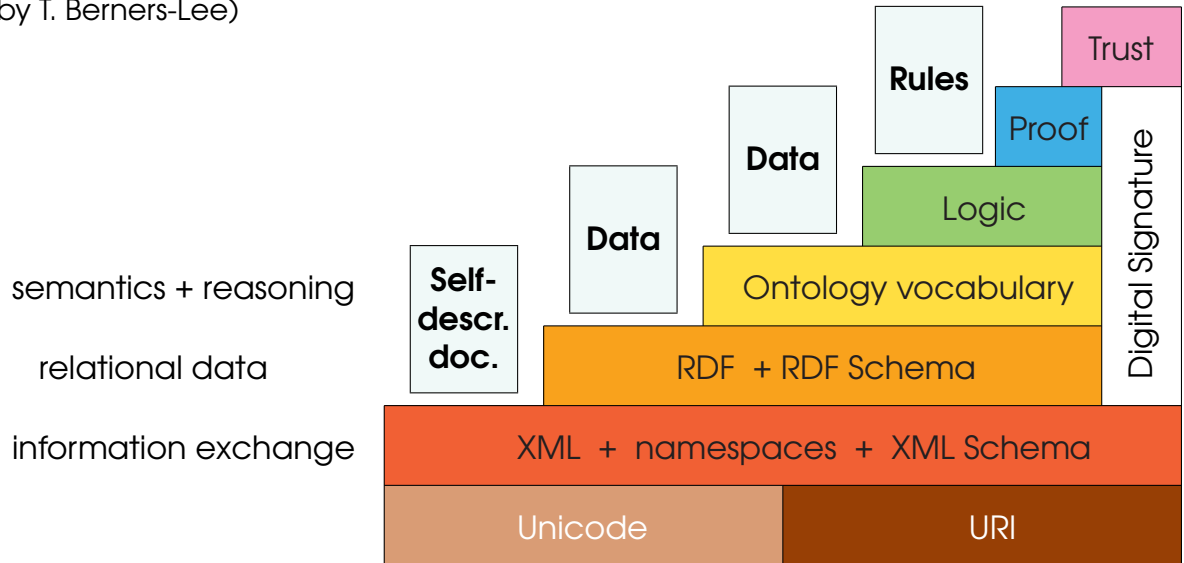


# Layered approach to the Semantic Web

(by T. Berners-Lee)



Broadly, **inference** on the Semantic Web means discovering new relationships. On the Semantic Web, data is modelled as a set of relationships between resources. **Inference** means that automatic procedures can generate new relationships based on the data and some additional information. Whether the new relationships are explicitly added to the data, or are returned at query time, is an implementation issue.

# OWL 2: automated reasoning

with description logic *ALC*

for a list of OWL reasoners see

<http://www.w3.org/2001/sw/wiki/OWL/Implementations>

## What are Description Logics?

- A **family** of logic-based **Knowledge Representation formalisms**
  - descendants of **semantic networks**
  - describe domain in terms of **concepts** (classes), **roles** (properties, relationships) and **individuals**

Distinguished by:

- **formal semantics** (typically model-theoretic)
  - decidable fragments of first-order logic
  - closely related to propositional modal and dynamic logics
- provision of **inference services**
  - sound and complete decision procedures for key problems
  - highly optimised reasoning systems

# DL architecture

## Knowledge Base (KB)

**TBox** (terminological box, schema)

$\text{Man} \equiv \text{Human} \sqcap \text{Male}$   
 $\text{Father} \equiv \text{Man} \sqcap \exists \text{hasChild.T}$   
...

**ABox** (assertion box, data)




$\text{john} : \text{Man}$   
 $(\text{john}, \text{mary}) : \text{hasChild}$   
...

**Inference System**

**Interface**

## Description logic $\mathcal{ALC}$

The language of  $\mathcal{ALC}$  consists of:

- **concept names**  $A_0, A_1, \dots$  (also use  $A, B, B_0, B_1, \dots$ )  
concept names denote sets of objects  **classes** in OWL  
typical examples are 'Person' and 'Female'
- **role names**  $R_0, R_1, \dots$  (also use  $R, S, S_0, S_1, \dots$ )  
role names denote sets of pairs of objects  **object properties** in OWL  
typical examples are 'hasChild' and 'loves'
- **individual names**  $a_0, a_1, \dots$  (also use  $a, b, c, \dots$ )  
individual names denote objects  **individuals** in OWL  
typical examples are 'john' and 'mary'

## The language of *ALC* (cont.)

- concept  $\top$   $\longrightarrow$  **owl:Thing** (denotes the set of all objects in the domain)
- concept  $\perp$   $\longrightarrow$  **owl:Nothing** (denotes the empty set  $\emptyset$ )
- concept constructor  $\sqcap$  (often called intersection, conjunction or simply 'and')  
 $\longrightarrow$  **owl:ObjectIntersectionOf**
- concept constructor  $\sqcup$  (often called union, disjunction or simply 'or')  
 $\longrightarrow$  **owl:ObjectUnionOf**
- concept constructor  $\neg$  (often called complement, negation or simply 'not')  
 $\longrightarrow$  **owl:ObjectComplementOf**
- concept constructor  $\exists$  (often called existential restriction)  
 $\longrightarrow$  **owl:ObjectSomeValuesFrom**
- concept constructor  $\forall$  (often called universal restriction)  
 $\longrightarrow$  **owl:ObjectAllValuesFrom**

## Definition of $\mathcal{ALC}$ concepts

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

- all concept names are  $\mathcal{ALC}$  concepts
- $\top$  and  $\perp$  are  $\mathcal{ALC}$  concepts
- if  $C$  is an  $\mathcal{ALC}$  concept, then  $\neg C$  is an  $\mathcal{ALC}$  concept
- if  $C$  and  $D$  are  $\mathcal{ALC}$  concepts and  $R$  is a role name, then

$$C \sqcap D, \quad C \sqcup D, \quad \exists R.C, \quad \forall R.C$$

are  $\mathcal{ALC}$  concepts

- nothing else is an  $\mathcal{ALC}$ -concept

## Examples of *ALC* concepts

Suppose **Human** and **Female** are concept names,

**hasChild**, **gender**, **hasParent** are role names

Then we obtain the following *ALC*-concepts:

- $\exists\text{hasChild.T}$  (everybody who has a child)
  - $\text{Human} \sqcap \exists\text{hasChild.T}$  (a human who has a child)
  - $\text{Human} \sqcap \exists\text{hasChild.Human}$  (a human who has a child that is human)
  - $\text{Human} \sqcap \exists\text{gender.Female}$  (a woman)
  - $\text{Human} \sqcap \exists\text{hasChild.T} \sqcap \exists\text{hasParent.T}$  (a human with a child and a parent)
  - $\text{Human} \sqcap \exists\text{hasChild}.\exists\text{gender.Female}$  (a human who has a daughter)
  - $\text{Human} \sqcap \exists\text{hasChild}.\exists\text{hasChild.T}$  (a human who has a grandchild)
- ‘ $\sqcap \exists\text{hasChild}$ ’ is not an *ALC*-concept. **why?**



## Examples of *ALC* concepts (cont.)

- $\text{Person} \sqcap \forall \text{hasChild.Male}$  (a person all of whose children are males)  
(in particular, a person without children!)
- $\text{Person} \sqcap \forall \text{hasChild.Male} \sqcap \exists \text{hasChild.T}$   
(everybody who has a child and whose children are all males)
- $\text{LivingBeing} \sqcap \neg \text{HumanBeing}$  (all living beings that are not human beings)
- $\text{Student} \sqcap \neg \exists \text{interestedIn.Mathematics}$  (all students not interested in mathematics)
- $\text{Student} \sqcap \forall \text{drinks.Tea}$  (all students who only drink tea)  
(in particular, the students who do not drink anything!)
- $\exists \text{hasChild.Male} \sqcup \forall \text{hasChild.}\perp$  (everybody who has a son or no child at all)

## OWL as DL: Class Constructors

$A$	$A$
<b>owl:Thing</b>	$\top$
<b>owl:Nothing</b>	$\perp$
<b>ObjectIntersectionOf</b> ( $C_1 C_2 \dots C_n$ )	$C_1 \sqcap C_2 \sqcap \dots \sqcap C_n$
<b>ObjectUnionOf</b> ( $C_1 C_2 \dots C_n$ )	$C_1 \sqcup C_2 \sqcup \dots \sqcup C_n$
<b>ObjectComplementOf</b> ( $C$ )	$\neg C$
<b>ObjectOneOf</b> ( $a_1 a_2 \dots a_n$ )	$\{a_1\} \sqcup \{a_2\} \sqcup \dots \sqcup \{a_n\}$
<b>ObjectAllValuesFrom</b> ( $R C$ )	$\forall R.C$
<b>ObjectSomeValuesFrom</b> ( $R C$ )	$\exists R.C$
<b>ObjectMinCardinality</b> ( $R n C$ )	$\geq n R.C$
<b>ObjectMaxCardinality</b> ( $R n C$ )	$\leq n R.C$
<b>ObjectHasValue</b> ( $R a$ )	$\exists R.\{a\}$

## *ALC* concept definitions and descriptions

Let  $A$  be a concept name and let  $C$  be an *ALC* concept. Then

- $A \equiv C$  is a **concept definition** (reads ‘ $A$  is equivalent to  $C$ ’)  
 $C$  gives necessary and sufficient conditions for being an  $A$
- $A \sqsubseteq C$  is a **concept description** (reads ‘ $A$  is subsumed by  $C$ ’)  
 $C$  describes only necessary conditions for being an  $A$

cf. [https://en.wikipedia.org/wiki/Necessity\\_and\\_sufficiency](https://en.wikipedia.org/wiki/Necessity_and_sufficiency)

### Examples:

- $\text{Father} \equiv \text{Person} \sqcap \exists \text{gender.Male} \sqcap \exists \text{hasChild.T}$
- $\text{Student} \equiv \text{Person} \sqcap \exists \text{isRegisteredAt.University}$
- $\text{Father} \sqsubseteq \text{Person}$  (what about  $\text{Father} \equiv \text{Person}$ ?)
- $\text{Father} \sqsubseteq \exists \text{hasChild.T}$  (what about  $\text{Father} \equiv \exists \text{hasChild.T}$ ?)

## *ALC* concept inclusions and TBoxes

More generally, let  $C$  and  $D$  be any *ALC* concepts. Then

- $C \sqsubseteq D$  is called an *ALC concept inclusion*. It states that every  $C$  is-a  $D$   
( $C$  is subsumed by  $D$ , or  $D$  subsumes  $C$ , or  $C$  is included in  $D$ )
- $C \equiv D$  is called an *ALC concept equation* ( $C$  and  $D$  are equivalent)  
can be regarded as an abbreviation for two inclusions  $C \sqsubseteq D$  and  $D \sqsubseteq C$

### Examples

- Disease  $\sqcap$   $\exists$ hasLocation.Heart  $\sqsubseteq$  NeedsTreatment
- $\exists$ studentOf.Computer Science  $\sqsubseteq$  HumanBeing  $\sqcap$   
 $\exists$ knows.ProgrammingLanguage

An *ALC TBox* is a finite set  $\mathcal{T}$  of *ALC* concept inclusions and equations

## OWL as DL: Classes

SubClassOf( $C$   $D$ )

$C \sqsubseteq D$

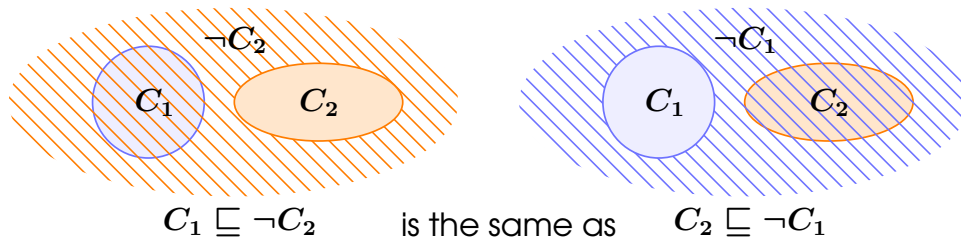
EquivalentClasses( $C_1$   $C_2$  ...  $C_n$ )

$C_1 \equiv C_2, C_2 \equiv C_3, \dots,$   
 $C_{n-1} \equiv C_n$

DisjointClasses( $C_1$   $C_2$  ...  $C_n$ )

$C_1 \sqsubseteq \neg C_2, C_1 \sqsubseteq \neg C_3, \dots, C_1 \sqsubseteq \neg C_n$   
 $C_2 \sqsubseteq \neg C_3, \dots, C_2 \sqsubseteq \neg C_n$   
 $\dots$   
 $C_{n-1} \sqsubseteq \neg C_n$

**Example:**  $C_1$  and  $C_2$  are **disjoint**:



## Concept hierarchies

The **concept hierarchy** induced by an  $\mathcal{ALC}$  TBox  $\mathcal{T}$  is defined as

$$\{A \sqsubseteq B \mid A, B \text{ are concept names in } \mathcal{T} \text{ s.t. } \mathcal{T} \text{ implies } A \sqsubseteq B\}$$

More generally, we are interested in the following **subsumption problem**:

- Given an  $\mathcal{ALC}$  TBox  $\mathcal{T}$  and  $\mathcal{ALC}$  concepts  $C$  and  $D$ , how can we decide whether  $\mathcal{T}$  **implies** that  $C \sqsubseteq D$

**Problem:** we do not yet have a precise definition of what it means that

$$\mathcal{T} \text{ implies } C \sqsubseteq D$$

so we do not have a precise definition of the concept hierarchy induced by a TBox

## Subsumption example: 1

Let  $\mathcal{T}$  be the following  $\mathcal{ALC}$  TBox:

Parent $\equiv$ Person $\sqcap$ $\exists$ hasChild.Person
Woman $\equiv$ Person $\sqcap$ Female
Mother $\equiv$ Parent $\sqcap$ Female

**Question 1:** Does  $\mathcal{T}$  imply that Mother  $\sqsubseteq$  Woman

**Answer:** Suppose  $x$  is any Mother.

By the third equation,  $x$  is both a Parent and a Female.

By the first equation,  $x$  is a Person.

Thus,  $x$  is both a Person and a Female.

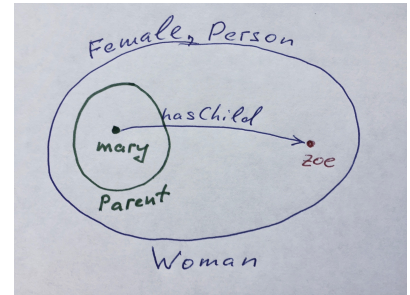
Now, the second equation states that  $x$  must be a Woman.

**What are the 'inference rules' for checking subsumption?**

## Subsumption example: 2

Let  $\mathcal{T}$  be the following *ALC* TBox:

Parent  $\equiv$  Person  $\sqcap \exists \text{hasChild}.\text{Person}$   
Woman  $\equiv$  Person  $\sqcap$  Female  
Mother  $\equiv$  Parent  $\sqcap$  Female



**Question 2:** Does  $\mathcal{T}$  imply that Woman  $\sqsubseteq$  Mother

**Answer:** Imagine a 'situation' or a 'world' with two individuals:

**mary**, who is a Female, a Person and hasChild **zoe**, and

**zoe**, who is a Female and a Person, but does not have a child

Then **mary** must also be a Parent, Woman and Mother

Also, **zoe** must be a Woman

However, in this world, **zoe** is neither a Parent nor a Mother

This **counterexample** shows that  $\mathcal{T}$  does not imply Woman  $\sqsubseteq$  Mother

(the counterexample is illustrated by the picture above. BTW, can you simplify it?)

**Is there a systematic way of constructing such counterexamples?**



## ALC semantics

Every **interpretation**  $\mathcal{I}$  consists of

a **domain** of interpretation, denoted  $\Delta^{\mathcal{I}}$ , which is just a non-empty set, and

an **interpretation function** that

- interprets any concept name  $A$  by a subset  $A^{\mathcal{I}}$  of  $\Delta^{\mathcal{I}}$   
the **interpretation of  $A$**  in  $\mathcal{I}$
- interprets any role name  $R$  by a binary relation  $R^{\mathcal{I}}$  over  $\Delta^{\mathcal{I}}$   
(thus,  $R^{\mathcal{I}}$  is some set of pairs from  $\Delta^{\mathcal{I}}$ )  
the **interpretation of  $R$**  in  $\mathcal{I}$
- interprets every individual name  $a$  by an element  $a^{\mathcal{I}}$  of  $\Delta^{\mathcal{I}}$   
the **interpretation of  $a$**  in  $\mathcal{I}$

## *ALC* semantics (cont.)

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

( $C, D$  are concepts and  $R$  a role name)

- $(\top)^{\mathcal{I}}$  is the whole domain  $\Delta^{\mathcal{I}}$  and  $(\perp)^{\mathcal{I}}$  is empty ( $= \emptyset$ )
- $(\neg C)^{\mathcal{I}}$  is the complement of  $C^{\mathcal{I}}$  in  $\Delta^{\mathcal{I}}$  (those elements of  $\Delta^{\mathcal{I}}$  that are not in  $C^{\mathcal{I}}$ )  
notation:  $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $(C \sqcap D)^{\mathcal{I}}$  is the intersection of  $C^{\mathcal{I}}$  and  $D^{\mathcal{I}}$  notation  $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $(C \sqcup D)^{\mathcal{I}}$  is the union of  $C^{\mathcal{I}}$  and  $D^{\mathcal{I}}$  notation  $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(\exists R.C)^{\mathcal{I}}$  is the set of all objects  $x$  in the domain  $\Delta^{\mathcal{I}}$  such that

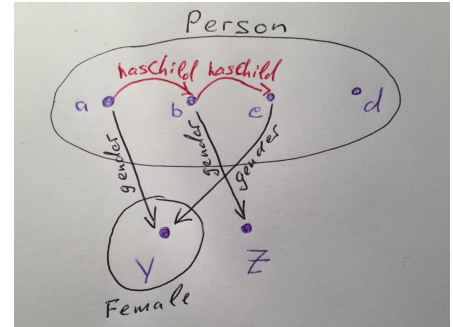
$x \xrightarrow[R]{\quad} y$  for some object  $y$  that belongs to  $C^{\mathcal{I}}$

- $(\forall R.C)^{\mathcal{I}}$  is the set of all objects  $x$  in the domain  $\Delta^{\mathcal{I}}$  such that
  - either  $x$  does not have  $R$ -successors
  - or whenever  $x \xrightarrow[R]{\quad} y$  then  $y$  belongs to  $C^{\mathcal{I}}$  (for all objects  $y$ )

## Example 1

Let  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ , where

- $\Delta^{\mathcal{I}} = \{a, b, c, d, Y, Z\}$
- $\text{Person}^{\mathcal{I}} = \{a, b, c, d\}$
- $\text{Female}^{\mathcal{I}} = \{Y\}$
- $\text{hasChild}^{\mathcal{I}} = \{(a, b), (b, c)\}$
- $\text{gender}^{\mathcal{I}} = \{(a, Y), (b, Z), (c, Y)\}$



**Exercise:** compute the following

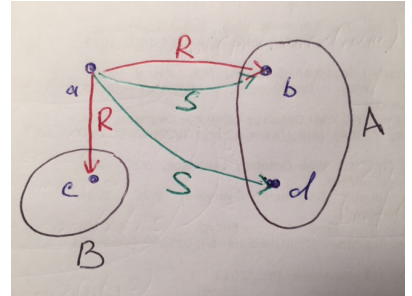
- $(\text{Person} \sqcap \exists \text{gender} . \top)^{\mathcal{I}} = \{a, b, c\}$  as  $\text{Person}^{\mathcal{I}} = \{a, b, c, d\}$ ,  $(\exists \text{gender} . \top)^{\mathcal{I}} = \{a, b, c\}$
- $(\text{Person} \sqcap \exists \text{gender} . \text{Female})^{\mathcal{I}} = \{a, c\}$  as  $(\exists \text{gender} . \text{Female})^{\mathcal{I}} = \{a, c\}$
- $(\text{Person} \sqcap \exists \text{hasChild} . \text{Person})^{\mathcal{I}} = \{a, b\}$  as  $(\exists \text{hasChild} . \text{Person})^{\mathcal{I}} = \{a, b\}$
- $(\text{Person} \sqcap \exists \text{hasChild} . \exists \text{gender} . \text{Female})^{\mathcal{I}} = \{b\}$
- $(\text{Person} \sqcap \exists \text{hasChild} . \exists \text{hasChild} . \top)^{\mathcal{I}} = \{a\}$

## Example 2

Consider the interpretation  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ , where

- $\Delta^{\mathcal{I}} = \{a, b, c, d\}$
- $A^{\mathcal{I}} = \{b, d\}$ ,  $B^{\mathcal{I}} = \{c\}$
- $R^{\mathcal{I}} = \{(a, b), (a, c)\}$ ,  $S^{\mathcal{I}} = \{(a, b), (a, d)\}$

Then we have:



- $(\forall R.A)^{\mathcal{I}} = \{b, c, d\}$ ,  $(\forall S.A)^{\mathcal{I}} = \{a, b, c, d\}$   
 $(\forall R.A)^{\mathcal{I}}$  is the set of objects  $x$  in  $\Delta^{\mathcal{I}}$  such that (i) either  $x$  has no outgoing  $R$ -arrow at all, or (ii) there are such arrows and their ends all belong to  $A^{\mathcal{I}}$
- $(\exists R.A \sqcap \forall R.A)^{\mathcal{I}} = \emptyset$ ,  $(\exists S.A \sqcap \forall S.A)^{\mathcal{I}} = \{a\}$
- $(\exists R.B \sqcap \exists R.A)^{\mathcal{I}} = \{a\}$ ,  $(\exists R.(A \sqcap B))^{\mathcal{I}} = \emptyset$
- $(\forall R.\neg A)^{\mathcal{I}} = \{b, c, d\}$ ,  $(\forall S.\neg A)^{\mathcal{I}} = \{b, c, d\}$

## Examples of equivalent concepts (classes)

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

- $(\neg\neg C)^{\mathcal{I}} = C^{\mathcal{I}}$
- $(\forall R.C)^{\mathcal{I}} = (\neg\exists R.\neg C)^{\mathcal{I}}$
- $(\neg(C \sqcap D))^{\mathcal{I}} = (\neg C \sqcup \neg D)^{\mathcal{I}}$
- $(\neg(C \sqcup D))^{\mathcal{I}} = (\neg C \sqcap \neg D)^{\mathcal{I}}$
- $(\neg\exists R.C)^{\mathcal{I}} = (\forall R.\neg C)^{\mathcal{I}}$
- $(\neg\forall R.C)^{\mathcal{I}} = (\exists R.\neg C)^{\mathcal{I}}$
- $(C \sqcap \neg C)^{\mathcal{I}} = \perp^{\mathcal{I}} = \emptyset$
- $(C \sqcup \neg C)^{\mathcal{I}} = \top^{\mathcal{I}} = \Delta^{\mathcal{I}}$

## Semantics: when is a concept inclusion true in an interpretation?

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

- We write  $\mathcal{I} \models C \sqsubseteq D$  if  $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$  If this is the case, we say that
  - $\mathcal{I}$  **satisfies**  $C \sqsubseteq D$  or, equivalently,
  - $C \sqsubseteq D$  is **true** in  $\mathcal{I}$  or, equivalently,
  - $\mathcal{I}$  is a **model** of  $C \sqsubseteq D$ .
- We write  $\mathcal{I} \models C \equiv D$  if  $C^{\mathcal{I}} = D^{\mathcal{I}}$
- We write  $\mathcal{I} \models \mathcal{T}$  if  $\mathcal{I} \models C \sqsubseteq D$  for all  $C \sqsubseteq D$  in  $\mathcal{T}$   
If this is the case, then we say that
  - $\mathcal{I}$  **satisfies**  $\mathcal{T}$  or, equivalently,
  - $\mathcal{I}$  is a **model** of  $\mathcal{T}$ .

## Semantics: when does a concept inclusion follow from a TBox?

Let  $\mathcal{T}$  be a TBox and  $C \sqsubseteq D$  a concept inclusion.

$C \sqsubseteq D$  follows from  $\mathcal{T}$  if every model of  $\mathcal{T}$  is also a model of  $C \sqsubseteq D$

Instead of saying that  $C \sqsubseteq D$  follows from  $\mathcal{T}$  we often write

- $\mathcal{T} \models C \sqsubseteq D$  or  $C \sqsubseteq_{\mathcal{T}} D$

**Example:** let **MED** be the **ALC** TBox with the following inclusions (SNOMED CT)

Pericardium  $\sqsubseteq$  Tissue  $\sqcap$   $\exists$ contIn.Heart

Pericarditis  $\sqsubseteq$  Inflammation  $\sqcap$   $\exists$ hasLoc.Pericardium

Inflammation  $\sqsubseteq$  Disease  $\sqcap$   $\exists$ actsOn.Tissue

Disease  $\sqcap$   $\exists$ hasLoc. $\exists$ contIn.Heart  $\sqsubseteq$  Heartdisease  $\sqcap$  NeedsTreatment

Pericarditis needs treatment if and only if  $\text{Pericarditis} \sqsubseteq_{\text{MED}} \text{NeedsTreatment}$

## Examples

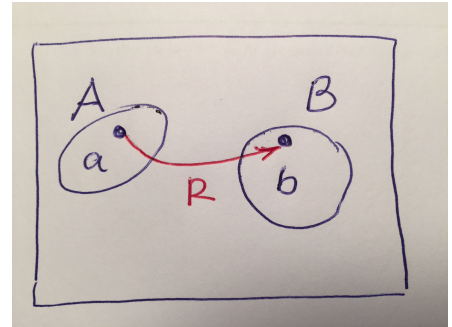
Let  $\mathcal{T} = \{A \subseteq \exists R.B\}$  Then  $\mathcal{T} \not\models A \subseteq B$

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

- $\mathcal{I} \models \mathcal{T}$
- $\mathcal{I} \not\models A \subseteq B$

Namely, define  $\mathcal{I}$  by taking

- $\Delta^{\mathcal{I}} = \{a, b\}$
- $A^{\mathcal{I}} = \{a\}$
- $R^{\mathcal{I}} = \{(a, b)\}$
- $B^{\mathcal{I}} = \{b\}$



Then  $A^{\mathcal{I}} = \{a\} \subseteq \{a\} = (\exists R.B)^{\mathcal{I}}$ , and so  $\mathcal{I} \models \mathcal{T}$

But  $\{a\} = A^{\mathcal{I}} \not\subseteq B^{\mathcal{I}} = \{b\}$ , and so  $\mathcal{I} \not\models A \subseteq B$



## Examples

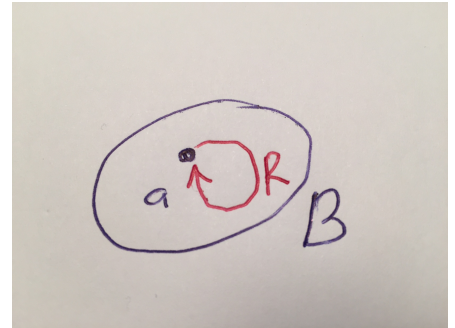
Let again  $\mathcal{T} = \{A \sqsubseteq \exists R.B\}$  Then  $\mathcal{T} \not\models \exists R.B \sqsubseteq A$

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

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

Define  $\mathcal{I}$  by taking

- $\Delta^{\mathcal{I}} = \{a\}$
- $A^{\mathcal{I}} = \emptyset$  (empty set)
- $R^{\mathcal{I}} = \{(a, a)\}$
- $B^{\mathcal{I}} = \{a\}$



Then  $A^{\mathcal{I}} = \emptyset \subseteq \{a\} = (\exists R.B)^{\mathcal{I}}$ , and so  $\mathcal{I} \models \mathcal{T}$

But  $(\exists R.B)^{\mathcal{I}} = \{a\} \not\subseteq \emptyset = A^{\mathcal{I}}$ , and so  $\mathcal{I} \not\models \exists R.B \sqsubseteq A$

## Example

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

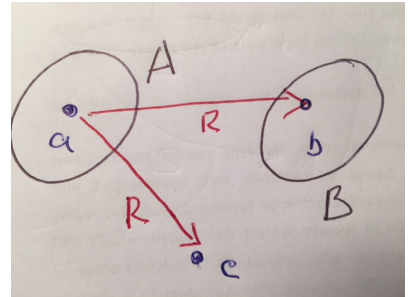
$$\mathcal{T} \not\models A \sqsubseteq \forall R.B$$

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

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

Define  $\mathcal{I}$  by taking

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



Then  $A^{\mathcal{I}} = \{a\} \subseteq \{a\} = (\exists R.B)^{\mathcal{I}}$ , and so  $\mathcal{I} \models \mathcal{T}$ .

But  $A^{\mathcal{I}} \not\subseteq (\forall R.B)^{\mathcal{I}} = \{b, c\}$ , 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

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

Define  $\mathcal{I}$  by taking

- $\Delta^{\mathcal{I}} = \{a\}$
- $A^{\mathcal{I}} = \{a\}$
- $R^{\mathcal{I}} = \emptyset$
- $B^{\mathcal{I}} = \emptyset$

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  $\mathcal{I} \not\models A \sqsubseteq \exists R.B$

## Reasoning with $\mathcal{ALC}$ (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  $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$  for all interpretations  $\mathcal{I}$ . In this case, we write  $\emptyset \models C \sqsubseteq D$ .
- **Concept satisfiability.** A concept  $C$  is called satisfiable 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 an  $\mathcal{ALC}$ -concept  $C$  is satisfiable.

## Concept satisfiability: example 1

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

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

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

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

## Concept satisfiability: 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) | $(x, y): \text{hasChild}$ and $y: \text{Male}$ , for fresh $y$          |

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

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

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

## Concept satisfiability: 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 Method

How can we show satisfiability of a concept?

Achieved by applying the **tableau method**

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

Proof procedure:

- 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
- a **clash** occurs when the rules produce both  $A$  and  $\neg A$ ,  
for some concept name  $A$
- 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 **ALC**-concept can be transformed into an equivalent one in NNF

using the following rules:

$\neg \top$	$\equiv$	$\perp$	
$\neg \perp$	$\equiv$	$\top$	
$\neg \neg C$	$\equiv$	$C$	
$\neg(C \sqcap D)$	$\equiv$	$\neg C \sqcup \neg D$	(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.

$$\begin{aligned} \neg\exists R.(A \sqcap \neg B) \sqcup \neg\forall R.(\neg A \sqcup \neg B) &\equiv && (\text{use } \neg\exists R.D \equiv \forall R.\neg D) \\ \forall R.\neg(A \sqcap \neg B) \sqcup \neg\forall R.(\neg A \sqcup \neg B) &\equiv && (\text{use } \neg(A \sqcap D) \equiv \neg A \sqcup \neg D) \\ \forall R.(\neg A \sqcup \neg\neg B) \sqcup \neg\forall R.(\neg A \sqcup \neg B) &\equiv && (\text{use } \neg\neg B \equiv B) \\ \forall R.(\neg A \sqcup B) \sqcup \neg\forall R.(\neg A \sqcup \neg B) &\equiv && (\text{use } \neg\forall R.D \equiv \exists R.\neg D) \\ \forall R.(\neg A \sqcup B) \sqcup \exists R.\neg(\neg A \sqcup \neg B) &\equiv && (\text{use } \neg(C \sqcup D) \equiv \neg C \sqcap \neg D) \\ \forall R.(\neg A \sqcup B) \sqcup \exists R.(\neg\neg A \sqcap \neg\neg B) &\equiv && (\text{use } \neg\neg C \equiv C) \\ \forall R.(\neg A \sqcup B) \sqcup \exists R.(A \sqcap B) &&& \end{aligned}$$

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

To determine whether a given concept  $C$  in NNF is satisfiable,

**Step 1** take the **constraint**  $x : C$  (saying that object  $x$  is an element of  $C$ )

**Step 2** apply to the constructed constraints the **completion rules** given below, generating one or more **constraint systems**

**Clash** a constraint system has a clash if it contains both constraints of the form  $x : A$  and  $x : \neg A$ , for some  $x$  and some concept name  $A$

**Step 3** if every constraint system contains a clash, then  $C$  is **not satisfiable**

**Step 4** if we have managed to construct a clash-free constraint system to which no completion rule is applicable then we can extract from it a **model satisfying**  $C$

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

**Rule  $\rightarrow_{\sqcap}$**  If (i) the current constraint system  $S$  contains  $x: C \sqcap D$  and (ii) does not contain at least one of  $x: C$  and  $x: D$ ,  
then add  $x: C$  and  $x: D$  to  $S$

if  $S$  contains both  $x: C$  and  $x: D$ , the rule is not applicable

**Rule  $\rightarrow_{\sqcup}$**  If (i) the current constraint system  $S$  contains  $x: C \sqcup D$  and (ii) does not contain both  $x: C$  and  $x: D$ ,  
then construct **two** alternative constraint systems  $S_1$  and  $S_2$   
 $S_1$  extends  $S$  with  $x: C$  and  $S_2$  extends  $S$  with  $x: D$

if  $S$  contains at least one of  $x: C$  and  $x: D$ , the rule is not applicable

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

**Rule  $\rightarrow_{\forall}$**  If (i) the current constraint system  $S$  contains  $x: \forall R.C$  and (ii) also contains  $(x, y): R$  and (iii) does not contain  $y: C$ , then add  $y: C$  to  $S$

if conditions (i)–(iii) are not satisfied, the rule is not applicable

**Rule  $\rightarrow_{\exists}$**  If (i) the current constraint system  $S$  contains  $x: \exists R.C$  and (ii) does not contain both  $(x, z): R$  and  $z: C$ , for any  $z$  then take a fresh object  $y$  and add  $(x, y): R$  and  $y: C$  to  $S$

if conditions (i) and (ii) are not satisfied, the rule is not applicable

**NB:**  $\rightarrow_{\exists}$  is 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 apply the tableau algorithm to the constraint system

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

The only rule applicable is  $\rightarrow_{\sqcup}$ . We have two possibilities. First, we can try

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

Then we can apply  $\rightarrow_{\sqcap}$  and add to  $S_1$  the constraints

$$x : A \quad \text{and} \quad x : \neg A$$

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

$$S_2 = S \cup \{x : B\}$$

No rule is applicable to  $S_2$ , it does not contain a clash, and so

$$(A \sqcap \neg A) \sqcup B \text{ is satisfiable}$$

A model  $\mathcal{I}$  satisfying it is given by  $\Delta^{\mathcal{I}} = \{x\}$ ,  $B^{\mathcal{I}} = \{x\}$ ,  $A^{\mathcal{I}} = \emptyset$ .

## Tableau Example 2

**Q:** is  $C = A \sqcap \exists R. \exists Q. B \sqcap \forall R. \neg B$  is satisfiable? It is in NNF, so we start with

$$S = \{x : A \sqcap \exists R. \exists Q. B \sqcap \forall R. \neg B\}$$

An application of  $\rightarrow_{\sqcap}$  gives add to  $S$  the constraints

$$x : A \quad \text{and} \quad x : \exists R. \exists Q. B \sqcap \forall R. \neg B$$

An application of  $\rightarrow_{\sqcap}$  further adds

$$x : \exists R. \exists Q. B \quad \text{and} \quad x : \forall R. \neg B$$

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

$$(x, y) : R \quad \text{and} \quad y : \exists Q. B$$

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

$$(y, z) : Q \quad \text{and} \quad z : B$$

An application of  $\rightarrow_{\forall}$  now adds

$$y : \neg B$$

No rule is applicable and there is no clash. So  $C$  is satisfiable. A model  $\mathcal{I}$  of  $C$ :

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

## Tableau Example 3

**Q:** is  $C = \exists R.A \sqcap \exists R.\neg A$  satisfiable?  $C$  is in NNF, so we start with

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

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

$$x : \exists R.A \quad \text{and} \quad x : \exists R.\neg A$$

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

$$(x, y) : R \quad \text{and} \quad y : A$$

Another application of  $\rightarrow_{\exists}$  adds

$$(x, z) : R \quad \text{and} \quad z : \neg A$$

No rule is applicable now and there is 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 construction always returns a correct result,  
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 to it, then the concept is satisfiable

One also has to identify the **computational complexity** of the tableau algorithm

## Tableau Calculus: Soundness

- Suppose that a constraint system  $S$  is satisfiable and

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

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

$$S_0, S_1, S_2, \dots$$

such that  $S_{i+1}$  is obtained from  $S_i$   
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.,  $C^{\mathcal{I}} \neq \emptyset$

## Complexity

Generating binary trees:

$$D_1 \equiv \exists R.C_1 \sqcap \exists R.C_2$$

$$D_2 \equiv (\exists R.C_1 \sqcap \exists R.C_2) \sqcap \forall R.(\exists R.C_1 \sqcap \exists R.C_2)$$

$$D_3 \equiv (\exists R.C_1 \sqcap \exists R.C_2) \sqcap \forall R.((\exists R.C_1 \sqcap \exists R.C_2) \sqcap \forall R.(\exists R.C_1 \sqcap \exists R.C_2))$$

...

The tableau algorithm constructs a model satisfying  $D_n$ ,

which is a **binary tree** of depth  $n$  (with  $2^n$  leaves)

In the worst case, the tableau algorithm requires **exponential time**

(i.e., is not tractable)

however, optimised reasoners work well for most **real-world** ontologies

## Reasoning Services for $\mathcal{ALC}$ with TBoxes

- **Subsumption w.r.t. TBoxes** We say that a concept inclusion  $C \sqsubseteq D$  **follows from** a TBox  $\mathcal{T}$  if **every** a model of  $\mathcal{T}$  is also a model of  $C \sqsubseteq D$ .

In this case, we write  $\mathcal{T} \models C \sqsubseteq D$

- **Concept satisfiability w.r.t. TBoxes** A concept  $C$  is **satisfiable** w.r.t. a TBox  $\mathcal{T}$  if there exists a model  $\mathcal{I}$  of  $\mathcal{T}$  such that  $C^{\mathcal{I}} \neq \emptyset$
- **TBox satisfiability** A TBox  $\mathcal{T}$  is satisfiable if there exists a model of  $\mathcal{T}$

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

- $\mathcal{T} \models C \sqsubseteq D$  if, and only if,  $C \sqcap \neg D$  is **not** satisfiable w.r.t.  $\mathcal{T}$
- $\mathcal{T}$  is satisfiable if, and only if,  $A$  is satisfiable w.r.t.  $\mathcal{T}$ ,

where  $A$  is a fresh concept name

Thus, it suffices to design an algorithm checking concept satisfiability w.r.t. 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  $D$ ,

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

$C \sqsubseteq D$  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:

$$y : D \quad (x, y) : R \quad \top \sqsubseteq D$$

## Reasoning with TBoxes (cont.)

**Rule  $\rightarrow_U$**  If (i) the current constraint system  $S$  contains  $T \sqsubseteq D$  and  
(ii) also an occurrence of  $x$  and  
(iii) does not contain  $x : D$ , then add  $x : D$  to  $S$

The tableau algorithm based on rules

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

**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

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

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*)

# Reasoning with ABoxes

# The structure of knowledge bases

## Knowledge Base (KB)

**TBox** (terminological box, schema)

$\text{Man} \equiv \text{Human} \sqcap \text{Male}$   
 $\text{HappyFather} \equiv \text{Man} \sqcap \exists \text{hasChild}$   
...

**ABox** (assertion box, data)

$\text{john} : \text{Man}$   
 $(\text{john}, \text{mary}) : \text{hasChild}$   
...

**Inference System**

**Interface**

## ABoxes

To represent (incomplete) knowledge about concrete objects,  
the language of description logic contains

- **individual** (or **object**) **names**  $a_0, a_1, \dots$  (e.g., john, mary, ...)

An ABox,  $\mathcal{A}$ , is a set of **assertions**

- $a : C$                     `a is an instance of C'
- $(a, b) : R$                 `a is R-related to b'

Every interpretation  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  specifies a value  $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ ,  
for every individual name  $a$

an interpretation  $\mathcal{I}$  **satisfies** (models) an assertion

- $\mathcal{I} \models a : C$     iff    $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- $\mathcal{I} \models (a, b) : R$     iff    $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

An interpretation  $\mathcal{I}$  is a **model** of a TBox  $\mathcal{T}$  and an ABox  $\mathcal{A}$     iff  
 $\mathcal{I}$  satisfies **every axiom** of  $\mathcal{T}$  and  $\mathcal{A}$

## ABox Inference Services

- **ABox consistency:**  
is the collection of assertions  $\mathcal{A}$  satisfiable (w.r.t. a TBox  $\mathcal{T}$ )?  
 $\mathcal{A}$  is consistent w.r.t. a TBox  $\mathcal{T}$  iff there exists **some model**  $\mathcal{I}$  of  $\mathcal{T}$  and  $\mathcal{A}$
- **instance checking:**  
is  $a$  an instance of a concept  $C$ ?  
 $a$  is an instance of  $C$  w.r.t. a TBox  $\mathcal{T}$  and an ABox  $\mathcal{A}$  iff  
 $a^{\mathcal{I}} \in C^{\mathcal{I}}$ , **for every model**  $\mathcal{I}$  of  $\mathcal{T}$  and  $\mathcal{A}$
- **ABox realisation:**  
for all individuals in  $\mathcal{A}$ ,  
compute their **most specific concept names** w.r.t.  $\mathcal{T}$

## ABox and TBox Inference Services based on Consistency

All inference services can be reduced to ABox consistency:

- **instance checking**

$a$  is an instance of  $C$  (w.r.t. a TBox  $\mathcal{T}$  and an ABox  $\mathcal{A}$ ) iff  
 $\mathcal{A} \cup \{a: \neg C\}$  is **inconsistent** (w.r.t.  $\mathcal{T}$ )

- **concept satisfiability**

$C$  is satisfiable w.r.t. a TBox  $\mathcal{T}$  iff  $\{a: C\}$  is **consistent** w.r.t.  $\mathcal{T}$   
( $a$  does not occur in  $\mathcal{T}$ )

- **concept subsumption**

$C$  is subsumed by  $D$  w.r.t. a TBox  $\mathcal{T}$  iff  
 $\{a: C \sqcap \neg D\}$  is **inconsistent** w.r.t.  $\mathcal{T}$   
( $a$  does not occur in  $\mathcal{T}$ )

## ABox Inference Services: Andrea example

Consider the ABox  $\mathcal{A}$ :

1. (john, susan) : friend
2. (john, andrea) : friend
3. (susan, andrea) : loves
4. (andrea, bill) : loves
5. susan : Female
6. bill :  $\neg$ Female

Represent the following query (to  $\mathcal{A}$ ) as an inference service problem and find an answer:

Does John have a female friend who is in love with a male (not female) person?

john :  $\exists$ friend.(Female  $\sqcap$   $\exists$ loves. $\neg$ Female)

## Andrea Example

To this end we check whether

$$\mathcal{A} \cup \{\text{john} : \neg \exists \text{friend}.(\text{Female} \sqcap \exists \text{loves}.\neg \text{Female})\}$$

has a model. If not, then the answer to the query

$$\text{john} : \exists \text{friend}.(\text{Female} \sqcap \exists \text{loves}.\neg \text{Female})$$

is YES.

Transformation into negation normal form gives:

$$\text{john} : \forall \text{friend}.(\neg \text{Female} \sqcup \forall \text{loves}.\text{Female})$$



## Andrea Example (cont.)

Thus, we apply the tableau to the constraint system

$$\mathcal{A} \cup \{\text{john} : \forall \text{friend} . (\neg \text{Female} \sqcup \forall \text{loves} . \text{Female})\}$$

given by

1. (john, susan) : friend
2. (john, andrea) : friend
3. (susan, andrea) : loves
4. (andrea, bill) : loves
5. susan : Female
6. bill :  $\neg$ Female
7. john :  $\forall \text{friend} . (\neg \text{Female} \sqcup \forall \text{loves} . \text{Female})$

## Andrea Example (cont.)

Two applications of the rule  $\rightarrow_{\forall}$  give the additional constraints:

$$\text{susan} : (\neg\text{Female} \sqcup \forall\text{loves.Female})$$

and

$$\text{andrea} : (\neg\text{Female} \sqcup \forall\text{loves.Female})$$

We now apply the rule  $\rightarrow_{\sqcup}$  to the first constraint:

- Adding the constraint  $\text{susan} : \neg\text{Female}$  results in a clash since we have already  $\text{susan} : \text{Female} \in \mathcal{A}$ .
- Thus we add the constraint  $\text{susan} : \forall\text{loves.Female}$  to the constraint system.

## Andrea Example (cont.)

We now apply  $\rightarrow_{\forall}$  to

$susan : \forall \text{loves.Female}, \quad (susan, andrea) : \text{loves}$

and add

$andrea : \text{Female}$

to the constraint system.

We apply  $\rightarrow_{\sqcup}$  to

$andrea : (\neg \text{Female} \sqcup \forall \text{loves.Female})$

- Adding  $andrea : \neg \text{Female}$  to the constraint systems results in a clash since  $andrea : \text{Female}$  is in the constraint system.
- Thus we add the constraint  $andrea : \forall \text{loves.Female}$  to the constraint system.

## Andrea Example (cont.)

Now we apply  $\rightarrow_{\forall}$  to

andrea :  $\forall \text{loves.Female}$ , (andrea, bill) : loves

and add

bill : Female

to the constraint system. But this results in a clash since bill :  $\neg\text{Female}$  is already in the constraint system.

It follows that every sequence of completion rule application results in a clash.