct or inverse) role R.
An R-**neighbour** of a node $v \in V$ is a node $w \in V$ such that:

- if $R = P$, then $(v, w) \in E$ and $P \in \mathcal{L}(v, w)$, and
- if $R = P^-$, then $(w, v) \in E$ and $P \in \mathcal{L}(w, v)$.

To deal with:

$(\geq n\, R. C)$: If a node v does not have n R-neighbours satisfying C, **new nodes** satisfying C are created and made R-neighbours of v.

$(\leq n\, R. C)$: If a node has more than n R-neighbours satisfying C, then two of them are **non-deterministically chosen** and merged by merging their labels and the subtrees in the completion tree rooted at these nodes.

The correct form of the completion rules is complicated by the following facts:

- They need to take into account blocking.
- For a node it might not be known whether it actually satisfies C or not.
- One needs to avoid jumping back and forth between merging and creating new nodes in the presence of potentially conflicting number restrictions.

Expansion rules for qualified number restrictions

Let us consider the following two rules:

$\geq$-rule	if	$(\geq n\,R.C) \in \mathcal{L}(v)$, v is not blocked, and
		there are less than n R-neighbours w such that $C \in \mathcal{L}(w)$
	then	create n new nodes $w_1, \ldots, w_n$, make them R-neighbours of v, and
		set $\mathcal{L}(w_i) := \{C\}$, for $1 \leq i \leq n$
$\leq$-rule	if	$(\leq n\,R.C) \in \mathcal{L}(v)$, v is not indirectly blocked,
		there are $n+1$ R-neighbours $w_0, \ldots, w_n$ of v
		with $C \in \mathcal{L}(w_i)$, for $0 \leq i \leq n$, and
		there are i, j such that w_j is not an ancestor of w_i
	then	set $\mathcal{L}(w_i) := \mathcal{L}(w_i) \cup \mathcal{L}(w_j)$,
		for each edge (w_j, z), add an edge (w_i, z) with $\mathcal{L}(w_i, z) := \mathcal{L}(w_j, z)$,
		and remove w_j from the tree

However, the rules in this form are problematic, since they might cause nodes to be repeatedly created and merged (**"yoyo"-effect**).

unibz

Dealing with "yoyo"-effect

To prevent the "yoyo"-effect, the algorithm maintains also a **set of explicit inequalities** between nodes:

$\geq$-rule	if	$(\geq n\, R.\, C) \in \mathcal{L}(v)$, v is not blocked, and
		there are less than n R-neighbours w such that $C \in \mathcal{L}(w)$
	then	create n new nodes $w_1, \ldots, w_n$, make them R-neighbours of v,
		set $\mathcal{L}(w_i) := \{C\}$, for $1 \leq i \leq n$, and
		set $w_i \neq w_j$**, for** $1 \leq i < j \leq n$
$\leq$-rule	if	$(\leq n\, R.\, C) \in \mathcal{L}(v)$, v is not indirectly blocked,
		there are $n+1$ R-neighbours $w_0, \ldots, w_n$ of v
		with $C \in \mathcal{L}(w_i)$, for $0 \leq i \leq n$, and
		there are i, j such that w_j is not an ancestor of w_i and **not** $w_i \neq w_j$
	then	set $\mathcal{L}(w_i) := \mathcal{L}(w_i) \cup \mathcal{L}(w_j)$,
		for each edge (w_j, z), add an edge (w_i, z) with $\mathcal{L}(w_i, z) := \mathcal{L}(w_j, z)$,
		for each node z **with** $w_j \neq z$**, add** $w_i \neq z$, and
		remove w_j from the tree

unibz

Clash for number restrictions

Number restrictions may give rise to an additional form of immediate contradiction. Hence, we add to the clash conditions also the following one:

Def.: Clash for number restrictions

A node v contains a clash if

- $(\leq n\, R.\, C) \in \mathcal{L}(v)$, and
- v has more than n R-neighbours $w_0, \ldots, w_n$ with $w_i \neq w_j$, for $0 \leq i < j \leq n$.

However, this does not suffice!

E.g., $(\leq 1\, R.\, A) \sqcap (\leq 1\, R.\, \neg A) \sqcap (\geq 3\, R.\, B)$ is unsatisfiable, but the algorithm would answer “satisfiable”.

Reason: if $(\leq n\, R.\, C) \in \mathcal{L}(x)$ and x has an R-neighbour y, we need to know whether y is an instance of C or of $\neg C$.

Choice rule

To solve the problem, we proceed as follows:

1. We extend the set of node labels to

$$Cl(C_0, \mathcal{T}) = sub(C_0, \mathcal{T}) \cup \{\dot{\neg} C \mid C \in sub(C_0, \mathcal{T})\},$$

   where:

   - $\dot{\neg} C$ denotes the NNF of $\neg C$, and
   - $sub(C_0, \mathcal{T})$ denotes the set of subconcepts of C_0 and of all concepts in $\mathcal{T}$.

2. We add an additional non-deterministic expansion rule: the **choice rule**:

?-rule:	if	$(\leq n\, R.\, C) \in \mathcal{L}(v)$, v is not indirectly blocked, and there is an R-neighbour w of v with $\{C, \dot{\neg} C\} \cap \mathcal{L}(w) = \emptyset$
	then	$\mathcal{L}(w) := \mathcal{L}(w) \cup \{E\}$ for some $E \in \{C, \dot{\neg} C\}$

However, this still does not suffice!

The reason is that equality blocking is too weak.

Problem with blocking strategy – Example

Consider the tableau for satisfiability of C_0 w.r.t. a TBox $\mathcal{T}$, where

$$\begin{array}{rcl} C_0 & = & \neg A \sqcap \exists P.D \\ D & = & A \sqcap (\leq 1\, P^-) \sqcap \exists P^-.\neg A \\ \mathcal{T} & = & \{\top \sqsubseteq \exists P.D\} \end{array}$$

$$\begin{tikzcd}
x \arrow[d, "P"] & \mathcal{L}(x) = \{C_0,\ \neg A,\ \exists P.D\} \\
y \arrow[loop left, "P"] \arrow[d, "P"] & \mathcal{L}(y) = \{D,\ A,\ (\leq 1\, P^-),\ \exists P^-.\neg A,\ \exists P.D\} \\
z & \mathcal{L}(z) = \{D,\ A,\ (\leq 1\, P^-),\ \exists P^-.\neg A,\ \exists P.D\}
\end{tikzcd}$$

$\mathcal{L}(z) = \mathcal{L}(y)$, so y would block z. But we cannot construct a model from this.

Blocking strategy and tableau algorithm for $\mathcal{ALCQI}$

Let $G = \langle V, E, \mathcal{L} \rangle$ be the expansion graph constructed by the tableau algorithm.

Def.: **Double blocking**

A node w is directly blocked if there are ancestors v, v', and w' of w such that:

- v is predecessor of w, and v' is predecessor of y'.
- $\mathcal{L}(v, w) = \mathcal{L}(v', w')$,
- $\mathcal{L}(v) = \mathcal{L}(v')$, and $\mathcal{L}(w) = \mathcal{L}(w')$.

Lemma

Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ALCQI}$ ontology (where $\mathcal{T}$ is a general TBox. Then:

1. The tableau algorithm terminates when applied to $\mathcal{O}$.
2. The rules can be applied such that they generate a clash-free and complete completion tree iff $\mathcal{O}$ is satisfiable.

unibz

Tableau algorithm for $\mathcal{ALCQI}$ – Correctness

Termination: The tree is no longer built monotonically, but $\neq$ prevents "yoyo"-effect.

Soundness: a complete, clash-free tree can be "unravelled" into an (infinite tree) model.

- Elements of the model are **paths** starting from the root.
  - Instead of going to a blocked node, go to its blocking node.
  - $p \in A^{\mathcal{I}}$ if $A \in \mathcal{L}(\mathbf{Tail}(p))$
  - Roughly speaking, set $(p, p|w) \in P^{\mathcal{I}}$ if w is a P-successor of $\mathbf{Tail}(p)$ (and similar for inverse roles), taking care of blocked nodes.
- Danger: assume two successors w, w' of v are blocked by the same node x:
  - Standard unravelling yields one path $[\ldots\ vx]$ for both nodes.
  - Hence, $[\ldots\ v]$ might not have enough P-successors for some $(\geq n\, R.\, C) \in \mathcal{L}(v)$.
  - Solution: annotate points in the path with blocked nodes: $[\ldots\ \frac{v}{v}\frac{x}{w}] \neq [\ldots\ \frac{v}{v}\frac{x}{w'}]$

Completeness: Identical to the proof for $\mathcal{ALCI}$, but for stricter invariance condition on mapping π from model to tableau.

Tableau algorithm for $\mathcal{SHIQ}$

$\mathcal{SHIQ}$ extends $\mathcal{ALCI}$ with role hierarchies and transitive roles:

- Roles in number restrictions are simple, i.e., don't have transitive subroles.
- If (**trans** S) and $R \sqsubseteq S$, then $S^{\mathcal{I}}$ is a transitive relation containing $R^{\mathcal{I}}$.

The additional constructs need to be taken into account in the tableau algorithm:

- The relational structure of the completion tree is only a "skeleton" (Hasse Diagram) of the relational structure of the model to be built.
  Specifically, **transitive edges are left out**.
- Also **edges** in the tree-shaped part are **labeled with sets of role names**.
  Example: Consider $\{S_1 \sqsubseteq P, S_2 \sqsubseteq P\} \subseteq \mathcal{T}$. A node satisfying $(\leq 1\, P) \sqcap (\geq 1\, S_1 . A) \sqcap (\geq 1\, S_2 . B)$ must have an outgoing edge labeled both with S_1 and with S_2.
- **To deal with transitivity, it suffices to propagate $\forall$ restrictions.**
  Specifically, if $\forall S.C \in \mathcal{L}(x)$, $R \in \mathcal{L}(x, y)$, $R \sqsubseteq S$, and (**trans** R), then $\forall R.C \in \mathcal{L}(y)$.

Outline of Part 6

unibz

Outline of Part 6

Nominals (a.k.a. objects) $\mathcal{O}$

In many cases it is convenient to define a set (concept) by **explicitly enumerating** its members.

Example

$$\begin{aligned}\mathsf{WeekDay} \equiv \{\ & \mathsf{friday}, \mathsf{monday}, \mathsf{saturday}, \mathsf{sunday},\\ & \mathsf{thursday}, \mathsf{tuesday}, \mathsf{wednesday}\ \}\end{aligned}$$

Def.: Nominals

A **nominal** is a concept representing a singleton set.

- If o is an individual, the expression $\{o\}$ is a concept, called **nominal**.
- The expression $\{o_1, \ldots, o_n\}$ for $n \geq 0$ denotes:
  - $\bot$, if $n = 0$, and
  - $\{o_1\} \sqcup \cdots \sqcup \{o_n\}$, if $n > 0$.

Semantics of nominals

The interpretation of a nominal, i.e., $\{o\}^{\mathcal{I}}$, is the singleton set $\{o^{\mathcal{I}}\}$.
As a consequence:

$$\{o_1, \ldots, o_n\}^{\mathcal{I}} = \{o_1^{\mathcal{I}}, \ldots, o_n^{\mathcal{I}}\}$$

Exercise (Modeling with Nominals:)

Express, in term of subsumptions between concepts, the following statements, using nominals, and all the DL constructs you studied so far:

1. There are **exactly 195 Countries**.
2. Alice loves Bob or Calvin.
3. Either John or Mary is a spy.
4. Everything is created by God.
5. Every person drives on the left or every person drives on the right.
6. $(\exists x.A(x)) \rightarrow (\forall x.B(x))$.

Exercise on nominals

1. There are exactly 195 Countries.

$$\begin{array}{l} \mathsf{Country} \equiv \{\mathsf{afghanistan}, \mathsf{albania}, \ldots, \mathsf{zimbabwe}\} \\ \{\mathsf{afghanistan}\} \sqsubseteq \neg\{\mathsf{albania}\}, \ldots, \{\mathsf{afghanistan}\} \sqsubseteq \neg\{\mathsf{zimbabwe}\} \\ \{\mathsf{albania}\} \sqsubseteq \neg\{\mathsf{algeria}\}, \ldots, \{\mathsf{albania}\} \sqsubseteq \neg\{\mathsf{zimbabwe}\} \\ \ldots \end{array}$$

2. Alice loves Bob or Calvin.

$$\{\mathsf{alice}\} \sqsubseteq \exists \mathsf{loves}.\{\mathsf{bob}, \mathsf{calvin}\}$$

3. Either John or Mary is a spy (but not both of them).

$$\begin{array}{rcl} \{\mathsf{john}\} & \sqsubseteq & \neg\{\mathsf{mary}\} \\ \{\mathsf{johnOrMary}\} & \sqsubseteq & \{\mathsf{john}, \mathsf{mary}\} \\ \{\mathsf{johnOrMary}\} & \sqsubseteq & \mathsf{Spy} \end{array} \qquad \begin{array}{rcl} \{\mathsf{johnOrMary}\} & \sqsubseteq & \neg\{\mathsf{johnOrMary2}\} \\ \{\mathsf{johnOrMary2}\} & \sqsubseteq & \{\mathsf{john}, \mathsf{mary}\} \\ \{\mathsf{johnOrMary2}\} & \sqsubseteq & \neg\mathsf{Spy} \end{array}$$

Exercise on nominals (cont'd)

4. Everything is created by God.

$$\top \sqsubseteq \exists \mathsf{creates}^{-}.\{\mathsf{god}\}$$

In this case god is called **spy point**, as every object of the domain can be observed (and predicated) by "god" through the relation "creates". Spy points allows for universal/existential quantification over the full domain.

5. Every person drives on the left or every person drives on the right.

$$\begin{aligned} \top &\sqsubseteq \exists \mathsf{creates}^{-}.\{\mathsf{god}\} \\ \{\mathsf{god}\} &\sqsubseteq \forall \mathsf{creates}.(\neg \mathsf{Person} \sqcup \mathsf{LeftDriver}) \sqcup \\ & \forall \mathsf{creates}.(\neg \mathsf{Person} \sqcup \mathsf{RightDriver}) \end{aligned}$$

6. $(\exists x.A(x)) \rightarrow (\forall x.B(x))$

$$\begin{aligned} \top &\sqsubseteq \exists \mathsf{creates}^{-}.\{\mathsf{god}\} \\ \{\mathsf{god}\} &\sqsubseteq \neg \exists \mathsf{creates}.A \sqcup \forall \mathsf{creates}.B \end{aligned}$$

unibz

Encoding ABoxes into TBoxes

Using nominals, one can immediately encode an ABox into a TBox:

- $C(a)$ becomes $\{a\} \sqsubseteq C$.
- $R(a, b)$ becomes $\{a\} \sqsubseteq \exists R.\{b\}$.

Note:

- Reasoning with nominals is in general much more complicated than reasoning with an ABox.
- State-of-the-art DL reasoners that are able to deal with nominals, process anyway ABox assertions in a very different way than TBox assertions involving nominals.
- However, this simple encoding of an ABox into a TBox is useful for theoretical purposes, and applies essentially to all DLs.

Outline of Part 6

Boolean TBoxes

Def.: **Boolean TBox**

A Boolean TBox is a propositional formula whose atomic components are concept inclusions. More formally:

- $C \sqsubseteq D$ is a boolean TBox, for every pair of concepts C and D.
- If α and β are boolean TBoxes, then so are $\neg\alpha$, $\alpha \land \beta$, $\alpha \lor \beta$ and $\alpha \to \beta$.

Example

$$\neg(\text{Driver} \sqsubseteq \text{Pilot}) \land ((\text{Driver} \sqsubseteq \text{LeftDriver}) \lor (\text{Driver} \sqsubseteq \text{RightDriver}))$$

This Boolean TBox states that not all drivers are pilots and that either all drivers drive on the left or all drivers drive on the right side of the road.

Internalizing boolean TBoxes using nominals

Theorem

In $\mathcal{ALCOI}$, a boolean TBox φ can be transformed into an equivalent standard TBox $\mathcal{T}_\varphi$.

Proof.

W.l.o.g., we can assume that φ is in CNF (w.r.t. the boolean operators), i.e., φ is a conjunction of clauses, where each clause c in φ is of the form:

$$c \;=\; \bigvee_{i=1}^{n}(C_i \sqsubseteq C_i') \;\vee\; \bigvee_{j=1}^{m}\neg(D_j \sqsubseteq D_j')$$

Let P_{cr} be a new role and spy a new object, not appearing in φ.
$\mathcal{T}_\varphi$ is the TBox that contains the inclusion $\top \sqsubseteq \exists P_{cr}^-.\{spy\}$ (i.e., spy is a **spy point**) and the following inclusion, for every clause c in φ:

$$\{spy\} \;\sqsubseteq\; \bigsqcup_{i=1}^{n}(\forall P_{cr}.(\neg C_i \sqcup C_i')) \;\sqcup\; \bigsqcup_{j=1}^{m}(\exists P_{cr}.(D_j \sqcap \neg D_j'))$$

□

$\mathcal{SHIQ}$ is strictly less expressive than $\mathcal{SHOIQ}$

Exercise

Show that boolean TBoxes cannot be represented in $\mathcal{SHIQ}$.
[Hint: use the fact that $\mathcal{SHIQ}$ is invariant under disjoint union of models.]

Theorem

$\mathcal{SHIQ}$ is strictly less expressive than $\mathcal{SHOIQ}$.

Proof.

Boolean $\mathcal{SHIQ}$ TBoxes can be encoded in standard $\mathcal{SHOIQ}$ TBoxes.
But these cannot be represented in $\mathcal{SHIQ}$. □

Outline of Part 6

Nominals and tree model property

The tree model property is a key property that makes modal logics, and hence description logics, robustly decidable [Vardi 1997].

The tree model property fails for DLs with nominals.

The concept $\{a\} \sqcap \exists R.\{a\}$ is satisfied only by a model containing a cycle on a.

The **interaction between nominals, number restrictions, and inverse roles**

- leads to the almost complete loss of the tree model property;
- causes the complexity of the ontology satisfiability problem to jump from ExpTime to NExpTime [Tobies 2000];
- makes it difficult to extend the $\mathcal{SHIQ}$ tableaux algorithm to $\mathcal{SHOIQ}$.

Example

Consider the TBox $\mathcal{T}$ that contains:

$$\top \sqsubseteq \exists P^-.\{o\} \qquad\qquad \{o\} \sqsubseteq (\leq 20\, P.\, A)$$

Completion Graph

Def.: Completion graph

Let $\mathcal{R}$ be an RBox (i.e., a role hierarchy) and C_0 a $\mathcal{SHOIQ}$-concept in NNF. A **completion graph for C_0** with respect to $\mathcal{R}$ is a directed graph

$$\mathbf{G} = \langle V, E, \mathcal{L}, \neq \rangle$$

where:

$$\begin{array}{rcl} \mathcal{L}(v) & \subseteq & Cl(C_0) \cup N_I \cup \\ & & \{(\leq m\, R.\, C) \mid (\leq n\, R.\, C) \in Cl(C_0) \text{ and } m < n\} \\ \mathcal{L}(v,w) & \subseteq & \{R \mid R \text{ is a role of } C_0\} \\ \neq & \subseteq & V \times V \end{array}$$

- $Cl(C_0)$ is the **syntactic closure** of C_0, and is constituted by C_0 all its subconcepts.
- N_I is the set of all individuals.

Clash

Def.: Clash

A completion graph G contains a **clash** if:

1. $\{A, \neg A\} \subseteq \mathcal{L}(x)$ for some A and x; ($\mathcal{ALC}$)
2. $(\leq n\, S.C) \in \mathcal{L}(x)$ and there are $n+1$ S-neighbours $y_0, \ldots, y_n$ of x with $C \in \mathcal{L}(y_i)$, and $y_i \neq y_j$ for $0 \leq i < j \leq n$ ($\mathcal{ALCQ}$)
3. $o \in \mathcal{L}(x) \cap \mathcal{L}(y)$, and $x \neq y$ for some nodes x, y and nominal o. ($\mathcal{SHOIQ}$)

Blockable nodes

Def.: Nominal node

A **nominal node** is a node x, such that $\mathcal{L}(x)$ contains a nominal o.

Def.: Blockable node

A **blockable node** is any node that is not a nominal node.

Def.: Safe neighbours

An R-neighbour y of a node x is **safe** if

- x is blockable, or
- x is a nominal node and y is not blocked.

Completion rules for $\mathcal{SHOIQ}$

$\sqcap$-rule	if 1.	$C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and
	2.	$\{C_1, C_2\} \not\in \mathcal{L}(x)$
	then	$\mathcal{L}(x) := \mathcal{L}(x) \cup \{C_1, C_2\}$
$\sqcup$-rule	if 1.	$C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and
	2.	$\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
	then	$\mathcal{L}(x) := \mathcal{L}(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$
$\exists$-rule	if 1.	$\exists S.C \in \mathcal{L}(x)$, x is not blocked, and
	2.	x has no safe S-neighbour y with $C \in \mathcal{L}(y)$,
	then	create a new node y with $\mathcal{L}(x, y) = \{S\}$ and $\mathcal{L}(y) = \{C\}$
$\forall$-rule	if 1.	$\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked, and
	2.	there is an S-neighbour y of x with $C \not\in \mathcal{L}(y)$
	then	$\mathcal{L}(y) := \mathcal{L}(y) \cup \{C\}$
$\forall_+$-rule	if 1.	$\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked, and
	2.	there is some R with (**trans** R) and $R \sqsubseteq^* S$, and
	3.	there is an R-neighbour y of x with $\forall R.C \not\in \mathcal{L}(y)$
	then	$\mathcal{L}(y) := \mathcal{L}(y) \cup \{\forall R.C\}$

Completion rules for $\mathcal{SHOIQ}$ (cont'd)

?-rule	if 1.	$(\leq n\, S.\, C) \in \mathcal{L}(x)$, x is not indirectly blocked, and
	2.	there is an S-neighbour y of x with $\{C, \dot{\neg}C\} \cap \mathcal{L}(y) = \emptyset$
	then	$\mathcal{L}(y) := \mathcal{L}(y) \cup \{E\}$ for some $E \in \{C, \dot{\neg}C\}$
$\geq$-rule	if 1.	$(\geq n\, S.\, C) \in \mathcal{L}(x)$, x is not blocked, and
	2.	there are not n safe S-neighbors $y_1, \ldots, y_n$ of x with $C \in \mathcal{L}(y_i)$ and $y_i \neq y_j$ for $1 \leq i < j \leq n$
	then	create n new nodes $y_1, \ldots, y_n$ with $\mathcal{L}(x, y_i) = \{S\}$, $\mathcal{L}(y_i) = \{C\}$, and $y_i \neq y_j$ for $1 \leq i < j \leq n$
$\leq$-rule	if 1.	$(\leq n\, S.\, C) \in \mathcal{L}(z)$, z is not indirectly blocked, and
	2.	$\#S^G(z, C) > n$ and there are two S-neighbours x, y of z with $C \in \mathcal{L}(x) \cap \mathcal{L}(y)$, and not $x \neq y$
	then 1.	if x is a nominal node, then $Merge(y, x)$
	2.	else if y is a nominal node or an ancestor of x, then $Merge(x, y)$
	3.	else $Merge(y, x)$

Blocking strategy in $\mathcal{SHOIQ}$

The blocking strategy is the same as in $\mathcal{SHIQ}$, namely **double-blocking**, but restricted to the non-nominal nodes (i.e., blockable nodes).

Def.: Blocking in $\mathcal{SHOIQ}$

A node x is **directly blocked** if it has ancestors x', y and y' such that

1. x is a successor of x' and y is a successor of y',
2. y, x and all nodes on the path from y to x are blockable,
3. $\mathcal{L}(x) = \mathcal{L}(y)$ and $\mathcal{L}(x') = \mathcal{L}(y')$, and
4. $\mathcal{L}(x', x) = \mathcal{L}(y', y)$.

A node is **indirectly blocked** if it is blockable and its predecessor is directly blocked.

A node is **blocked** if it is directly or indirectly blocked.

Merging nodes

$Merge(y, x)$ is obtained by

- adding $\mathcal{L}(y)$ to $\mathcal{L}(x)$;
- redirecting to x all the edges leading to y;
- redirecting all the edges leading from y to nominal nodes so that they lead from x to the same nominal nodes;
- removing y (and blockable sub-trees below y).

Tableaux rules for $\mathcal{SHOIQ}$ (rules for nominals)

o-rule	if	for some nominal o **there are 2 nodes** x, y **with** $o \in \mathcal{L}(x) \cap \mathcal{L}(y)$ **and not** $x \neq y$
	then	$Merge(x, y)$
$o?$-rule	if 1.	$(\leq n\, S.C) \in \mathcal{L}(x)$, x is a nominal node, and there is a blockable S-neighbour y of x such that $\{C\} \in \mathcal{L}(y)$ and x is a successor of y and
	2.	there is no m with $1 \leq m \leq n$, $(\leq m\, S.C) \in \mathcal{L}(x)$ and there are m nominal S-neighbours $z_1, \ldots z_m$ of x with $C \in \mathcal{L}(z_i)$ and $z_i \neq z_j$ for all $1 \leq i < j \leq m$
	then 1.	guess $m \leq n$ and set $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(\leq m\, S.C)\}$
	2.	create m new nodes $y_1, \ldots, y_m$ with $\mathcal{L}(x, y_i) := \{S\}$, $\mathcal{L}(y_i) = \{C, o_i\}$ for $o_i \in N_I$ new in G, and $y_i \neq y_j$ for all $1 \leq i < j \leq m$

Outline of Part 6

unibz

More expressive role constructs

$\mathcal{SROIQ}$ [Horrocks, Kutz, and Sattler 2006], at the basis of the OWL 2 language, and its extension $\mathcal{SROIQB}$ [Rudolph, Krötzsch, and Hitzler 2008] allow for more expressive RBoxes.

Note: We need to distinguish between:

- arbitrary roles R: are those implied by role composition;
- simple roles S: may be used in number restrictions and with booleans.

Role composition: $R_1 \circ R_2$ in the left-hand-side of role inclusions.
Example: hasParent $\circ$ hasBrother $\sqsubseteq$ hasUncle

Role properties: Direct statements about (simple) roles, such as (**trans** R), (**sym** R), (**asym** S), (**refl** R), (**irrefl** S), (**funct** S), (**invFunct** S), and (**disj** S_1 S_2)
Example: (**trans** hasAncestor), (**sym** spouse), (**asym** hasChild), (**refl** hasRelative), (**irrefl** parentOf), (**funct** hasHusband), (**invFunct** hasHusband), (**disj** hasSibling hasCousin)

Boolean combination of simple roles (in $\mathcal{SROIQB}$): $\neg S$, $S_1 \sqcup S_2$, $S_1 \sqcap S_2$
Example: hasParent $\equiv$ hasMother $\sqcap$ hasFather, $\neg$likes

The description logic $\mathcal{SROIQB}$

Construct	Syntax	Semantics
inverse role	R^-	$\{(o,o') \mid (o',o) \in R^\mathcal{I}\}$
universal role	U	$\Delta^\mathcal{I} \times \Delta^\mathcal{I}$
role negation	$\neg S$	$(\Delta^\mathcal{I} \times \Delta^\mathcal{I}) \setminus S^\mathcal{I}$
role conjunction	$S_1 \sqcap S_2$	$S_1^\mathcal{I} \cap S_2^\mathcal{I}$
role disjunction	$S_1 \sqcup S_2$	$S_1^\mathcal{I} \cup S_2^\mathcal{I}$
top	$\top$	$\Delta^\mathcal{I}$
bottom	$\bot$	$\emptyset$
negation	$\neg C$	$\Delta^\mathcal{I} \setminus C^\mathcal{I}$
conjunction	$C \sqcap D$	$C^\mathcal{I} \cap D^\mathcal{I}$
disjunction	$C \sqcup D$	$C^\mathcal{I} \cup D^\mathcal{I}$
nominal	$\{a\}$	$\{a^\mathcal{I}\}$
value restriction	$\forall R.C$	$\{o \mid \forall o'.\, (o,o') \in R^\mathcal{I} \rightarrow o' \in C^\mathcal{I}\}$
existential restr.	$\exists R.C$	$\{o \mid \exists o'.\, (o,o') \in R^\mathcal{I} \land o' \in C^\mathcal{I}\}$
Self concept	$\exists S.\textbf{Self}$	$\{o \mid (o,o) \in S^\mathcal{I}\}$
qualified number	$(\geq n\, S.\, C)$	$\{o \mid \#\{o' \mid (o,o') \in S^\mathcal{I} \land o' \in C^\mathcal{I}\} \geq n\}$
restrictions	$(\leq n\, S.\, C)$	$\{o \mid \#\{o' \mid (o,o') \in S^\mathcal{I} \land o' \in C^\mathcal{I}\} \leq n\}$

unibz

Dealing with complex role inclusion axioms (RIAs)

Unrestricted use of role composition in RIAs causes undecidability.
To regain decidability, we need to impose some restrictions.

Role inclusion axioms as a grammar

A set $\mathcal{R}$ of RIAs can be seen as a context-free grammar:

$$R_1 \circ \cdots \circ R_n \sqsubseteq R \qquad \Longrightarrow \qquad R \ \longrightarrow_{\mathcal{R}} \ R_1 \cdots R_n$$

We can consider the language that the grammar for $\mathcal{R}$ associates to a role R:

$$L_{\mathcal{R}}(R) = \{R_1 \cdots R_n \ \mid \ R \ \overset{*}{\longrightarrow}_{\mathcal{R}} \ R_1 \cdots R_n\}$$

Regular RIAs

The tableaux algorithm for $\mathcal{SROIQ}$ is based on using finite-state automata for $L_{\mathcal{R}}(R)$. Hence, decidability can be obtained by restricting to RBoxes corresponding to **regular** context free grammars.

Regular RIAs – Examples

Example (Regular RIAs)

$$R \circ S \sqsubseteq R$$
$$S \circ R \sqsubseteq R$$

Generates the language $S^* R S^*$, which is regular.

Example (Non regular RIAs)

$$S \circ R \circ S \sqsubseteq R$$

Generates the language $S^n R S^n$, which is **not regular**.

Ensuring decidability in $\mathcal{SROIQ}$

Checking if a context-free grammar is regular is undecidable, hence one cannot check regularity of a set of RIAs.

$\mathcal{SROIQ}$ provides a **sufficient condition for the regularity** of RIAs.

Def.: Regular RIAs

A role inclusion assertion is **$\prec$-regular** if it has one of the forms:

$$\begin{array}{rcl} R \circ R & \sqsubseteq & R \\ R^- & \sqsubseteq & R \end{array} \qquad \begin{array}{rcl} S_1 \circ \cdots \circ S_n & \sqsubseteq & R \\ R \circ S_1 \circ \cdots \circ S_n & \sqsubseteq & R \\ S_1 \circ \cdots \circ S_n \circ R & \sqsubseteq & R \end{array}$$

where $\prec$ is a **strict partial order** on direct and inverse roles such that

- $S \prec R$ iff $S^- \prec R$, and
- $S_i \prec R$, for $1 \leq i \leq n$.

A set $\mathcal{R}$ of RIAs is **regular** if there is a $\prec$ s.t. all RIAs in $\mathcal{R}$ are $\prec$-regular.

Regular RIAs – Examples

Exercise

Check whether the following set $\mathcal{R}_1$ of RIAs satisfies regularity of $\mathcal{SROIQ}$:

$$\begin{array}{rcl} \text{isProperPartOf} & \sqsubseteq & \text{isPartOf} \\ \text{isPartOf} \circ \text{isPartOf} & \sqsubseteq & \text{isPartOf} \\ \text{isPartOf} \circ \text{isProperPartOf} & \sqsubseteq & \text{isPartOf} \\ \text{isProperPartOf} \circ \text{isPartOf} & \sqsubseteq & \text{isPartOf} \end{array}$$

Then define $L_{\mathcal{R}_1}(\text{isPartOf})$.

Exercise

Check whether the following set $\mathcal{R}_2$ of RIAs satisfies regularity of $\mathcal{SROIQ}$:

$$\begin{array}{rcl} R \circ R & \sqsubseteq & R \\ S & \sqsubseteq & R \end{array} \qquad\qquad \begin{array}{rcl} R \circ S & \sqsubseteq & S \\ S \circ R & \sqsubseteq & S \end{array}$$

Then define $L_{\mathcal{R}_2}(R)$ and $L_{\mathcal{R}_2}(S)$ and check if they are regular languages.

Reasoning in $\mathcal{SROIQ}$ – Overview

To reason in $\mathcal{SROIQ}$, one can proceed as follows:

1. Eliminate role assertions of the form (**funct** S), (**invFunct** S), (**sym** R), (**trans** R), (**irrefl** R).
2. Eliminate the universal role.
3. Reduce reasoning w.r.t. an ontology consisting of TBox+ABox+RBox to reasoning w.r.t. an RBox only.
   The resulting RBox is of a simplified form and is called a **reduced RBox**.
4. Provide tableaux rules that are able to check concept satisfiability w.r.t. a reduced RBox.

We look at these steps a bit more in detail.

Reasoning in $\mathcal{SROIQ}$ – 1. Eliminating role assertions

We have the following equivalences that allow us to eliminate some of the role assertions:

- (**funct** S) is equivalent to the concept inclusion $\top \sqsubseteq (\leq 1\, S)$.
- (**invFunct** S) is equivalent to the concept inclusion $\top \sqsubseteq (\leq 1\, S^-)$.
- (**sym** R) is equivalent to the role inclusion $R \sqsubseteq R^-$.
- (**trans** R) is equivalent to the role inclusion $R \circ R \sqsubseteq R$.
- (**irrefl** R) is equivalent to the concept inclusion $\top \sqsubseteq \neg\exists R.$**Self**.

Notice also that (**refl** R) is equivalent to the concept inclusion $\top \sqsubseteq \exists R.$**Self**. However, this concept inclusion can only be used when R is a simple role, and hence does not allow us to eliminate (**refl** R) in general.

Reasoning in $\mathcal{SROIQ}$ – 2. Eliminating universal role

To **eliminate the universal role**:

1. Consider U as any other role (without special interpretation).
2. Define the following concept:

$$C_{\mathcal{T}} \equiv \forall U.(\bigsqcap_{C \sqsubseteq D \in \mathcal{T}} \neg C \sqcup D) \sqcap \bigsqcap_{o \in N} \exists U.\{o\}.$$

3. Extend the RBox with the following assertions: $R \sqsubseteq U$, (**trans** U), (**sym** U), and (**refl** U).

This encoding is correct, since one can show that a satisfiable $\mathcal{SROIQ}$ ontology has a **nominal connected model**, i.e., a model that is a union of connected components, where each such component contains a nominal, and where any two elements of a connected component are connected by a role path over the roles occurring in the ontology.

unibz

Reasoning in $\mathcal{SROIQ}$ – 3. Internalizing ABox and TBox

We have already seen that using nominals we can:

1. **encode an ABox** by means of TBox assertions, and
2. **internalize a (boolean) TBox** and reduce concept satisfiability and subsumption w.r.t. a TBox to satisfiability of a single (nominal) concept.

Hence, it suffices to consider only (un)satisfiability of $\mathcal{SROIQ}$ concepts w.r.t. RBoxes that:

- do not contain the universal role,
- contain a regular role hierarchy, and
- contain only role assertions of the form (**refl** R), (**asym** R), and (**disj** S_1 S_2).

We call such RBoxes **reduced**.

Reasoning in $\mathcal{SROIQ}$ – 4. Additional tableaux rules

- The tableaux algorithm uses for each (direct or inverse) role S a non-deterministic finite state automaton $\mathcal{B}_S$ defined by the reduced RIAs $\mathcal{R}$.
- $L(\mathcal{B})$ denotes the regular language accepted by an NFA $\mathcal{B}$.
- For a state p of $\mathcal{B}$, $\mathcal{B}(p)$ denotes the NFA identical to $\mathcal{B}$ but with initial state p.

Self-Ref-rule	if then	$\exists S.$**Self** $\in \mathcal{L}(x)$ or (**refl** $S) \in \mathcal{R}$, x is not blocked, and $S \notin \mathcal{L}(x,x)$ add an edge (x,x) if it does not yet exist, and set $\mathcal{L}(x,x) := \mathcal{L}(x,x) \cup \{S\}$
$\forall_1$-rule	if then	$\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked, and $\forall \mathcal{B}_S.C \notin \mathcal{L}(x)$ $\mathcal{L}(x) := \mathcal{L}(x) \cup \{\forall \mathcal{B}_S.C\}$
$\forall_2$-rule	if 1. 2. then	$\forall \mathcal{B}(p).C \in \mathcal{L}(x)$, x is not indirectly blocked, $p \xrightarrow{S} q$ in $\mathcal{B}(p)$, and there is an S-neighbour y of x with $\forall \mathcal{B}(q).C \notin \mathcal{L}(y)$ $\mathcal{L}(y) := \mathcal{L}(y) \cup \{\forall \mathcal{B}(q).C\}$
$\forall_3$-rule	if then	$\forall \mathcal{B}.C \in \mathcal{L}(x)$, x is not indirectly blocked, $\varepsilon \in L(\mathcal{B})$, and $C \notin \mathcal{L}(x)$ $\mathcal{L}(x) := \mathcal{L}(x) \cup \{C\}$

Decidability of reasoning in $\mathcal{SROIQ}$

Theorem (Termination, Soundness, and Completeness of $\mathcal{SROIQ}$ tableaux)

Let C_0 be a $\mathcal{SROIQ}$ concept in NNF and $\mathcal{R}$ a reduced RBox.

1. The tableaux algorithm terminates when started with C_0 and $\mathcal{R}$.
2. The tableaux rules can be applied to C_0 and $\mathcal{R}$ so as to yield a complete and clash-free completion graph iff there is a tableau for C_0 w.r.t. $\mathcal{R}$.

From the previous encodings, we obtain decidability of reasoning in $\mathcal{SROIQ}$.

Theorem (Decidability of $\mathcal{SROIQ}$)

The tableaux algorithm decides satisfiability and subsumption of $\mathcal{SROIQ}$ concepts with respect to ABoxes, RBoxes, and TBoxes.

Note:

- The NFA constructed from a set $\mathcal{R}$ of regular RIAs may be exponential in the size of $\mathcal{R}$. This blowup is essentially unavoidable [Kazakov 2008].
- The tableaux algorithm is not computationally optimal.

References I

[1] Franz Baader, Diego Calvanese, et al., eds. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[2] Johan van Benthem. "Modal Correspondence Theory". PhD thesis. Mathematish Instituut and Instituut voor Grondslagenonderzoek, University of Amsterdam, 1976.

[3] Johan van Benthem. *Modal Logic and Classical Logic*. Bibliopolis, Napoli, 1983.

[4] Franz Baader and Ulrike Sattler. "An Overview of Tableau Algorithms for Description Logics". In: *Studia Logica* 69.1 (2001), pp. 5–40.

[5] Manfred Schmidt-Schauss and Gert Smolka. "Attributive Concept Descriptions with Complements". In: *Artificial Intelligence* 48.1 (1991), pp. 1–26.

References II

[6] Francesco M. Donini, Bernhard Hollunder, et al. "The Complexity of Existential Quantification in Concept Languages". In: *Artificial Intelligence* 2–3 (1992), pp. 309–327.

[7] Francesco M. Donini, Maurizio Lenzerini, et al. "The Complexity of Concept Languages". In: *Information and Computation* 134 (1997), pp. 1–58.

[8] Francesco M. Donini. "Complexity of Reasoning". In: *The Description Logic Handbook: Theory, Implementation and Applications*. Ed. by Franz Baader et al. Cambridge University Press, 2003. Chap. 3, pp. 96–136.

[9] Vaugham R. Pratt. "Models of Program Logic". In: *Proc. of the 20th Annual Symp. on the Foundations of Computer Science (FOCS 1979)*. 1979, pp. 115–122.

[10] Klaus Schild. "A Correspondence Theory for Terminological Logics: Preliminary Report". In: *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI 1991)*. 1991, pp. 466–471.

References III

[11] Diego Calvanese and Giuseppe De Giacomo. “Expressive Description Logics”. In: *The Description Logic Handbook: Theory, Implementation and Applications*. Ed. by Franz Baader et al. Cambridge University Press, 2003. Chap. 5, pp. 178–218.

[12] Ralf Möller and Volker Haarslev. “Description Logic Systems”. In: *The Description Logic Handbook: Theory, Implementation and Applications*. Ed. by Franz Baader et al. Cambridge University Press, 2003. Chap. 8, pp. 282–305.

[13] Moshe Y. Vardi. “Why Is Modal Logic so Robustly Decidable”. In: *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. Vol. 31. American Mathematical Society, 1997, pp. 149–184.

[14] Stephan Tobies. “The Complexity of Reasoning with Cardinality Restrictions and Nominals in Expressive Description Logics”. In: *J. of Artificial Intelligence Research* 12 (2000), pp. 199–217.

References IV

[15] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. “The Even More Irresistible $\mathcal{SROIQ}$”. In: *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*. 2006, pp. 57–67.

[16] Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. “Cheap Boolean Role Constructors for Description Logics”. In: *Proc. of the 11th Eur. Conference on Logics in Artificial Intelligence (JELIA 2008)*. Vol. 5293. Lecture Notes in Computer Science. Springer, 2008, pp. 362–374.

[17] Yevgeny Kazakov. “$\mathcal{RIQ}$ and $\mathcal{SROIQ}$ Are Harder than $\mathcal{SHOIQ}$”. In: *Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2008)*. 2008, pp. 274–284.