The description logic $\mathcal{ALC}$: terminological part

The description logic $\mathcal{ALC}$

We have investigated (the terminological part of) two types of **lightweight** description logics:

- $\mathcal{EL}$ which has been designed to represent large-scale ontologies such as SNOMED CT and in which terminological reasoning is tractable;
- the DL-Lite family which has been designed to access data using conceptual models such as ER and UML diagrams and in which corresponding querying tasks are often tractable.

We now consider the basic **expressive** description logic $\mathcal{ALC}$. All other expressive description logics are defined as extensions of $\mathcal{ALC}$.

Unfortunately, reasoning in $\mathcal{ALC}$ is not tractable!

$\mathcal{ALC}$ (syntax)

- **Language for $\mathcal{ALC}$ concepts (classes)**
  - concept names $A_0, A_1, \ldots$
  - role names $r_0, r_1, \ldots$
  - the concept $\top$ (often called "thing")
  - the concept $\bot$ (stands for the empty class)
  - the concept constructor $\sqcap$ (often called intersection, conjunction, or simply "and").
  - the concept constructor $\exists$ (often called existential restriction).
  - the concept constructor $\forall$ (often called value restriction).
  - the concept constructor $\sqcup$ (often called union, disjunction, or simply "or").
  - the concept constructor $\neg$ (often called complement or negation).

$\mathcal{ALC}$

$\mathcal{ALC}$ concepts are defined inductively as follows:

- All concept names, $\top$ and $\bot$ are $\mathcal{ALC}$ concepts;
- if C is a $\mathcal{ALC}$ concept, then $\neg C$ is a $\mathcal{ALC}$ concept;
- if C and D are $\mathcal{ALC}$ concepts and r is a role names, then

$$(C \sqcap D), \quad (C \sqcup D), \quad \exists r.C, \quad \forall r.C$$

  are $\mathcal{ALC}$ concepts.

A **$\mathcal{ALC}$ concept-inclusion** is of the form

$$C \sqsubseteq D,$$

where C, D are $\mathcal{ALC}$ concepts.

Examples of $\mathcal{ALC}$ concepts

- **Person** $\sqcap \forall$**hasChild.Male** (everybody whose children are all male);
- **Person** $\sqcap \forall$**hasChild.Male** $\sqcap \exists$**hasChild.**$\top$ (everybody who has a child and whose children are all male).
- **Living_being** $\sqcap \neg$**Human_being** (all living beings that are not human beings);
- **Student** $\sqcap \neg\exists$**interested_in.Mathematics** (all students not interested in mathematics);
- **Student** $\sqcap \forall$**drinks.tea** (all students who only drink tea).
- $\exists$**hasChild.Male** $\sqcup \forall$**hasChild.**$\bot$ (everybody who has a son or no child).

Description logics: $\mathcal{ALC}$ (semantics)

Interpretations are defined as before:

- Recall that an **interpretation** is a structure $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ in which
  - $\Delta^{\mathcal{I}}$ is the **domain** (a non-empty set)
  - $\cdot^{\mathcal{I}}$ is an **interpretation function** that maps:
    - every concept name A to a subseteq $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ $\quad (A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}})$
    - every role name r to a binary relation $r^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$ $\quad (r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})$

- interpretation of **complex concepts** in $\mathcal{I}$:

  (C, D are concepts and r a role name)

  - $(\top)^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $(\bot)^{\mathcal{I}} = \emptyset$
  - $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
  - $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$ and $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
  - $(\forall r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid$ for all $y \in \Delta^{\mathcal{I}}$ with $(x, y) \in r^{\mathcal{I}}$ we have $y \in C^{\mathcal{I}}\}$
  - $(\exists r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid$ exists $y \in \Delta^{\mathcal{I}}$ such that $(x, y) \in r^{\mathcal{I}}$ and $y \in C^{\mathcal{I}}\}$

$(\forall r.C)^{\mathcal{I}}$ 对所有 y，若 x, y 有 r 关系，则

Example

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be defined by setting

- $\Delta^{\mathcal{I}} = \{a, b, c, d\}$;
- $A^{\mathcal{I}} = \{b, d\}$, $B^{\mathcal{I}} = \{c\}$;
- $r^{\mathcal{I}} = \{(a, b), (a, c)\}$, $s^{\mathcal{I}} = \{(a, b), (a, d)\}$.

Then

- $(\forall r.A)^{\mathcal{I}} = \{b, c, d\}$, $(\forall s.A)^{\mathcal{I}} = \{a, b, c, d\}$;
- $(\exists r.A \sqcap \forall r.A)^{\mathcal{I}} = \emptyset$, $(\exists s.A \sqcap \forall s.A)^{\mathcal{I}} = \{a\}$;
- $(\exists r.B \sqcap \exists r.A)^{\mathcal{I}} = \{a\}$, $(\exists r.(A \sqcap B))^{\mathcal{I}} = \emptyset$;
- $(\forall r.\neg A)^{\mathcal{I}} = \{b, c, d\}$, $(\forall s.\neg A)^{\mathcal{I}} = \{b, c, d\}$.

Examples of equivalent concepts (classes)

For all interpretations $\mathcal{I}$ and all concepts C, D and roles r the following holds:

- $(\neg\neg C)^{\mathcal{I}} = C^{\mathcal{I}}$;
- $(\forall r.C)^{\mathcal{I}} = (\neg\exists r.\neg C)^{\mathcal{I}}$;
- $(\neg(C \sqcap D))^{\mathcal{I}} = (\neg C \sqcup \neg D)^{\mathcal{I}}$;
- $(\neg(C \sqcup D))^{\mathcal{I}} = (\neg C \sqcap \neg D)^{\mathcal{I}}$;
- $(\neg\exists r.C)^{\mathcal{I}} = (\forall r.\neg C)^{\mathcal{I}}$;
- $(\neg\forall r.C)^{\mathcal{I}} = (\exists r.\neg C)^{\mathcal{I}}$;
- $(C \sqcap \neg C)^{\mathcal{I}} = \bot^{\mathcal{I}} = \emptyset$;
- $(C \sqcup \neg C)^{\mathcal{I}} = \top^{\mathcal{I}} = \Delta^{\mathcal{I}}$. $= C^{\mathcal{I}} \cup (\neg C)^{\mathcal{I}} = C^{\mathcal{I}} \cup (\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}) = \Delta^{\mathcal{I}}$

Concept inclusions and TBoxes

- A **$\mathcal{ALC}$-concept inclusion** is an expression

$$C \sqsubseteq D,$$

  where C and D are $\mathcal{ALC}$-concepts.

- A **$\mathcal{ALC}$-TBox** is a finite set T of $\mathcal{ALC}$-concept inclusions. $\mathcal{ALC}$-terminologies and acyclic terminologies are defined following the definition for $\mathcal{EL}$.

Semantics: exactly the same as for $\mathcal{EL}$

Let $\mathcal{I}$ be an interpretation, $C \sqsubseteq D$ a $\mathcal{ALC}$ concept inclusion, and $\mathcal{T}$ a $\mathcal{ALC}$-TBox.

- Now set $\mathcal{I} \models C \sqsubseteq D$ if, and only if, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. In words:
  - $\mathcal{I}$ satisfies $C \sqsubseteq D$ or
  - $C \sqsubseteq D$ is true in $\mathcal{I}$ or
  - $\mathcal{I}$ is a model of $C \sqsubseteq D$.
- We set $\mathcal{I} \models \mathcal{T}$ if, and only if, $\mathcal{I} \models E \sqsubseteq F$ for all $E \sqsubseteq F$ in $\mathcal{T}$. In words:
  - $\mathcal{I}$ satisfies $\mathcal{T}$ or
  - $\mathcal{I}$ is a model of $\mathcal{T}$.

Example

Let $\mathcal{T} = \{A \sqsubseteq \exists r.B\}$. Then

$$\mathcal{T} \not\models A \sqsubseteq \forall r.B.$$

To see this, construct an interpretation $\mathcal{I}$ such that

- $\mathcal{I} \models \mathcal{T}$;
- $\mathcal{I} \not\models A \sqsubseteq \forall r.B$.

Let $\mathcal{I}$ be defined by

- $\Delta^{\mathcal{I}} = \{a, b, c\}$;
- $A^{\mathcal{I}} = \{a\}$;
- $r^{\mathcal{I}} = \{(a, b), (a, c)\}$;
- $B^{\mathcal{I}} = \{b\}$.

Then $A^{\mathcal{I}} = \{a\} \subseteq \{a\} = (\exists r.B)^{\mathcal{I}}$ and so $\mathcal{I} \models \mathcal{T}$. But $A^{\mathcal{I}} \not\subseteq \{b, c\} = (\forall r.B)^{\mathcal{I}}$ and so $\mathcal{I} \not\models A \sqsubseteq \forall r.B$.

Example

Let $\mathcal{T} = \{A \sqsubseteq \forall r.B\}$. Then

$$\mathcal{T} \not\models A \sqsubseteq \exists r.B.$$

To see this, construct an interpretation $\mathcal{I}$ such that

- $\mathcal{I} \models \mathcal{T}$;
- $\mathcal{I} \not\models A \sqsubseteq \exists r.B$.

Let $\mathcal{I}$ be defined by

- $\Delta^{\mathcal{I}} = \{a\}$;
- $A^{\mathcal{I}} = \{a\}$;
- $r^{\mathcal{I}} = \emptyset$;
- $B^{\mathcal{I}} = \emptyset$.

Then $A^{\mathcal{I}} = \{a\} \subseteq \{a\} = (\forall r.B)^{\mathcal{I}}$ and so $\mathcal{I} \models \mathcal{T}$. But $A^{\mathcal{I}} \not\subseteq \emptyset = (\exists r.B)^{\mathcal{I}}$ and so $\mathcal{I} \not\models A \sqsubseteq \exists r.B$.

Domain and Range Restrictions in $\mathcal{ALC}$

Recall that

$$\exists r.\top \sqsubseteq C$$

states that the domain of r is contained in C. This inclusion is in $\mathcal{ALC}$.

Recall that, on the other hand,

$$\exists r^{-}.\top \sqsubseteq C$$

states that the range of r is contained in C. This inclusion is not in $\mathcal{ALC}$. We can express such a range restriction in $\mathcal{ALC}$, however, as

$$\top \sqsubseteq \forall r.C$$

Modelling in $\mathcal{ALC}$: Disjoint Classes

In $\mathcal{EL}$ we cannot represent that two (or more) classes are disjoint (have no common elements). In $\mathcal{ALC}$ we can state this in many different ways.

'Vegetable, Meat, Seafood, and Cheese are mutually disjoint' can be represented by the inclusions

$$\textbf{Vegetable} \sqcap \textbf{Meat} \sqsubseteq \bot, \quad \textbf{Vegetable} \sqcap \textbf{Seafood} \sqsubseteq \bot, \quad \textbf{Vegetable} \sqcap \textbf{Cheese} \sqsubseteq \bot,$$

$$\textbf{Meat} \sqcap \textbf{Seafood} \sqsubseteq \bot, \quad \textbf{Meat} \sqcap \textbf{Cheese} \sqsubseteq \bot, \quad \textbf{Seafood} \sqcap \textbf{Cheese} \sqsubseteq \bot,$$

Equivalently, we could write $\textbf{Vegetable} \sqsubseteq \neg\textbf{Meat}$, etc.

Note, however, that

$$\textbf{Vegetable} \sqcap \textbf{Meat} \sqcap \textbf{Seafood} \sqcap \textbf{Cheese} \sqsubseteq \bot$$

is a weaker assertion stating that nothing is Vegetable, Meat, Seafood, and Cheese at the same time. So there could still be something that is Meat and Cheese.

Modelling in $\mathcal{ALC}$: typical mistake for $\forall$

Assume we state that the domain of hasTopping is pizza (only pizza's have a topping):

$$\exists \mathbf{hasTopping}.\top \sqsubseteq \mathbf{Pizza}$$

and we add that ice cream cones have a topping that is ice cream:

$$\mathbf{IceCreamCone} \sqsubseteq \exists \mathbf{hasTopping}.\mathbf{IceCream}$$

then, if we assume that ice cream cones and pizzas are disjoint

$$\mathbf{Pizza} \sqcap \mathbf{IceCreamCone} \sqsubseteq \bot,$$

we obtain that the class IceCreamCone is empty!

Reasoning for $\mathcal{ALC}$ (without TBox)

We first consider reasoning without TBoxes:

- **Subsumption**. We say that a concept inclusion $C \sqsubseteq D$ follows from the empty TBox (or that C is subsumed by D) if, and only if, for all interpretations $\mathcal{I}$ we have that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. In this case, we often write $\emptyset \models C \sqsubseteq D$.
- **Concept satisfiability**. A concept C is satisfiable if, and only if, there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$.

We have: $\emptyset \models C \sqsubseteq D$ if, and only if, $C \sqcap \neg D$ is not satisfiable. Thus, in $\mathcal{ALC}$, subsumption is reducible to concept satisfiability.

We give an algorithm deciding whether a $\mathcal{ALC}$-concept C is satisfiable.

Remark This problem is *not* tractable. Its complexity is between NP-complete and ExpTime-complete (precisely: PSpace-complete). The algorithm we present requires exponential time.

Satisfiability of Concepts: example 1

Q: Is $(\forall \text{hasChild}.\text{Male}) \sqcap (\exists \text{hasChild}.\neg\text{Male})$ satisfiable?

Let us try to construct an **interpretation** satisfying this concept

(1)		x : $(\forall \text{hasChild}.\text{Male}) \sqcap (\exists \text{hasChild}.\neg\text{Male})$
(2)	from (1)	x : $\forall \text{hasChild}.\text{Male}$
(3)	from (1)	x : $\exists \text{hasChild}.\neg\text{Male}$
(4)	from (3)	(x, y) : hasChild and y : $\neg$Male, for fresh y
(5)	from (2) & (4)	y : Male
(6)	from (4) & (5)	**contradiction:** y : Male and y : $\neg$Male

A: the concept is **not satisfiable!**

Satisfiability of Concepts: example 2

Q: Is $(\forall \text{hasChild.Male}) \sqcap (\exists \text{hasChild.Male})$ satisfiable?

Let us try to construct a **interpretation** satisfying this concept

(1)		x : $(\forall \text{hasChild.Male}) \sqcap (\exists \text{hasChild.Male})$
(2)	from (1)	x : $\forall \text{hasChild.Male}$
(3)	from (1)	x : $\exists \text{hasChild.Male}$
(4)	from (3)	(x, y) : hasChild and y : Male, for fresh y

A: the concept is **satisfiable** and a satisfying model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is

$$\Delta^{\mathcal{I}} = \{x, y\}, \qquad \text{Male}^{\mathcal{I}} = \{y\}, \qquad \text{hasChild}^{\mathcal{I}} = \{(x, y)\}$$

Then $x \in \big((\forall \text{hasChild.Male}) \sqcap (\exists \text{hasChild.Male})\big)^{\mathcal{I}}$

Satisfiability of Concepts: example 3

Q: Is $\forall r.(\neg C \sqcup D) \sqcap \exists r.(C \sqcap D)$ satisfiable?

(1)		$x : \forall r.(\neg C \sqcup D) \sqcap \exists r.(C \sqcap D)$
(2)	from (1)	$x : \forall r.(\neg C \sqcup D)$
(3)	from (1)	$x : \exists r.(C \sqcap D)$
(4)	from (3)	$(x, y) : r$ and $y : C \sqcap D$, for fresh y
(5)	from (4)	$y : C$
(6)	from (4)	$y : D$
(7)	from (2)	$y : \neg C \sqcup D$

Two ways of continue **(branching!)**:

(8.1)	from (7)	$y : \neg C$
(8.2)	from (7)	$y : D$

A: (8.1) is a **contradiction**, while (8.2) is not and yields a **satisfying model**

Tableau Methods

How can we prove satisfiability of a concept?

Achieved by applying **tableau methods**

(set of **completion rules** operating on **constraint systems** or **tableaux**)

Proof procedure:

$\exists r.\neg A$ 是 $\sqsubseteq$ $\neg\exists r.A$ 不是

- transform a given concept into **Negation Normal Form** (NNF)
  (all occurrences of negations are in front of concept names)
- apply **completion rules** in arbitrary order as long as possible.
- the concept is **satisfiable** if, and only if, a **clash-free** tableau can be derived to which no completion rule is applicable.

Negation Normal Form (NNF)

A concept is in **Negation Normal Form** (NNF)
if all occurrences of negations in it are in front of *concept names*

Every $\mathcal{ALC}$-concept can be transformed into an equivalent one in NNF
using the following rules:

$\neg\top$	$\equiv$	$\bot$	
$\neg\bot$	$\equiv$	$\top$	
$\neg\neg C$	$\equiv$	C	
$\neg(C \sqcap D)$	$\equiv$	$\neg C \sqcup \neg D$	(De Morgan's law)
$\neg(C \sqcup D)$	$\equiv$	$\neg C \sqcap \neg D$	(De Morgan's law)
$\neg\forall r.C$	$\equiv$	$\exists r.\neg C$	
$\neg\exists r.C$	$\equiv$	$\forall r.\neg C$	

Negation Normal Form: example

Transform the concept

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

to an equivalent concept in negation normal form.

$\neg\exists r.(A \sqcap \neg B) \sqcup \neg\forall r.(\neg A \sqcup \neg B) \equiv$ (use $\neg\exists r.D \equiv \forall r.\neg D$)

$\forall r.\neg(A \sqcap \neg B) \sqcup \neg\forall r.(\neg A \sqcup \neg B) \equiv$ (use $\neg(A \sqcap D) \equiv \neg A \sqcup \neg D$)

$\forall r.(\neg A \sqcup \neg\neg B) \sqcup \neg\forall r.(\neg A \sqcup \neg B) \equiv$ (use $\neg\neg B \equiv B$)

$\forall r.(\neg A \sqcup B) \sqcup \neg\forall r.(\neg A \sqcup \neg B) \equiv$ (use $\neg\forall r.D \equiv \exists r.\neg D$)

$\forall r.(\neg A \sqcup B) \sqcup \exists r.\neg(\neg A \sqcup \neg B) \equiv$ (use $\neg(C \sqcup D) \equiv \neg C \sqcap \neg D$)

$\forall r.(\neg A \sqcup B) \sqcup \exists r.(\neg\neg A \sqcap \neg\neg B) \equiv$ (use $\neg\neg C \equiv C$)

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

Tableau Calculus for $\mathcal{ALC}$ concept satisfiability

Constraint: expression of the form $x : C$ or $(x, y) : r$,
where C is a concept in NNF and r a role name

Constraint system: a finite non-empty set S of constraints

Completion rules: $S \rightarrow S'$, where S' is a constraint system containing S

Clash: S contains **clash** if

$$\{ x : A, \quad x : \neg A \} \subseteq S, \quad \text{for some } x \text{ and concept name } A$$

Aim: starting from $S_0 = \{x : C\}$ apply completion rules to construct a clash-free system S_n to which no completion rule is applicable

- If this is possible, then we can extract a **model satisfying** C
- Otherwise, C is **not satisfiable**.

Completion Rules for $\mathcal{ALC}$ concept satisfiability (1)

$$S \rightarrow_{\sqcap} S \cup \{ x : C,\ x : D \}$$

if (a) $x : C \sqcap D$ is in S
(b) $x : C$ and $x : D$ are not both in S

$$S \rightarrow_{\sqcup} S \cup \{ x : E \}$$

if (a) $x : C \sqcup D$ is in S
(b) neither $x : C$ nor $x : D$ is in S
(c) $E = C$ or $E = D$ **(branching!)**

NB: Non-deterministically add any of the disjuncts to the constraint system

NB: Clashes eliminate branches in the OR tree

Completion Rules for $\mathcal{ALC}$ concept satisfiability (2)

$$S \rightarrow_{\forall} S \cup \{ y : C \}$$

if (a) $x : \forall r.C$ is in S
(b) $(x, y) : r$ is in S
(c) $y : C$ is not in S

NB: Only applicable if role successors can be found

$$S \rightarrow_{\exists} S \cup \{ (x, y) : r,\ y : C \}$$

if (a) $x : \exists r.C$ is in S
(b) y is a fresh individual
(c) there is no z such that
both $(x, z) : r$ and $z : C$ are in S

NB: The only rule that creates new individuals in a constraint system

Tableau Example 1

We check whether $(A \sqcap \neg A) \sqcup B$ is satisfiable.

It is in NNF, so we can directly apply the tableau algorithm to

$$S_0 = \{x : (A \sqcap \neg A) \sqcup B\}$$

The only rule applicable is $\rightarrow_{\sqcup}$. We have two possibilities.

Firstly we can try

$$S_1 = S_0 \cup \{x : A \sqcap \neg A\}.$$

Then we can apply $\rightarrow_{\sqcap}$ and obtain

$$S_2 = S_1 \cup \{x : A, x : \neg A\}$$

We have obtained a clash, thus this choice was unsuccessful.

Secondly, we can try

$$S_1^* = S_0 \cup \{x : B\}.$$

No rule is applicable to S_1^* and it does not contain a clash. Thus, $(A \sqcap \neg A) \sqcup B$ is satisfiable.

A model $\mathcal{I}$ satisfying it is given by

$$\Delta^{\mathcal{I}} = \{x\}, \quad B^{\mathcal{I}} = \{x\}, \quad A^{\mathcal{I}} = \emptyset.$$

Tableau Example 2

We check whether $C = A \sqcap \exists r.\exists s.B \sqcap \forall r.\neg B$ is satisfiable.

It is in NNF, so we can directly apply the tableau algorithm to

$$S_0 = \{x : A \sqcap \exists r.\exists s.B \sqcap \forall r.\neg B\}$$

An application of $\rightarrow_\sqcap$ gives

$$S_1 = S_0 \cup \{x : A, x : \exists r.\exists s.B \sqcap \forall r.\neg B\}$$

An application of $\rightarrow_\sqcap$ gives

$$S_2 = S_1 \cup \{x : \exists r.\exists s.B, x : \forall r.\neg B\}$$

An application of $\rightarrow_\exists$ gives

$$S_3 = S_2 \cup \{(x, y) : r, y : \exists s.B\}$$

An application of $\rightarrow_{\exists}$ gives

$$\boldsymbol{S_4 = S_3 \cup \{(y, z) : s, z : B\}}$$

Tableau Example 2

Recall that

$$S_4 = S_3 \cup \{(y, z) : s, z : B\}$$

An application of $\rightarrow_\forall$ gives

$$S_5 = S_4 \cup \{y : \neg B\}$$

No rule is applicable to S_5 and S_5 contains no clash. Thus, the concept C is satisfiable.

A model $\mathcal{I}$ of C is given by

$$\Delta^{\mathcal{I}} = \{x, y, z\}, \quad A^{\mathcal{I}} = \{x\}, \quad B^{\mathcal{I}} = \{z\}, \quad r^{\mathcal{I}} = \{(x, y)\}, \quad s^{\mathcal{I}} = \{(y, z)\}$$

Tableau Example 3

We check whether $C = \exists r.A \sqcap \exists r.\neg A$ is satisfiable.

C is in NNF, so we can directly apply the tableau algorithm to

$$S_0 = \{x : \exists r.A \sqcap \exists r.\neg A\}$$

An application of $\rightarrow_{\sqcap}$ gives

$$S_1 = S_0 \cup \{x : \exists r.A, x : \exists r.\neg A\}$$

An application of $\rightarrow_{\exists}$ gives

$$S_2 = S_1 \cup \{(x, y) : r, y : A\}$$

Tableau Example 3

Recall that

$$S_2 = S_1 \cup \{(x, y) : r, y : A\}$$

Another application of $\rightarrow_\exists$ gives

$$S_3 = S_2 \cup \{(x, z) : r, z : \neg A\}$$

No rule is applicable to S_3 and S_3 contains no clash. Thus, C is satisfiable.

A model $\mathcal{I}$ of C is given by

$$\Delta^{\mathcal{I}} = \{x, y, z\}, \quad A^{\mathcal{I}} = \{y\}, \quad r^{\mathcal{I}} = \{(x, y), (x, z)\}$$

Analysing the Tableau Calculus

To show that the tableau does what it is supposed to do one has to show

- Soundness: If the concept is satisfiable, then there is a branch without clash such that no rule is applicable;
- Termination: The tableau terminates after finitely many steps for any input concept in NNF;
- Completeness: If there is a branch without clash such that no rule is applicable, then the concept is satisfiable.

Tableau Calculus: Soundness

- Suppose that a constraint system S is satisfiable and

$$S \rightarrow_{\sqcap} S', \quad S \rightarrow_{\forall} S' \quad \text{or} \quad S \rightarrow_{\exists} S'.$$

  Then S' is also satisfiable.

- If

$$S \rightarrow_{\sqcup} S' \quad \text{and} \quad S \rightarrow_{\sqcup} S''$$

  then one of S' and S'' is satisfiable (or perhaps both).

Thus, having started with a satisfiable constraint system
we cannot derive clashes in all branches

Tableau Calculus: Termination

For every constraint system S_0,
there is no infinite sequence of the form

$$S_0, S_1, S_2, \ldots$$

such that S_{i+1} is obtained form S_i
by an application of one of the completion rules

Proof: All rules but $\rightarrow_\forall$ are never applied twice to the same constraint

$\rightarrow_\forall$ is never applied to an individual x more times than
the number of direct successors of x (i.e., y such that $(x, y) : r$),
which is bounded by the length of the concept

Each rule application to a constraint $y : C$
adds constraints $z : D$ such that D is a subconcept of C

Tableau Calculus: Completeness

If starting from $S_0 = \{x : C\}$ and applying the completion rules we construct a **clash-free** constraint system S_n to which no rule is applicable then C is satisfiable

S_n determines an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$:

- $\Delta^{\mathcal{I}}$ contains all individuals in S_n
- for $x \in \Delta^{\mathcal{I}}$ and a concept name A,
$$x \in A^{\mathcal{I}} \quad \text{iff} \quad x : A \text{ is in } S_n$$
- for $x, y \in \Delta^{\mathcal{I}}$ and a role name r,
$$(x, y) \in r^{\mathcal{I}} \quad \text{iff} \quad (x, y) : r \text{ is in } S_n$$

It is easy to check that C is satisfied in $\mathcal{I}$, i.e., $C^{\mathcal{I}} \neq \emptyset$

Reasoning Services for $\mathcal{ALC}$ (with TBox)

- **Subsumption w.r.t. TBoxes**. We say that a concept inclusion $C \sqsubseteq D$ follows from a TBox T if, and only if, every interpretation $\mathcal{I}$ that is a model of T is a model of $C \sqsubseteq D$. In this case, we often write $T \models C \sqsubseteq D$.
- **Concept satisfiability w.r.t. TBoxes**. A concept C is satisfiable w.r.t. a TBox T if, and only if, there exists an interpretation $\mathcal{I}$ that is a model of T such that $C^{\mathcal{I}} \neq \emptyset$.
- **TBox satisfiability**. A TBox T is satisfiable if, and only if, there exists a model of T.

We have the following reductions to concept satisfiability w.r.t. TBoxes:

- $T \models C \sqsubseteq D$ if, and only if, $C \sqcap \neg D$ is not satisfiable w.r.t. T.
- T is satisfiable if, and only if, A is satisfiable w.r.t. T (A a fresh concept name).

Thus, it is sufficient to design an algorithm checking concept satisfiability w.r.t. TBoxes.

Discussion

The concept satisfiability problem w.r.t. $\mathcal{ALC}$-TBoxes is ExpTime-complete. Thus, it is not tractable:

- There is no guarantee that existing implementations (or future implementations) of algorithms for this problem will terminate in a reasonable amount of time for every $\mathcal{ALC}$-TBox.
- Nevertheless, there are a number of systems (FACT, PELLET, RACER) which work for most currently existing TBoxes.

Reasoning with TBoxes

Given a TBox $\mathcal{T}$ and a concept C,

how to determine whether $\mathcal{T} \cup \{ x : C \}$ has a model

(**concept satisfiability w.r.t. a TBox**)

Note that, for any interpretation $\mathcal{I}$ and any two concepts C and D,

$$\mathcal{I} \models C \sqsubseteq D \quad \text{iff} \quad \mathcal{I} \models \top \sqsubseteq \neg C \sqcup D$$

So, $C \sqsubseteq D$ is equivalent to $\top \sqsubseteq \neg C \sqcup D$.

The **initial constraint system** S_0 for $\mathcal{T} \cup \{ x : C \}$ is defined by

$$S_0 = \{ x : C \} \cup \{ \top \sqsubseteq \neg C \sqcup D \mid C \sqsubseteq D \in \mathcal{T} \}$$

So, now we have three different types of constraints:

$$y : D \qquad (x, y) : r \qquad \top \sqsubseteq D$$

Reasoning with TBoxes (cont.)

$$S \;\rightarrow_U\; S \cup \{\, x : D \,\}$$

if (a) $\top \sqsubseteq D$ is in S
(b) x occurs in S
(c) $x : D$ is not in S

The tableau algorithm based on rules

$$\rightarrow_\sqcap, \quad \rightarrow_\sqcup, \quad \rightarrow_\forall, \quad \rightarrow_\exists \quad \text{and} \quad \rightarrow_U$$

does not terminate:

in general, even if $\mathcal{T} \cup \{\, x : C \,\}$ has model,
the algorithm can produce an **infinite** model for it
(although finite models exist)

see the next slide for an example...

Reasoning with TBoxes: example

$$
\begin{array}{ll}
S_0 & = \{\, x_0 : \top, \quad \top \sqsubseteq \exists r.A \,\} \\
S_0 \rightarrow_U S_1 & = S_0 \cup \{\, x_0 : \exists r.A \,\} \\
S_1 \rightarrow_\exists S_2 & = S_1 \cup \{\, (x_0, x_1) : r, \quad x_1 : A \,\} \\
S_2 \rightarrow_U S_3 & = S_2 \cup \{\, x_1 : \exists r.A \,\} \\
S_3 \rightarrow_\exists S_4 & = S_3 \cup \{\, (x_1, x_2) : r, \quad x_2 : A \,\} \\
S_4 \rightarrow_U S_5 & = S_4 \cup \{\, x_2 : \exists r.A \,\} \\
\quad \ldots & \ldots
\end{array}
$$

This gives an **infinite model** which can easily be reconstructed into a finite one

Rule $\rightarrow_\exists$ can be modified in such a way that
the resulting algorithm always **terminates**
(using so-called *blocking technique*)