

OWL as a Description Logic

OWL 2 DL is very close to the ***SR*OIQ(D)** Description Logic

- **S** = **ALC**_{R+} — **ALC** with transitive roles
TBox axioms of the form *transitive*(*R*)
- **R** — **R**ole inclusions with chains
TBox axioms of the form $R_1 \circ \dots \circ R_n \sqsubseteq S$, where R_1, \dots, R_n, S are roles
- **O** — **n**Ominals
concept constructor $\{a\}$, where a is an individual name
- **I** — **I**nverse roles
role constructor R^- where R is a role name
- **N** — **Q**ualified number restrictions
concept constructors $\leq n R.C$ and $\geq n R.C$
- **(D)** — **D**atatypes (datatype properties)

OWL 1 DL is very close to the ***SH*OIN(D)** Description Logic

- **H** — **r**ole **H**ierarchies
TBox axioms of the form $R \sqsubseteq S$, where R, S are roles
- **N** — simple **N**umber restrictions
concept constructors $\leq n R$ and $\geq n R$

OWL as DL: Class Constructors

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

OWL as DL: Classes

SubClassOf($C \ D$)

$C \sqsubseteq D$

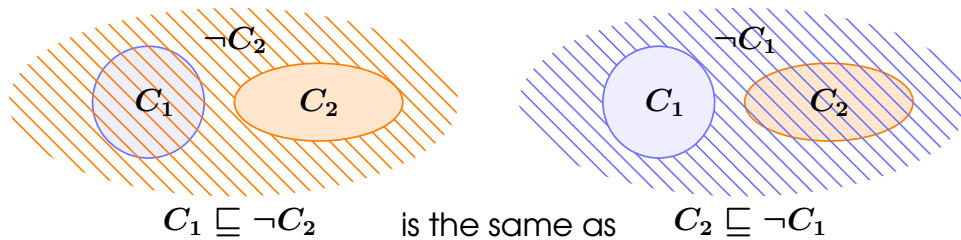
EquivalentClasses($C_1 \ C_2 \ \dots \ C_n$)

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

DisjointClasses($C_1 \ C_2 \ \dots \ C_n$)

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

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



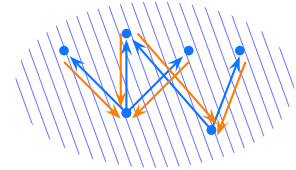
OWL as DL: Object Properties

SubObjectPropertyOf(R S)

EquivalentObjectProperties(R S)

$R \sqsubseteq S$ property R and its inverse R^-

$R \equiv S$



InverseObjectProperties(R S)

TransitiveObjectProperty(R)

FunctionalObjectProperty(R)

InverseFunctionalObjectProperty(R)

SymmetricObjectProperty(R)

ObjectPropertyRange(R C)

ObjectPropertyDomain(R D)

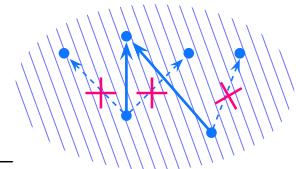
$R \equiv S^-$

R is functional

$R \circ R \sqsubseteq R$

$\top \sqsubseteq \leq 1 R$

$\top \sqsubseteq \leq 1 R^-$

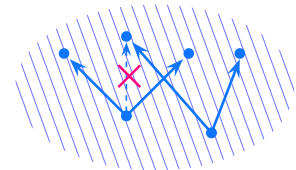


$R^- \sqsubseteq R$

R is inverse functional

$\top \sqsubseteq \forall R.C$

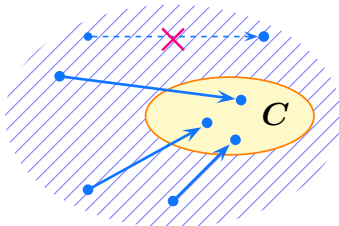
$\exists R.\top \sqsubseteq D$



OWL as DL: Domain and Range Constraints

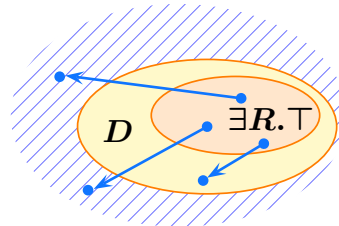
ObjectPropertyRange(R C)

$$\top \sqsubseteq \forall R.C$$



ObjectPropertyDomain(R D)

$$\exists R.\top \sqsubseteq D$$



NB: another way of representing
the range constraint:

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

the domain constraint:

$$\top \sqsubseteq \forall R^{-}.D$$

DL Knowledge Bases (Ontologies)

knowledge base KB = TBox \mathcal{T} + ABox \mathcal{A}

an interpretation \mathcal{I} is a **model** of a knowledge base KB ($\mathcal{I} \models KB$) if
 \mathcal{I} satisfies **every** axiom of \mathcal{T} and **every** assertion of \mathcal{A}

- $a: C$ is a **logical consequence** of a knowledge base KB if,
for every interpretation \mathcal{I} , $\mathcal{I} \models KB$ implies $\mathcal{I} \models a: C$
(in other words, if a is an instance of C w.r.t. \mathcal{T} and \mathcal{A})
- $(a, b): R$ is a **logical consequence** of a knowledge base KB if,
for every interpretation \mathcal{I} , $\mathcal{I} \models KB$ implies $\mathcal{I} \models (a, b): R$

From DL to OWL

Note that DL axioms can be represented in OWL in various ways. For instance,

$$A \sqsubseteq B \quad (A \text{ and } B \text{ are concept names})$$

corresponds to

1. **SubClassOf**($A \ B$)
2. **DisjointClasses**($A \ \text{ObjectComplementOf}(B)$)
3. **SubClassOf**($\text{owl:Thing} \ \text{ObjectUnionOf}(A \ \text{ObjectComplementOf}(B))$)

$$\top \sqsubseteq \leq 1 R \quad (R \text{ is a role name})$$

corresponds to

1. **FunctionalObjectProperty**(R)
2. **SubClassOf**($\text{owl:Thing} \ \text{ObjectMaxCardinality}(1 \ R)$)
3. **DisjointClasses**($\text{owl:Thing} \ \text{ObjectComplementOf}(\text{ObjectMaxCardinality}(1 \ R))$)

Brief summary of guidelines

- Always **paraphrase** a description or definition before encoding it in OWL
- Make all **primitives disjoint** — which requires that primitives form trees
- Use **ObjectSomeValuesFrom** as the default quantifier in restrictions
- Be careful to make **defined classes** defined (the default is primitive in Protégé).
The classifier will place nothing under a primitive class
(except in the presence of axioms/domain/range constraints)
- Remember the **open world assumption**.
Insert closure restrictions if that is what you mean
- Be careful with **domain** and **range** constraints.
Check them carefully if classification does not work as expected
- Be careful about the use of “and” and “or” (**intersection** and **union**)
- To spot **trivially satisfiable** restrictions early, always have an existential (**some**) restriction corresponding to every universal (**all**) restriction,
either in the class or in one of its superclasses
- Run the classifier **frequently**; spot errors early

Examples

(showing how to do the course work)

Travel service ontology

Sketch a normalised ontology for use by travel agency covering the following:

Hotel, restaurant, sports, luxury hotel, bed and breakfast, safari, activity, hiking, spa treatment, sunbathing, sightseeing, accommodation rating (three stars, etc.), campground, surfing.

Build a class hierarchy and indicate which classes in it are primitive and which are definable. Indicate the required roles, their properties, domains and ranges, as well as individuals.

Define the following classes using OWL abstract syntax:

1. A two star hotel.
2. A spa resort (i.e., a destination offering a spa treatment).
3. A destination with sport activities but without safari.
4. A destination where all hotels have three star rating.
5. A destinations with at least three restaurants and at least four hotels.

1. Card sorting

hotel

restaurant

sports

luxury hotel

bed & breakfast

two stars

sunbathing

sightseeing

accommodation rating

three stars

campground

swimming pool

hiking

activity

spa treatment

safari

surfing

destination

2. Arrange Concepts/Properties into Hierarchy

self-standing	modifiers	relations	definable
<ul style="list-style-type: none"> - accommodation - hotel <ul style="list-style-type: none"> - luxury hotel - b&b - campground - activity <ul style="list-style-type: none"> - sports <ul style="list-style-type: none"> - surfing - hiking - safari - sightseeing - relaxation <ul style="list-style-type: none"> - sunbathing - spa treatment - restaurant - destination 	accommodation rating <ul style="list-style-type: none"> - two stars - three stars - four stars - five stars hotel facility <ul style="list-style-type: none"> - swimming pool - meeting facilities 	hasAccommodation hasActivity hasRating hasRestaurant hasFacility	two star hotel ...

NB. All siblings in the hierarchy of self-standing entities are **disjoint**.

Class Hierarchy represents an “IS-A” Relation

a class *A* is a **subclass** of *B* if **every** instance of *A* is also an instance of *B*

- hotels and B&B are accommodation
therefore, *Hotel* and *B&B* may be regarded as subclasses of *Accommodation*
- hiking and surfing are sport activities
therefore, *Hiking* and *Surfing* may be regarded as subclasses of *Sports*
- **however**, neither a restaurant is a hotel nor a swimming pool is a hotel
therefore, neither *Restaurant* nor *SwimmingPool* can be a subclass of *Hotel*
(although a hotel may have a restaurant/swimming pool)

Do not confuse containment and a subclass relation!

Relations

```
ObjectPropertyDomain(hasAccommodation Destination)
ObjectPropertyRange(hasAccommodation Accommodation))
ObjectPropertyDomain(hasActivity Destination)
ObjectPropertyDomain(hasActivity Activity)
ObjectPropertyDomain(hasRestaurant ObjectUnionOf(Destination Hotel))
ObjectPropertyRange(hasRestaurant Restaurant)
ObjectPropertyDomain(hasFacility Hotel)
ObjectPropertyDomain(hasFacility HotelFacility)
```

Modifiers: using value sets

Consider modifier *AccommodationRating*
(one star, two stars, three stars, four stars, five stars)

```
ObjectPropertyDomain(hasRating Accommodation)
FunctionalObjectProperty(hasRating)
ObjectPropertyRange(hasRating AccommodationRating)
EquivalentClasses(AccommodationRating
                    ObjectOneOf(oneStar, twoStars, . . . , fiveStars))
DifferentIndividuals(oneStar, twoStars, . . . , fiveStars)
```

Specifying additional restrictions

'all luxury hotels have restaurants, swimming pools and meeting facilities'

```
SubClassOf(LuxuryHotel ObjectIntersectionOf(  
  ObjectSomeValueFrom(hasRestaurant owl:Thing)  
  ObjectSomeValueFrom(hasFacility SwimmingPool)  
  ObjectSomeValueFrom(hasFacility MeetingFacility)))
```

Q1: Definitions vs. Descriptions

A two star hotel (**definition**):

'a two star hotel is any hotel that has two star rating'

```
EquivalentClasses(TwoStarHotel ObjectIntersectionOf(Hotel  
ObjectHasValue(hasRating twoStars)))
```

Consider now the following **description**:

'all w-two-star-hotels have two star rating'

```
SubClassOf(W-TwoStarHotel ObjectIntersectionOf(Hotel  
ObjectHasValue(hasRating twoStars)))
```

Then

- *chelseaInn* is an **instance** of *TwoStarHotel* (necessary & sufficient cond.)
- but *chelsea_inn* **not** an **instance** of *W-TwoStarHotel* (necessary cond.)

w.r.t. the following ABox

```
chelsea_inn: Hotel, (chelsea_inn, twoStars): hasRating
```

Q2: Use *someValuesFrom* as the default quantifier in restrictions

Destinations with spa treatment:

'a spa resort is any destination that has a spa treatment'

```
EquivalentClasses(SpaResort ObjectIntersectionOf(  
  Destination  
  ObjectSomeValuesFrom(hasActivity SpaTreatment)))
```

Note that **any destination without activities** is an instance of the concept

```
EquivalentClasses(W-SpaResort ObjectIntersectionOf(  
  Destination  
  ObjectAllValuesFrom(hasActivity SpaTreatment)))
```

'only' does not imply 'some'!

Q3: Use *someValuesFrom* as the default quantifier (cont.)

A destination with sport activities but without safari:

'a destination with sport activities but without safari is
any destination that has some sport activity but does not have any safari'

```
EquivalentClasses(DestinationWithSportButNoSafari ObjectIntersectionOf(  
  Destination  
  ObjectSomeValuesFrom(hasActivity Sports)  
  ObjectComplementOf(ObjectSomeValuesFrom(hasActivity Safari))))
```

or, **equivalently**,

$\neg\exists R.C$ means $\forall R.\neg C$

'... is any destination that has some sport activity and
has only activities that are not safari activities'

```
EquivalentClasses(DestinationWithSportButNoSafari ObjectIntersectionOf(  
  Destination  
  ObjectSomeValuesFrom(hasActivity Sports)  
  ObjectAllValuesFrom(hasActivity ObjectComplementOf(Safari))))
```

NB. Safari and Sports are **disjoint**

Q4: Unions, Intersections and Complements

A destination where all hotels have three star rating.

‘a destination where all hotels have three star rating is
any destination that has only either not hotels or three star hotels’

```
EquivalentClasses(ThreeStarHotelDestination ObjectIntersectionOf(  
  Destination  
  ObjectAllValuesFrom(hasAccommodation ObjectUnionOf(  
    ObjectComplementOf(Hotel)  
    ObjectIntersectionOf(Hotel  
      ObjectHasValue(hasRating threeStars))))))
```

Alternatively,

‘a destination where all hotels have three star rating is
any destination that has no hotel that has not a three star rating’

```
EquivalentClasses(ThreeStarHotelDestination ObjectIntersectionOf(  
  Destination  
  ObjectComplementOf(ObjectSomeValuesFrom(hasAccommodation  
    ObjectIntersectionOf(Hotel  
      ObjectComplementOf(  
        ObjectHasValue(hasRating threeStars))))))
```

Q5: Number restrictions

A destinations with at least three restaurants and at least four hotels.

'a destination with at least three restaurants and at least four hotels is
any destination that has at least three restaurants
and has at least four **places to stay**'

```
EquivalentClasses(TRFHDestination ObjectIntersectionOf(  
  Destination  
  ObjcetMinCardinality(3 hasRestaurant)  
  ObjectMinCardinality(4 hasAccommodation)))
```

NB. OWL DL 1 has no **qualified** number restrictions (OWL 2 has them),
so in this example we have to slightly extend the definition
and include any types of accommodation (not only hotels).

Translating OWL into DL

1. DisjointClasses(Journal Book Newspaper)

$$\text{Journal} \sqsubseteq \neg \text{Book}, \quad \text{Journal} \sqsubseteq \neg \text{Newspaper}, \quad \text{Book} \sqsubseteq \neg \text{Newspaper}$$

2. InverseObjectProperties(isBorrowedBy borrows)

FunctionalObjectProperty(isBorrowedBy)

ObjectPropertyDomain(isBorrowedBy Book)

ObjectPropertyRange(isBorrowedBy Person)

$$\text{isBorrowedBy} \equiv \text{borrows}^{-}, \quad \text{T} \sqsubseteq \leq 1 \text{ isBorrowedBy}, \\ \exists \text{isBorrowedBy.T} \sqsubseteq \text{Book}, \quad \text{T} \sqsubseteq \forall \text{isBorrowedBy.Person}$$

3. SymmetricObjectProperty(hasSameGrade)

ObjectPropertyDomain(hasSameGrade Student)

ObjectPropertyRange(hasSameGrade Student)

$$\text{hasSameGrade}^{-} \sqsubseteq \text{hasSameGrade} \\ \exists \text{hasSameGrade.T} \sqsubseteq \text{Student}, \quad \text{T} \sqsubseteq \forall \text{hasSameGrade.Student}$$

Translating OWL into DL (cont.)

4. SubClassOf(Cow ObjectIntersectionOf(
Animal ObjectAllValuesFrom(eats
ObjectUnionOf(Plant ObjectSomeValuesFrom(partOf Plant))))))

$Cow \sqsubseteq Animal \sqcap \forall eats. (Plant \sqcup \exists partOf. Plant)$

5. EquivalentClasses(MadCow ObjectIntersectionOf(
Cow ObjectSomeValuesFrom(eats
ObjectIntersectionOf(Brain ObjectSomeValuesFrom(partOf Sheep))))))

$MadCow \equiv Cow \sqcap \exists eats. (Brain \sqcap \exists partOf. Sheep)$

Translating English into *ALC*

Using the individuals 'tom' and 'margaret', concepts 'OldLady', 'Person' and 'Cat' and role 'hasPet' represent the following knowledge base as an *ALC* knowledge base:

1. Every old lady has a pet.

$$\text{OldLady} \sqsubseteq \exists \text{hasPet.T}$$

2. Old ladies have only cats as their pets.

$$\text{OldLady} \sqsubseteq \forall \text{hasPet.Cat}$$

3. An old lady is a person.

$$\text{OldLady} \sqsubseteq \text{Person}$$

4. A cat owner is a person that has a cat as a pet.

$$\text{CatOwner} \equiv \text{Person} \sqcap \exists \text{hasPet.Cat}$$

5. Tom is Margaret's cat.

$$(\text{margaret}, \text{tom}) : \text{hasPet} \\ \text{tom} : \text{Cat}$$

- Is the statement 'Old ladies are cat owners' a logical consequence of this KB?
- Is the statement 'Margaret is an old lady' a logical consequence of this KB?

Computing concept interpretations

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

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

Show how $\cdot^{\mathcal{I}}$ can be extended to interpret the following class expressions:

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

- $(\neg A)^{\mathcal{I}} = \{c, d, e\}$, $(\neg B)^{\mathcal{I}} = \{a, c, e\} \Rightarrow (\neg A \sqcup \neg B)^{\mathcal{I}} = \{a, c, d, e\}$
- $(\exists R.(\neg A \sqcup \neg B))^{\mathcal{I}} = \{a, b, c\}$
- $(A \sqcap B)^{\mathcal{I}} = \{b, f\} \Rightarrow (\forall R.(B \sqcap A))^{\mathcal{I}} = \{d, e, f\}$
- $(\forall R.(\forall R.(B \sqcap A)))^{\mathcal{I}} = \{c, d, e, f\}$

Transforming concepts to NNF

Transform the concept

$$C = \exists R.A \sqcap \neg \forall R.(B \sqcap A) \sqcap \forall R.\neg(A \sqcap \neg B)$$

to an equivalent concept in negation normal form.

$$\exists R.A \sqcap \neg \forall R.(B \sqcap A) \sqcap \forall R.\neg(A \sqcap \neg B) \equiv \quad (\text{use } \neg \forall R.D \equiv \exists R.\neg D)$$

$$\exists R.A \sqcap \exists R.\neg(B \sqcap A) \sqcap \forall R.\neg(A \sqcap \neg B) \equiv \quad (\text{use } \neg(B \sqcap A) \equiv \neg B \sqcup \neg A)$$

$$\exists R.A \sqcap \exists R.(\neg B \sqcup \neg A) \sqcap \forall R.\neg(A \sqcap \neg B) \equiv \quad (\text{use } \neg(A \sqcap D) \equiv \neg A \sqcup \neg D)$$

$$\exists R.A \sqcap \exists R.(\neg B \sqcup \neg A) \sqcap \forall R.(\neg A \sqcup \neg \neg B) \equiv \quad (\text{use } \neg \neg B \equiv B)$$

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

Satisfiability by tableaux

Prove that $C = \exists R.A \sqcap \exists R.(\neg B \sqcup \neg A) \sqcap \forall R.(\neg A \sqcup B)$ is satisfiable

S_0	$= \{ x : \exists R.A \sqcap \exists R.(\neg B \sqcup \neg A) \sqcap \forall R.(\neg A \sqcup B) \}$	
$S_0 \rightarrow_{\sqcap} S_1$	$= S_0 \cup \{ x : \exists R.A, x : \exists R.(\neg B \sqcup \neg A), x : \forall R.(\neg A \sqcup B) \}$	
$S_1 \rightarrow_{\exists} S_2$	$= S_1 \cup \{ (x, y) : R, y : A \}$	
$S_2 \rightarrow_{\forall} S_3$	$= S_2 \cup \{ y : \neg A \sqcup B \}$	
$+ S_3 \rightarrow_{\sqcup} S_{4.1}$	$= S_3 \cup \{ y : \neg A \}$	clash
$+ S_3 \rightarrow_{\sqcup} S_{4.2}$	$= S_3 \cup \{ y : B \}$	
$S_{4.2} \rightarrow_{\exists} S_{5.2}$	$= S_{4.2} \cup \{ (x, z) : R, z : \neg B \sqcup \neg A \}$	
$S_{5.2} \rightarrow_{\forall} S_{6.2}$	$= S_{5.2} \cup \{ z : \neg A \sqcup B \}$	
$+ S_{6.2} \rightarrow_{\sqcup} S_{7.2.1}$	$= S_{6.2} \cup \{ z : \neg B \}$	
$+ S_{7.2.1} \rightarrow_{\sqcup} S_{8.2.1.1}$	$= S_{7.2.1} \cup \{ z : \neg A \}$	C is satisfiable

Satisfying model: $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where

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

NB: there are two(!) other branches in the tableau (and one of them contains clash)

Satisfiability by tableaux (full derivation tree)

S_0	$= \{ x : \exists R.A \sqcap \exists R.(\neg B \sqcup \neg A) \sqcap \forall R.(\neg A \sqcup B) \}$
$S_0 \rightarrow_{\sqcap} S_1$	$= S_0 \cup \{ x : \exists R.A, x : \exists R.(\neg B \sqcup \neg A), x : \forall R.(\neg A \sqcup B) \}$
$S_1 \rightarrow_{\exists} S_2$	$= S_1 \cup \{ (x, y) : R, y : A \}$
$S_2 \rightarrow_{\forall} S_3$	$= S_2 \cup \{ y : \neg A \sqcup B \}$
$+ S_3 \rightarrow_{\sqcup} S_{4.1}$	$= S_3 \cup \{ y : \neg A \}$ clash
$+ S_3 \rightarrow_{\sqcup} S_{4.2}$	$= S_3 \cup \{ y : B \}$
$S_{4.2} \rightarrow_{\exists} S_{5.2}$	$= S_{4.2} \cup \{ (x, z) : R, z : \neg B \sqcup \neg A \}$
$S_{5.2} \rightarrow_{\forall} S_{6.2}$	$= S_{5.2} \cup \{ z : \neg A \sqcup B \}$
$+ S_{6.2} \rightarrow_{\sqcup} S_{7.2.1}$	$= S_{6.2} \cup \{ z : \neg B \}$
$+ S_{7.2.1} \rightarrow_{\sqcup} S_{8.2.1.1}$	$= S_{7.2.1} \cup \{ z : \neg A \}$ C is satisfiable
$+ S_{7.2.1} \rightarrow_{\sqcup} S_{8.2.1.2}$	$= S_{7.2.1} \cup \{ z : B \}$ clash
$+ S_{6.2} \rightarrow_{\sqcup} S_{7.2.2}$	$= S_{6.2} \cup \{ z : \neg A \}$ C is satisfiable

NB: \rightarrow_{\sqcup} is not applicable to $z : \neg A \sqcup B$ in $S_{7.2.2}$ since $\neg A$ is already in $S_{7.2.2}$ otherwise (b) would be violated in the rule \rightarrow_{\sqcup} on slide 15 (Lecture 9)

Using tableaux: example

Does the inclusion $\neg\exists R.\neg A \sqcap B \sqcap C \sqsubseteq \neg(A \sqcap \neg B) \sqcap \forall R.A$
hold in **all** interpretations?

The answer to this question is NO **iff** there is an interpretation \mathcal{I}

with some element x such that

$$x \in (\neg\exists R.\neg A \sqcap B \sqcap C)^{\mathcal{I}} \quad \text{and} \quad x \notin (\neg(A \sqcap \neg B) \sqcap \forall R.A)^{\mathcal{I}}$$

or, equivalently,

$$x \in (\neg\exists R.\neg A \sqcap B \sqcap C)^{\mathcal{I}} \quad \text{and} \quad x \in (\neg(\neg(A \sqcap \neg B) \sqcap \forall R.A))^{\mathcal{I}}$$

Such an interpretation exists iff the constraint system

$$S_0 = \{x: \neg\exists R.\neg A \sqcap B \sqcap C, \quad x: \neg(\neg(A \sqcap \neg B) \sqcap \forall R.A)\}$$

can be expanded, using tableau rules, to a complete clash-free system.

Transform the concepts in S_0 into their negation normal forms, thus obtaining

$$S_0 = \{x: \forall R.A \sqcap B \sqcap C, \quad x: (A \sqcap \neg B) \sqcup \exists R.\neg A\}$$

Now use tableaux...

Using tableaux: example (cont.)

S_0	$= \{ x: \forall R.A \sqcap B \sqcap C, x: (A \sqcap \neg B) \sqcup \exists R.\neg A \}$	
$S_0 \rightarrow_{\sqcap} S_1$	$= S_0 \cup \{ x: \forall R.A, x: B, x: C \}$	
$+ S_1 \rightarrow_{\sqcup} S_{2.1}$	$= S_1 \cup \{ x: A \sqcap \neg B \}$	
$S_{2.1} \rightarrow_{\sqcap} S_{3.1}$	$= S_{2.1} \cup \{ x: A, x: \neg B \}$	clash
$+ S_1 \rightarrow_{\sqcup} S_{2.2}$	$= S_1 \cup \{ x: \exists R.\neg A \}$	
$S_{2.2} \rightarrow_{\exists} S_{3.2}$	$= S_{2.2} \cup \{ (x, y): R, y: \neg A \}$	
$S_{3.2} \rightarrow_{\forall} S_{4.2}$	$= S_{3.2} \cup \{ y: A \}$	clash

Therefore, there is no clash-free and complete extension of S_0 , which means that the inclusion $\neg \exists R.\neg A \sqcap B \sqcap C \sqsubseteq \neg(A \sqcap \neg B) \sqcap \forall R.A$

holds in **all** interpretations.