Knowledge Representation and Ontologies

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

Diego Calvanese

Faculty of Computer Science
Master of Science in Computer Science

A.Y. 2011/2012

Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

Part 6

Reasoning in the $\mathcal{ALC}$ family

Outline of Part 6

Outline of Part 6

Outline of Part 6

1. Properties of $\mathcal{ALC}$
   - $\mathcal{ALC}$ and first-order logic
   - Bisimulations
   - Properties of $\mathcal{ALC}$
2. Reasoning over $\mathcal{ALC}$ concept expressions
3. Reasoning over $\mathcal{ALC}$ knowledge bases
4. Extensions of $\mathcal{ALC}$
5. Reasoning in extensions of $\mathcal{ALC}$
6. $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$
7. References

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.it

$\mathcal{ALC}$ 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:

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

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

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 (FOL$_{\text{bin}}$).

To translate an $\mathcal{ALC}$ TBox to FOL$_{\text{bin}}$ 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:

$$t_x(A) = A(x) \qquad t_x(C \sqcap D) = t_x(C) \wedge t_x(D)$$
$$t_x(\neg C) = \neg t_x(C) \qquad t_x(C \sqcup D) = t_x(C) \vee t_x(D)$$
$$t_x(\exists P.C) = \exists y.\, P(x, y) \wedge t_y(C)$$
$$t_x(\forall P.C) = \forall y.\, P(x, y) \rightarrow t_y(C)$$

(with y a new variable)

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.:

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

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.)

unibz.it

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

Question

Is it possible to define a transformation $\tau(\cdot)$ from FOL$_{\text{bin}}$ 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.

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:

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

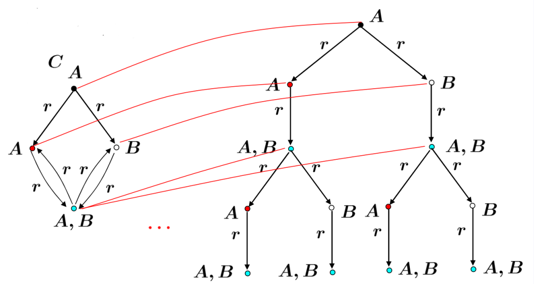
Tree model property of DLs

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_{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_{bin} is that:

Expressive power of $\mathcal{ALC}$

$\mathcal{ALC}$ is **strictly less expressive** than FOL_{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 $(\mathcal{I}, o_1) \sim (\mathcal{J}, o_2)$

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

Theorem ([van Benthem, 1976; van Benthem, 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

1. Properties of $\mathcal{ALC}$
2. Reasoning over $\mathcal{ALC}$ concept expressions
   - Tableaux for concept satisfiability
   - Complexity of concept satisfiability
3. Reasoning over $\mathcal{ALC}$ knowledge bases
4. Extensions of $\mathcal{ALC}$
5. Reasoning in extensions of $\mathcal{ALC}$
6. $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$
7. References

Negation Normal Form

Definition

A concept C is in **negation normal form (NNF)** if the '$\neg$' operator is applied only to atomic concepts

Lemma

Every concept C can be transformed in linear time into an equivalent concept in NNF.

Proof.

A concept C can be transformed in NNF by the following rewriting rules that push inside the $\neg$ operator:

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

□

Tableaux rules for checking concept satisfiability

Let C_0 be an $\mathcal{ALC}$ concept in NNF.

To test satisfiability of C_0, a tableaux algorithm:

1. starts with $\mathcal{A}_0 := \{C_0(x_0)\}$, and
2. constructs new ABoxes, by applying the following **tableaux rules**:

Rule	Condition	$\longrightarrow$	Effect
$\rightarrow_\sqcap$	$(C_1 \sqcap C_2)(x) \in \mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{C_1(x), C_2(x)\}$
$\rightarrow_\sqcup$	$(C_1 \sqcup C_2)(x) \in \mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{C_1(x)\}$ **or** $\mathcal{A} := \mathcal{A} \cup \{C_2(x)\}$
$\rightarrow_\exists$	$(\exists P.C)(x) \in \mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{P(x,y), C(y)\}$, where y is fresh
$\rightarrow_\forall$	$(\forall P.C)(x), P(x,y) \in \mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{C(y)\}$

Note:

- A rule is applicable to an ABox $\mathcal{A}$ only if it has an effect on $\mathcal{A}$, i.e., if it adds some new assertion; otherwise it is not applicable to $\mathcal{A}$.
- Since the $\rightarrow_\sqcup$ rule is non-deterministic, starting from $\mathcal{A}_0$, we obtain after each rule application a set $\mathcal{S}$ of ABoxes.

Complete and clash-free ABoxes

Definition

An ABox $\mathcal{A}$

- is **complete** if none of the tableaux rules applies to it.
- has a **clash** if $\{C(x), \neg C(x)\} \subseteq \mathcal{A}$, and is **clash-free** otherwise.

A clash represents an obvious contradiction. Hence, it is immediate so see that an ABox containing a clash is unsatisfiable.

Termination, soundness, and completeness

For a set finite $\mathcal{S}$ of ABoxes, we say that $\mathcal{S}$ is **consistent** if it contains at least one satisfiable ABox.

Lemma

1. **Termination:** There cannot be an infinite sequence of rule applications

$$\mathcal{S}_0 = \{\{C_0(x_0)\}\} \longrightarrow \mathcal{S}_1 \longrightarrow \mathcal{S}_2 \longrightarrow \cdots$$

2. **Soundness:** If by applying a tableaux rule to the set $\mathcal{S}$ of ABoxes we obtain the set $\mathcal{S}'$, then **$\mathcal{S}$ is consistent iff $\mathcal{S}'$ is consistent**.

3. **Completeness:** Every **complete and clash-free** ABox $\mathcal{A}$ is **satisfiable**.

Canonical interpretation and decidability of satisfiability

To show that every complete and clash-free ABox $\mathcal{A}$ is satisfiable, we describe how to generate from such an $\mathcal{A}$ an interpretation $\mathcal{I}_\mathcal{A}$ that is a model of $\mathcal{A}$.

This interpretation is called ...

Def.: Canonical interpretation $\mathcal{I}_\mathcal{A}$ of a complete and clash-free ABox $\mathcal{A}$

- $\Delta^{\mathcal{I}_\mathcal{A}} = \{x \mid C(x), P(x,y),$ or $P(y,x) \in \mathcal{A}\}$.
- $A^{\mathcal{I}_\mathcal{A}} = \{x \mid A(x) \in \mathcal{A}\}$, for every atomic concept A.
- $P^{\mathcal{I}_\mathcal{A}} = \{(x,y) \mid P(x,y) \in \mathcal{A}\}$, for every atomic role P.

Theorem

Satisfiability of $\mathcal{ALC}$ concepts is decidable.

Proof.

Is based on showing that the canonical interpretation of an ABox $\mathcal{A}$ obtained starting from a concept C is indeed a model of C. □

Outline of Part 6

Complexity of reasoning in $\mathcal{ALC}$

Exercise

Consider the concept C_n inductively defined as follows;

$$\begin{array}{rcl} C_1 & = & \exists P.A \sqcap \exists P.\neg A \\ C_{n+1} & = & \exists P.A \sqcap \exists P.\neg A \sqcap \forall P.C_n \end{array}$$

Check the form of the canonical interpretation of the ABox obtained starting from $\{C_n(x_0)\}$.

Solution

Given the input concept C_n, the satisfiability algorithm generates a complete and open ABox whose canonical interpretation is a binary tree of depth n, and thus consists of $2^{n+1} - 1$ individuals.

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

Complexity of reasoning in $\mathcal{ALC}$

Theorem [Schmidt-Schauss and Smolka, 1991]

Satisfiability of $\mathcal{ALC}$ concepts is $\textsc{PSpace}$-complete.

Proof sketch of membership in $\textsc{PSpace}$.

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

- Since $\textsc{PSpace} = \textsc{NPSpace}$, we consider a non-deterministic algorithm that for each application of the $\rightarrow_{\sqcup}$-rule, chooses the "correct" ABox.
- 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 as follows:
  1. Apply the $\rightarrow_{\sqcap}$ and $\rightarrow_{\sqcup}$ rules exhaustively, and check for clashes.
  2. Choose a node x and apply the $\rightarrow_{\exists}$-rule to generate all necessary direct successors of x.
  3. Apply the $\rightarrow_{\forall}$ rule to propagate concepts to the newly generated successors.
  4. Successively handle the successors in the same way. $\square$

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.it

Satisfiability of $\mathcal{ALC}$ ABoxes

To test whether a given ABox $\mathcal{A}$ is satisfiable:

1. Convert $\mathcal{A}$ in NNF, obtaining $\mathcal{A}_0$.
2. Apply the tableaux algorithm starting simply from $\mathcal{A}_0$.

Theorem

Satisfiability of $\mathcal{ALC}$ ABoxes is PSPACE-complete.

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

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

1. $\forall R.(A \sqcap B) \sqsubseteq \forall R.A \sqcap \forall R.B$
2. $\forall R.A \sqcap \forall R.B \sqsubseteq \forall R.(A \sqcap B)$
3. $\forall R.A \sqcup \forall R.B \sqsubseteq \forall R.(A \sqcup B)$
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. $\exists R.(A \sqcap B) \sqsubseteq \exists R.A \sqcap \exists R.B$
6. $\exists R.(A \sqcup B) \sqsubseteq \exists R.A \sqcup \exists R.B$
7. $\exists R.A \sqcup \exists R.B \sqsubseteq \exists R.(A \sqcup B)$
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

Outline of Part 6

TBox 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., $\mathcal{T} \models C_1 \equiv C_2$.

We can reduce all reasoning tasks to concept satisfiability wrt a TBox.
[Exercise]

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] $\square$

Complexity of unfolding w.r.t. an acyclic TBox

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. an acyclic TBox

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

Rule	Condition	$\longrightarrow$	Effect
$\rightarrow_\sqcap$	$(C_1 \sqcap C_2)(x) \in \mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{C_1(x), C_2(x)\}$
$\rightarrow_\sqcup$	$(C_1 \sqcup C_2)(x) \in \mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{C_1(x)\}$ or $\mathcal{A} := \mathcal{A} \cup \{C_2(x)\}$
$\rightarrow_\exists$	$(\exists P.C)(x) \in \mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{P(x, y), C(y)\}$, where y is fresh
$\rightarrow_\forall$	$(\forall P.C)(x), P(x, y) \in \mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{C(y)\}$
$\rightarrow_\mathcal{T}$	$A(x) \in \mathcal{A}$ and $A \equiv C \in \mathcal{T}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{\textsc{nnf}(C)(x)\}$

Theorem

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

Outline of Part 6

Tableaux rule for TBox axioms

We rely 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 $\top \sqsubseteq \bigsqcup_i (\neg C_i \sqcup D_i)$.
- If $\top \sqsubseteq C$ is an axiom of $\mathcal{T}$, then for every ABox generated by the tableaux and for every occurrence of some x in $\mathcal{A}$, we have to add also the fact $C(x)$.
- We can obtain this effect by adding a suitable rule to the tableaux rules:

Rule	**Condition**	$\longrightarrow$	**Effect**
$\rightarrow_\sqcap$	$(C_1 \sqcap C_2)(x) \in \mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{C_1(x), C_2(x)\}$
$\rightarrow_\sqcup$	$(C_1 \sqcup C_2)(x) \in \mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{C_1(x)\}$ or $\mathcal{A} := \mathcal{A} \cup \{C_2(x)\}$
$\rightarrow_\exists$	$(\exists P.C)(x) \in \mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{P(x,y), C(y)\}$, where y is fresh
$\rightarrow_\forall$	$(\forall P.C)(x), P(x,y) \in \mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{C(y)\}$
$\rightarrow_\mathcal{T}$	x occurs in $\mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{\bigsqcup_{C \sqsubseteq D \in \mathcal{T}} \mathrm{NNF}(\neg C \sqcup D)(x)\}$

unibz.it

Tableaux rule for TBox axioms – Example

Exercise

Check if C is satisfiable w.r.t. the TBox $\{C \sqsubseteq \exists R.C\}$.

Solution

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

Termination is no longer guaranteed!

Due to the application of the $\rightarrow_{\mathcal{T}}$-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.

Def.: Blocking

- y is an **ancestor** of x in an ABox $\mathcal{A}$, if $\mathcal{A}$ contains

$$R_0(y, x_1),\ R_1(x_1, x_2), \ldots,\ R_n(x_n, x).$$

- We label objects with sets of concepts: $\mathcal{L}(x) = \{C \mid C(x) \in \mathcal{A}\}$.
- x is **directly blocked** in $\mathcal{A}$ if it has an ancestor y with $\mathcal{L}(x) \subseteq \mathcal{L}(y)$.
- If y is the closest such node to x, we say that x is **blocked by** y.
- A node is **blocked** if it is directly blocked or one of its ancestors is blocked.

The application of all rules is restricted to nodes that are not blocked.
With this **blocking strategy**, one can show that the algorithm is guaranteed to terminate.

Blocking – Exercise

Exercise

Check if C is satisfiable w.r.t. the TBox $\{C \sqsubseteq \exists R.C\}$.

Solution

$$
\begin{array}{lll}
\{C(x_0)\} & \to_{\mathcal{T}} & \{C(x_0), (\neg C \sqcup \exists R.C)(x_0)\} \\
 & \to_{\sqcup} & \{C(x_0), (\neg C \sqcup \exists R.C)(x_0), (\exists R.C)(x_0)\} \\
 & \to_{\exists} & \{C(x_0), (\neg C \sqcup \exists R.C)(x_0), (\exists R.C)(x_0), R(x_0, x_1), C(x_1)\}
\end{array}
$$

x_1 is blocked by x_0 since $\mathcal{L}(x_1) = \{C\}$ and $\mathcal{L}(x_0) = \{C, \neg C \sqcup \exists R.C, \exists R.C\}$, hence $\mathcal{L}(x_1) \subseteq \mathcal{L}(x_0)$.

Complexity of concept satisfiability w.r.t. a TBox

Cyclic interpretations

The interpretation $\mathcal{I}_\mathcal{A}$ generated from an ABox $\mathcal{A}$ obtained by the tableaux algorithm with blocking strategy is defined as follows:

- $\Delta^{\mathcal{I}_\mathcal{A}} = \{x \mid C(x) \in \mathcal{A}$ and x is not blocked$\}$
- $A^{\mathcal{I}_\mathcal{A}} = \{x \mid x \in \Delta^{\mathcal{I}_\mathcal{A}}$ and $A(x) \in \mathcal{A}\}$
- $P^{\mathcal{I}_\mathcal{A}} = \{(x,y) \mid \{x,y\} \subseteq \Delta^{\mathcal{I}_\mathcal{A}}$ and $P(x,y) \in \mathcal{A}\} \cup$
  $\{(x,y) \mid x \in \Delta^{\mathcal{I}_\mathcal{A}},\ P(x,y') \in \mathcal{A},$ and y' is blocked by $y\}$

Complexity

The algorithm is **no longer in PSPACE** since it may generate role paths of exponential length before blocking occurs.

Theorem [Fischer and Ladner, 1979; Pratt, 1979; Schild, 1991]

Satisfiability of an $\mathcal{ALC}$ concept w.r.t. a general TBox is **EXPTIME-complete**.

Finite model property

Theorem

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

Proof.

The model constructed via tableaux is finite.
Completeness of the tableaux procedure implies that if a TBox is satisfiable, then the algorithm will find a model, which is indeed finite □

Outline of Part 6

1. Properties of $\mathcal{ALC}$
2. Reasoning over $\mathcal{ALC}$ concept expressions
3. Reasoning over $\mathcal{ALC}$ knowledge bases
4. Extensions of $\mathcal{ALC}$
   - Some important extensions of $\mathcal{ALC}$
   - Inverse roles
   - Number restrictions
   - Encoding number restrictions
   - Role constructs
   - TBox internalization
5. Reasoning in extensions of $\mathcal{ALC}$

Outline of Part 6

1. Properties of $\mathcal{ALC}$

2. Reasoning over $\mathcal{ALC}$ concept expressions

3. Reasoning over $\mathcal{ALC}$ knowledge bases

4. Extensions of $\mathcal{ALC}$
   - Some important extensions of $\mathcal{ALC}$
   - Inverse roles
   - Number restrictions
   - Encoding number restrictions
   - Role constructs
   - TBox internalization

5. Reasoning in extensions of $\mathcal{ALC}$

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 (**funct** R))
    Example: $\top \sqsubseteq (\leq 1\, \mathsf{hasFather})$
  - local functionality: $A \sqsubseteq (\leq 1\, R)$
    Example: $\mathsf{Person} \sqsubseteq (\leq 1\, \mathsf{hasFather})$
- 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)
- Subsumption 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

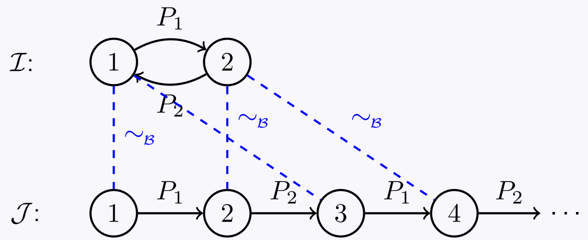
Inverse roles increase the expressive power

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 $\mathcal{ALCI}$.



$\models \exists P_1.\top \sqsubseteq \exists P_2^-.\top$

$\not\models \exists P_1.\top \sqsubseteq \exists P_2^-.\top$

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. $\mathsf{LonelyPerson} \equiv \mathsf{Person} \sqcap \neg\exists \mathsf{hasFriend}^-.\top \sqcap \neg\exists \mathsf{hasFriend}.\top$
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 $\mathsf{Person} \sqsubseteq \forall \mathsf{hasFather}.(\forall \mathsf{hasFather}^-.\mathsf{Person})$ 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 $o_1 \sim_{\mathcal{B}} o_2$:

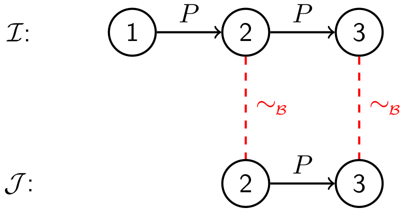
- for each o'_1 with $(o'_1, o_1) \in P^{\mathcal{I}}$, there is an $o'_2 \in \Delta^{\mathcal{J}}$ with $(o'_2, o_2) \in P^{\mathcal{J}}$ such that $o'_1 \sim_{\mathcal{B}} o'_2$.
- The same property in the opposite direction.

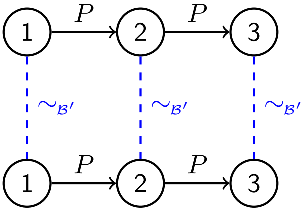
We call these properties the **inverse relation equivalence**.

$(\mathcal{I}, o_1) \sim_{\mathcal{ALCI}} (\mathcal{J}, o_2)$ 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 $(\mathcal{I}, 2) \sim_{\mathcal{ALCI}} (\mathcal{J}, 2)$.

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 $\exists P^-.C$:

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

□

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** $\pi = (o_1, o_2, \ldots, o_n)$ 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 $i \in \{1, \ldots, n-1\}$.
- $A^{\mathcal{J}} = \{\pi o_n \mid o_n \in A^{\mathcal{I}}\}$
- $P^{\mathcal{J}} = \{(\pi o_n \ , \ \pi o_n o_{n+1}) \mid (o_n, o_{n+1}) \in P^{\mathcal{I}}\} \cup$ $\{(\pi o_n o_{n+1} \ , \ \pi o_n) \mid (o_{n+1}, o_n) \in P^{\mathcal{I}}\}$

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 $o_i \sim_{\mathcal{B}} \pi o_i$. □

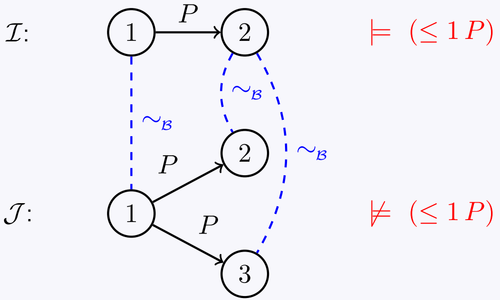
Outline of Part 6

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 $\mathcal{ALCQ}$.

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.

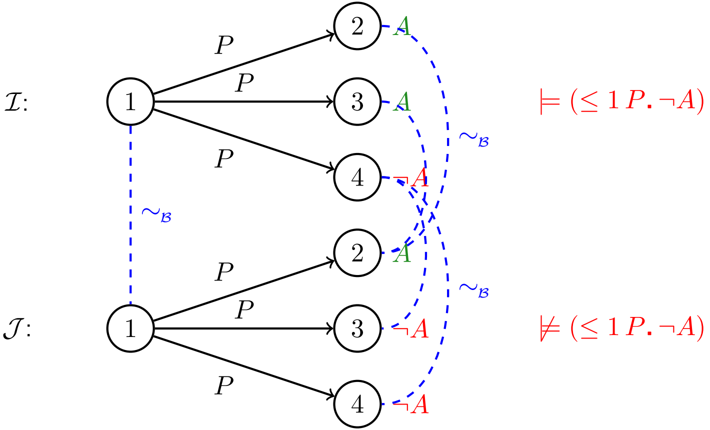
By induction on the structure of C.

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 $\mathcal{ALCN}$.



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:

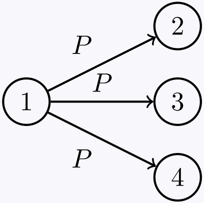
- a set $\mathcal{T}_r$ of $\mathcal{ALCFI}$-axioms, and
- a transformation π from $\mathcal{ALCN}$-concepts to $\mathcal{ALCFI}$-concepts

such that:

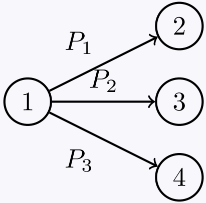
C is satisfiable w.r.t. $\mathcal{T}$ in $\mathcal{ALCN}$ iff
$\pi(C)$ is satisfiable w.r.t. $\pi(\mathcal{T}) \cup \mathcal{T}_r$ in $\mathcal{ALCFI}$

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:

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

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

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.: Semantics

$\mathcal{I} \models$ (**trans** P) if $P^{\mathcal{I}}$ is a transitive relation.

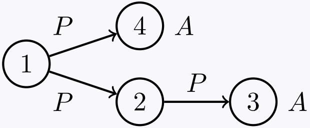
Note: if a role P is transitive, also P^- is transitive. Hence, we can restrict transitivity assertions to atomic roles only without losing expressive power.

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., $\bigwedge \Gamma \to \phi$.

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

Tableaux rules for $\mathcal{ALCI}$

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

Rule	Condition	$\longrightarrow$	Effect
$\rightarrow_\sqcap$	$(C_1 \sqcap C_2)(x) \in \mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{C_1(x), C_2(x)\}$
$\rightarrow_\sqcup$	$(C_1 \sqcup C_2)(x) \in \mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{C_1(x)\}$ or $\mathcal{A} := \mathcal{A} \cup \{C_2(x)\}$
$\rightarrow_\exists$	$(\exists P.C)(x) \in \mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{P(x,y), C(y)\}$, where y is fresh
	$(\exists P^-.C)(x) \in \mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{P(y,x), C(y)\}$, where y is fresh
$\rightarrow_\forall$	$(\forall P.C)(x), P(x,y) \in \mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{C(y)\}$
	$(\forall P^-.C)(x), P(y,x) \in \mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{C(y)\}$
$\rightarrow_\mathcal{T}$	x occurs in $\mathcal{A}$	$\longrightarrow$	$\mathcal{A} := \mathcal{A} \cup \{\bigsqcup_{C \sqsubseteq D \in \mathcal{T}} \mathrm{NNF}(\neg C \sqcup D)(x)\}$

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

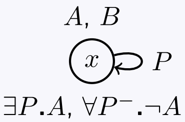
Tableaux for $\mathcal{ALCI}$ – Example

Example

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

Solution

$(A \sqcap \exists P.A \sqcap \forall P^-.\neg A)(x)$
$B(x)$
$A(x),\ (\exists P.A)(x),\ (\forall P^-.\neg A)(x)$
$P(x, y),\ A(y)$
y is blocked by x



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 $\mathcal{ALCI}$

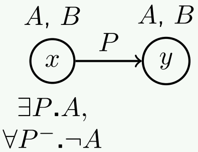
For $\mathcal{ALCI}$, subset-blocking, where the blocking condition is $\mathcal{L}(x) \subseteq \mathcal{L}(y)$, is no longer sufficient. We need to adopt a stronger blocking strategy.

Def.: Equality blocking

A node x is called directly blocked if it has an ancestor y with $L(x) = L(y)$.

For the previous example

$(A \sqcap \exists P.A \sqcap \forall P^{-}.\neg A)(x)$
$B(x)$
$A(x),\ (\exists P.A)(x),\ (\forall P^{-}.\neg A)(x)$
$P(x,y),\ A(y)$
$B(y)$
y is not blocked anymore by x



Decidability and complexity of $\mathcal{ALCI}$

Theorem

Let $\mathcal{T}$ be a general $\mathcal{ALCI}$-TBox and C an $\mathcal{ALCI}$-concept. Then:

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

Corollary

- Satisfiability of $\mathcal{ALCI}$-concepts w.r.t. general TBoxes is **decidable**.
- $\mathcal{ALCI}$ has the **finite model property**.

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

- **Termination:** As for $\mathcal{ALC}$.
- **Soundness:** if the algorithm generates a class-free tableaux, then C is satisfiable w.r.t. $\mathcal{T}$.
  - $\Delta^{\mathcal{I}} = \{x \mid C(x) \in \mathcal{A}$ and x is not blocked$\}$
  - $A^{\mathcal{I}} = \{x \mid x \in \Delta^{\mathcal{I}}$ and $A(x) \in \mathcal{A}\}$
  - $P^{\mathcal{I}} = \{(x,y) \mid \{x,y\} \subseteq \Delta^{\mathcal{I}_{\mathcal{A}}}$ and $P(x,y) \in \mathcal{A}\} \cup$
    $\{(x,y) \mid x \in \Delta^{\mathcal{I}},\ P(x,y') \in \mathcal{A},$ and y' is blocked by $y\} \cup$
    $\{(x,y) \mid y \in \Delta^{\mathcal{I}},\ P(x',y) \in \mathcal{A},$ and x' is blocked by $x\}$
- **Completeness:** given a model $\mathcal{I}$ of C, we can use it to steer the application of the non-deterministic rule for $\sqcup$.
  At the end we obtain a tableaux that generates a model $\mathcal{J}$ that is bisimilar to the initial model $\mathcal{I}$.

Outline of Part 6

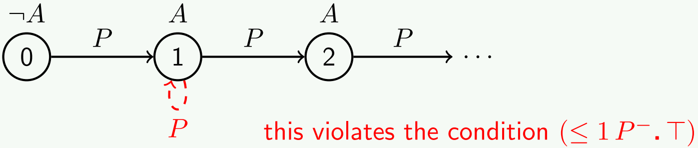
$\mathcal{ALCQI}$ and finite models

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

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

Consider satisfiability of the concept $\neg A$ w.r.t. the TBox
$\mathcal{T} = \{\top \sqsubseteq \exists P.A \sqcap (\leq 1\, P^-.\top)\}$.

$\neg A$ is satisfied only in an infinite model.



Tableaux rules for number restrictions – Intuition

To deal with:

- $(\geq n\, R.\, C)$: If a node x does not have n R-neighbours satisfying C, **new nodes** satisfying C are created and made R-successors x.
- $(\leq n\, R.\, C)$: 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 tableaux rooted at these nodes.

The correct form of the tableaux 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.

Tableaux rules for qualified number restrictions

Let us consider the following two rules:

$\rightarrow_{\geq}$: **if** $(\geq n\, R.\, C) \in \mathcal{L}(x)$, x is not blocked, and
x has less than n R-neighbours y_i with $C \in \mathcal{L}(y_i)$
then create n new R-successors $y_1, \ldots, y_n$ of x with
$\mathcal{L}(y_i) = \{C\}$ for $1 \leq i \leq n$

$\rightarrow_{\leq}$: **if** $(\leq n\, R.\, C) \in \mathcal{L}(x)$, x is not indirectly blocked, x has $n+1$
R-neighbours $y_0, \ldots, y_n$ with $C \in \mathcal{L}(y_i)$ for $0 \leq i \leq n$, and
there are i, j such that y_j is not an ancestor of y_i
then let $\mathcal{L}(y_i) := \mathcal{L}(y_i) \cup \mathcal{L}(y_j)$, make the successors of y_j to
successors of y_i, and remove y_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**).

Dealing with "yoyo"-effect

To prevent the "yoyo"-effect we use explicit **inequality**:

$\rightarrow_{\geq}$: **if** $(\geq n\, R.\, C) \in \mathcal{L}(x)$, x is not blocked, and x has less than n R-neighbours y_i with $C \in \mathcal{L}(y_i)$
then create n new R-successors $y_1, \ldots, y_n$ of x with $\mathcal{L}(y_i) := \{C\}$ for $1 \leq i \leq n$ and $y_i \neq y_j$ for $1 \leq i < j \leq n$

$\rightarrow_{\leq}$: **if** $(\leq n\, R.\, C) \in \mathcal{L}(x)$, x is not indirectly blocked, x has $n+1$ R-neighbours $y_0, \ldots, y_n$ with $C \in \mathcal{L}(y_i)$ for $0 \leq i \leq n$, and there are i, j s.t. not $y_i \neq y_j$ and y_j is not an ancestor of y_i
then let $\mathcal{L}(y_i) := \mathcal{L}(y_i) \cup \mathcal{L}(y_j)$,
make the successors of y_j to successors of y_i,
add $y_i \neq z$ for each z with $y_j \neq z$, and
remove y_j from the tree

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 x contains a clash if

- $(\leq n\, R.\, C) \in \mathcal{L}(x)$, and
- x has more than n R-neighbours $y_0, \ldots, y_n$ with $y_i \neq y_j$ 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 tableaux rule: choice rule

$\rightarrow_?$: if $(\leq n\, S.\, C) \in \mathcal{L}(x)$, x is not indirectly blocked, and there is an R-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 $E \in \{C, \dot{\neg} C\}$

Does this suffice? **No** . . .

Problem with blocking strategy – Example

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

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

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

z would block y, but we cannot construct a model from this.

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

We use $E(x, y)$ to denote the label of edge (x, y) of the tableaux.

Def.: Double blocking

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

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

Lemma

Let $\mathcal{T}$ be a general $\mathcal{ALCQI}$ TBox and C_0 an $\mathcal{ALCQI}$ concept. Then:

1. The tableaux algorithm terminates when applied to $\mathcal{T}$ and C_0.
2. The rules can be applied such that they generate a clash-free and complete completion tree iff C_0 is satisfiable w.r.t. $\mathcal{T}$.

Tableaux 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 $(p, p|y) \in P^{\mathcal{I}}$ if y is a P-successor of $\mathbf{Tail}(p)$ (and similar for inverse roles), taking care of blocked nodes.
- Danger: assume two successors y, y' of x are blocked by the same node z:
  - Standard unravelling yields one path $[\ldots\ xz]$ for both nodes.
  - Hence, $[\ldots\ x]$ might not have enough P-successors for some $(\geq n\, R.C) \in \mathcal{L}(x)$.
  - Solution: annotate points in the path with blocked nodes: $[\ldots\ \frac{x}{x}\frac{z}{y}] \neq [\ldots\ \frac{x}{x}\frac{z}{y'}]$

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

Tableaux algorithm for ABox satisfiability

Two alternative possibilities:

For DLs without inverse roles: use **pre-completion**.

- Reduce ABox-satisfiability to (several) satisfiability tests by completing the ABox using all but generating rules (i.e., $\rightarrow_\sqcap$, $\rightarrow_\sqcup$, $\rightarrow_\forall$).
- Example: $\{P_1(a,b),\ (A \sqcap \forall P_1.\forall P_2.(\neg A \sqcup B))(a),$
  $P_2(b,a),\ (A \sqcap \exists P_2.\neg B)(b)\}$

For DLs without inverse roles: use **completion forests**.

- Similar to a pre-completion, but root nodes can be related.
- Example: $\{P_1(a,b),\ (A \sqcap \forall P_1.\forall P_2.(\neg A \sqcup B))(a),$
  $P_2(b,a),\ (A \sqcap \exists P_2.(\forall P_2^-.\forall P_1^-.\neg A))(b)\}$

Tableaux 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 (**transitive** 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 tableaux 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.

- Edges are labelled 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 $\forall S.C \in \mathcal{L}(x)$, $R \in E(x, y)$, and (**transitive** S), then $\forall R.C \in \mathcal{L}(x)$.

Outline of Part 6

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

$$\text{WeekDay} \equiv \{\ \text{friday}, \text{monday}, \text{saturday}, \text{sunday}, \text{thursday}, \text{tuesday}, \text{wednesday}\ \}$$

Def.: Nominals

A **nominal** is a concept with cardinality equal to 1, 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 either Bob or Calvin.
3. Either John or Mary is a spy.
4. Everything is created by God.
5. Everybody drives on the left or everybody drives on the right.
6. $(\exists x.A(x)) \rightarrow (\forall x.B(x))$.

Exercise on nominals

1. There are exactly 195 Countries.

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

2. Alice loves either Bob or Calvin.

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

3. Either John or Mary is a spy.

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

Exercise on nominals (cont'd)

4. Everything is created by God.

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

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. Everybody drives on the left or everybody drives on the right.

$$\begin{aligned}\{\text{god}\} \sqsubseteq\ & \forall \text{creates}.(\neg \text{Person} \sqcup \text{LeftDriver}) \sqcup \\ & \forall \text{creates}.(\neg \text{Person} \sqcup \text{RightDriver})\end{aligned}$$

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

$$\{\text{god}\} \sqsubseteq \neg \exists \text{creates}.A \sqcup \forall \text{creates}.B$$

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

1. Properties of $\mathcal{ALC}$
2. Reasoning over $\mathcal{ALC}$ concept expressions
3. Reasoning over $\mathcal{ALC}$ knowledge bases
4. Extensions of $\mathcal{ALC}$
5. Reasoning in extensions of $\mathcal{ALC}$
6. $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$
   - Nominals
   - Boolean TBoxes
   - Reasoning with nominals
   - Enhancing role expressivity

Boolean TBoxes

Def.: Boolean TBox

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

- $A \sqsubseteq B$ is a boolean TBox, for every pair of concepts A and B.
- 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 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:

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

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

$$\{o\} \;\sqsubseteq\; \bigsqcup_{i=1}^{n}(\forall P.(\neg A_i \sqcup B_i)) \;\sqcup\; \bigsqcup_{j=1}^{m}(\exists P.(C_j \sqcap \neg D_j))$$

□

$\mathcal{SHIQ}$ 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:

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

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:

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

- $Cl(C_0)$ 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 contain a **clash** if:

1. $\{A, \neg A\} \subset \mathcal{L}(x)$ 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. $(\mathcal{SHIQ})$

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.

Tableau rules for $\mathcal{SHOIQ}$

$\rightarrow_{\sqcap}$: if 1. $C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and
2. $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
then $\mathcal{L}(x) := \mathcal{L}(x) \cup \{C_1, C_2\}$

$\rightarrow_{\sqcup}$: 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 $C \in \{C_1, C_2\}$

$\rightarrow_{\exists}$: 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 $\mathcal{L}(y) = \{C\}$

$\rightarrow_{\forall}$: 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 $C \notin \mathcal{L}(y)$
then $\mathcal{L}(y) := \mathcal{L}(y) \cup \{C\}$

$\rightarrow_{\forall_+}$: 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 $\forall R.C \notin \mathcal{L}(y)$
then $\mathcal{L}(y) := \mathcal{L}(y) \cup \{\forall R.C\}$

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

$\rightarrow_?$: 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 $E \in \{C, \dot{\neg}C\}$

$\rightarrow_\geq$: 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 $1 \leq i < j \leq n$

$\rightarrow_\leq$: 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)

$\rightarrow_o$: 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 $Merge(x, y)$

$\rightarrow_{o?}$: 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

More expressive role constructs

$\mathcal{SROIQ}$ [Horrocks *et al.*, 2006], at the basis of the OWL 2, and its extension $\mathcal{SROIQB}$ [Rudolph *et al.*, 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 right-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	$\forall R.C$	$\{o \mid \forall o'.\, (o,o') \in R^\mathcal{I} \to o' \in C^\mathcal{I}\}$
existential restr.	$\exists R.C$	$\{o \mid \exists o'.\, (o,o') \in R^\mathcal{I} \land o' \in C^\mathcal{I}\}$
Self concept	$\exists S.$**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	$(\leq n\, S.C)$	$\{o \mid \#\{o' \mid (o,o') \in S^\mathcal{I} \land o' \in C^\mathcal{I}\} \leq n\}$

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:

$$R_1 \circ \cdots \circ R_n \sqsubseteq R \qquad \Longrightarrow \qquad R \longrightarrow R_1 \cdots R_n$$

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

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

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 $1 \leq i \leq n$.

An 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 $\mathcal{SROIQ}$:

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

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

Exercise

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

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

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. only 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:

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

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.

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.

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

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.

Outline of Part 6

References I

[Fischer and Ladner, 1979] Michael J. Fischer and Richard E. Ladner.
Propositional dynamic logic of regular programs.
J. of Computer and System Sciences, 18:194–211, 1979.

[Horrocks *et al.*, 2006] 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)*, pages 57–67, 2006.

[Kazakov, 2008] 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)*, pages 274–284, 2008.

[Pratt, 1979] Vaugham R. Pratt.
Models of program logic.
In *Proc. of the 20th Annual Symp. on the Foundations of Computer Science (FOCS'79)*, pages 115–122, 1979.

References II

[Rudolph *et al.*, 2008] 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)*, volume 5293 of *Lecture Notes in Computer Science*, pages 362–374. Springer, 2008.

[Schild, 1991] Klaus Schild.
A correspondence theory for terminological logics: Preliminary report.
In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 466–471, 1991.

[Schmidt-Schauss and Smolka, 1991] Manfred Schmidt-Schauss and Gert Smolka.
Attributive concept descriptions with complements.
Artificial Intelligence, 48(1):1–26, 1991.

[Tobies, 2000] Stephan Tobies.
The complexity of reasoning with cardinality restrictions and nominals in expressive description logics.
J. of Artificial Intelligence Research, 12:199–217, 2000.

References III

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

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

[Vardi, 1997] Moshe Y. Vardi.
Why is modal logic so robustly decidable.
In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 31, pages 149–184. American Mathematical Society, 1997.