

# COMP 521: Knowledge Representation (Part 1: Modal and Description Logics)

Wiebe van der Hoek

Department of Computer Science  
University of Liverpool  
Liverpool, UK

<http://www.csc.liv.ac.uk/~wiebe/Teaching/COMP521/>

# Knowledge Representation and Reasoning Part 1: Modal and Description Logics

Wiebe van der Hoek

## Lecture 13: Description logic (1)

### Description logic: Introduction (1)

- The modal logics we have looked at so far are mainly concerned with modes of truth, but use a very simple logic to describe the facts to which these modes are applied, namely propositional logic.
- There are much more expressive logics than propositional logic that we could use instead.
- However, there is a trade-off between expressiveness of a logic and completeness and termination of deductive systems for a logic.

- p.1

### Description logic: Introduction (2)

- For second-order logic there is no complete deductive system;
- For first-order logic there are complete deductive systems, but in general they are not guaranteed to terminate;
- For propositional logic there are complete and terminating deductive systems, but the satisfiability of a propositional formula can in general not be decided in polynomial time (assuming  $NP \neq P$ );
- For propositional Horn logic there are complete and terminating deductive system which decide the satisfiability of any finite set of Horn clauses in polynomial time, but its expressive power is very limited.

- p.2

### Description logic: Introduction (3)

- Originally, description logics were intended to be fragments of first-order logics whose satisfiability problem is decidable in polynomial time.
- Currently, description logics are intended to be decidable logics (not necessarily fragments of first-order logic) with good deductive systems (not necessarily polynomial time decidable).
- Just as there are many different modal logics, there are also many different description logics.
- We will focus on the description logic  $\mathcal{ALC}$  (Attribute Logic with Complement) first introduced by Schmidt-Schauß and Smolka in 1988.

- p.3

### Description logic: Applications

Description logics have applications in

- semantic web
- configuration of telecoms equipment
- support for schema design, query optimisation, source integration for relational databases
- medical terminologies

- p.4

### $\mathcal{ALC}$ : Syntax (1)

- A signature of  $\mathcal{ALC}$  is given by an ordered tuple  $\Sigma = (\mathbf{O}, \mathbf{C}, \mathbf{R})$  of three disjoint alphabets:
  - the set  $\mathbf{C}$  of concept symbols,
  - the set  $\mathbf{R}$  of role symbols, and
  - the set  $\mathbf{O}$  of object symbols (or objects).Concept symbols and role symbols are also called atomic concepts and atomic roles.

- p.5

### $\mathcal{ALC}$ : Syntax (2)

- The set of concept terms (or just concepts) is inductively defined as follows:
  - ▶  $\top$  and  $\perp$  are concepts;
  - ▶ every concept symbol is a concept;
  - ▶ if  $C$  and  $D$  are concepts and  $R$  is a role symbol, then
    - $\neg C$  (complement of  $C$ )
    - $C \sqcup D$  (union of  $C$  and  $D$ )
    - $C \sqcap D$  (intersection of  $C$  and  $D$ )
    - $\forall R.C$  (universal restriction)
    - $\exists R.C$  (existential restriction)are concepts

- p.6



## Lifting the interpretation function

- The interpretation function  $\cdot^{\mathcal{I}}$  of a terminological interpretation  $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$  only maps concept symbols to subsets of  $\Delta$ .
- To give meaning to concepts, we need to lift  $\cdot^{\mathcal{I}}$  to a function  $\cdot^{\bar{\mathcal{I}}}$  mapping arbitrary concepts to subsets of  $\Delta$ .
- $\cdot^{\bar{\mathcal{I}}}$  is given by

$$\begin{aligned} \top^{\bar{\mathcal{I}}} &= \Delta & \perp^{\bar{\mathcal{I}}} &= \emptyset \\ A^{\bar{\mathcal{I}}} &= A^{\mathcal{I}} & (\neg C)^{\bar{\mathcal{I}}} &= \Delta \setminus C^{\bar{\mathcal{I}}} \\ (C \sqcup D)^{\bar{\mathcal{I}}} &= C^{\bar{\mathcal{I}}} \cup D^{\bar{\mathcal{I}}} & (C \sqcap D)^{\bar{\mathcal{I}}} &= C^{\bar{\mathcal{I}}} \cap D^{\bar{\mathcal{I}}} \\ (\forall R.C)^{\bar{\mathcal{I}}} &= \{x \mid \text{for every } y, (x, y) \in R^{\mathcal{I}} \text{ implies } y \in C^{\bar{\mathcal{I}}}\} \\ (\exists R.C)^{\bar{\mathcal{I}}} &= \{x \mid \text{there exists } y, (x, y) \in R^{\mathcal{I}} \text{ and } y \in C^{\bar{\mathcal{I}}}\} \end{aligned}$$

- p.15

## Lifting the interpretation function: Example

Let person and male be concept symbols, and hasChild be a role symbol.

Let the terminological interpretation  $\mathcal{I}_1 = (\Delta_1, \cdot^{\mathcal{I}_1})$  be given by

$$\begin{aligned} \Delta_1 &= \{tim, tom, jim, jane, blacky\} \\ person^{\mathcal{I}_1} &= \{tim, tom, jim, jane\} \\ male^{\mathcal{I}_1} &= \{tim, tom, jim\} \\ hasChild^{\mathcal{I}_1} &= \{(tim, tom), (tim, jane), (tom, jim)\} \end{aligned}$$

Then

$$\begin{aligned} (\neg male)^{\bar{\mathcal{I}}_1} &= \{jane, blacky\} \\ (person \sqcap \neg male)^{\bar{\mathcal{I}}_1} &= \{jane\} \\ (person \sqcap \exists hasChild.male)^{\bar{\mathcal{I}}_1} &= \{tim, tom\} \\ (person \sqcap \forall hasChild.male)^{\bar{\mathcal{I}}_1} &= \{tom, jim, jane\} \end{aligned}$$

- p.16

## Exercise 8

Let the terminological interpretation  $\mathcal{I}_2 = (\Delta_2, \cdot^{\mathcal{I}_2})$  be given by

$$\begin{aligned} \Delta_2 &= \{tim, tom, jay, jane, blacky\} \\ person^{\mathcal{I}_2} &= \{tim, tom, jay, jane\} \\ female^{\mathcal{I}_2} &= \{jane, jay, blacky\} \\ hasChild^{\mathcal{I}_2} &= \{(tim, tom), (tim, jane), (jane, tim)\} \end{aligned}$$

Determine the interpretation of the following concepts in  $\mathcal{I}_2$ :

- person  $\sqcap$  female
- female  $\sqcup$   $\exists$ hasChild.female

- p.17

## Exercise 8: Answer

Let the terminological interpretation  $\mathcal{I}_2 = (\Delta_2, \cdot^{\mathcal{I}_2})$  be given by

$$\begin{aligned} \Delta_2 &= \{tim, tom, jay, jane, blacky\} \\ person^{\mathcal{I}_2} &= \{tim, tom, jay, jane\} \\ female^{\mathcal{I}_2} &= \{jane, jay, blacky\} \\ hasChild^{\mathcal{I}_2} &= \{(tim, tom), (tim, jane), (jane, tim)\} \end{aligned}$$

Determine the interpretation of the following concepts in  $\mathcal{I}_2$ :

- person  $\sqcap$  female  
**Answer:** (person  $\sqcap$  female) $^{\bar{\mathcal{I}}_2} = \{jane, jay\}$
- female  $\sqcup$   $\exists$ hasChild.female  
**Answer:**  
(female  $\sqcup$   $\exists$ hasChild.female) $^{\bar{\mathcal{I}}_2} = \{jane, jay, blacky, tim\}$

- p.18

## Semantics of knowledge bases (1)

Let  $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$  be a terminological interpretation

- We say  $\mathcal{I}$  satisfies
  - a concept definition  $A \sqsubseteq C$  iff  $A^{\mathcal{I}} \subseteq C^{\bar{\mathcal{I}}}$
  - a concept definition  $A \doteq C$  iff  $A^{\mathcal{I}} = C^{\bar{\mathcal{I}}}$
  - a concept assertion  $a : C$  iff  $a^{\mathcal{I}} \in C^{\bar{\mathcal{I}}}$
  - a role assertion  $(a, b) : R$  iff  $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

- p.19

## Semantics of knowledge bases (2)

Let  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$  be a knowledge base and  $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$  be a terminological interpretation

- We say  $\mathcal{I}$  satisfies  $\mathcal{T}$  iff  $\mathcal{I}$  satisfies every concept definition in  $\mathcal{T}$
- We say  $\mathcal{I}$  satisfies  $\mathcal{A}$  iff  $\mathcal{I}$  satisfies every concept assertion and every role assertion in  $\mathcal{A}$
- We say  $\mathcal{I}$  satisfies  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$  iff  $\mathcal{I}$  satisfies both  $\mathcal{T}$  and  $\mathcal{A}$
- If  $\mathcal{I}$  satisfies  $\mathcal{K}$ , then  $\mathcal{I}$  is a model of  $\mathcal{K}$

- p.20

## Semantics of knowledge bases (3)

Let  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$  be a knowledge base and  $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$  be a terminological interpretation

- We say  $\mathcal{K}$  entails a concept definition, concept or role assertion  $\alpha$ , written  $\mathcal{K} \models \alpha$ , iff every terminological interpretation satisfying  $\mathcal{K}$  also satisfies  $\alpha$
- A concept  $C$  is coherent with respect to a TBox  $\mathcal{T}$  iff there is a model  $\mathcal{I}$  of  $\mathcal{T}$  such that  $C^{\mathcal{I}}$  is non-empty

- p.21

## Semantics of knowledge bases: Example (1)

TBox	
ABox	tim : person G500 : course (tim, G500) : enrolledOn

Then  $\mathcal{I}_3 = (\{tim, G500\}, \cdot^{\mathcal{I}_3})$  given by

$$\begin{aligned} tim^{\mathcal{I}_3} &= tim & G500^{\mathcal{I}_3} &= G500 \\ person^{\mathcal{I}_3} &= \{tim\} & course^{\mathcal{I}_3} &= \{G500\} \\ enrolledOn^{\mathcal{I}_3} &= \{(tim, G500)\} \end{aligned}$$

satisfies this knowledge base

- p.22

### Semantics of knowledge bases: Example (2)

TBox	$student \doteq person \sqcap \exists enrolledOn.course$
ABox	$tim : person$ $G500 : course$ $(tim, G500) : enrolledOn$

Then  $\mathcal{I}_4 = (\{tim, G500\}, \cdot^{\mathcal{I}_4})$  given by  
 $tim^{\mathcal{I}_4} = tim$        $G500^{\mathcal{I}_4} = G500$   
 $person^{\mathcal{I}_4} = \{tim\}$        $course^{\mathcal{I}_4} = \{G500\}$   
 $student^{\mathcal{I}_4} = \emptyset$   
 $enrolledOn^{\mathcal{I}_4} = \{(tim, G500)\}$

does not satisfy the TBox, since

$$tim \in (person \sqcap \exists enrolledOn.course)^{\overline{\mathcal{I}_4}} \text{ but } tim \notin student^{\overline{\mathcal{I}_4}}$$

- p.23

### Semantics of knowledge bases: Example (2)

TBox	$student \doteq person \sqcap \exists enrolledOn.course$
ABox	$tim : person$ $G500 : course$ $(tim, G500) : enrolledOn$

Then  $\mathcal{I}_5 = (\{tim, G500\}, \cdot^{\mathcal{I}_5})$  given by  
 $tim^{\mathcal{I}_5} = tim$        $G500^{\mathcal{I}_5} = G500$   
 $person^{\mathcal{I}_5} = \{tim\}$        $course^{\mathcal{I}_5} = \{G500\}$   
 $student^{\mathcal{I}_5} = \{tim\}$   
 $enrolledOn^{\mathcal{I}_5} = \{(tim, G500)\}$

does satisfy this knowledge base

- p.24

### Semantics of knowledge bases: Example (3)

TBox	$course \sqsubseteq \exists consistsOf.module$
ABox	$tim : person$ $tim : \neg module$ $G500 : course$ $G500 : \neg module$ $(tim, G500) : enrolledOn$

Then  $\mathcal{I}_6 = (\{tim, G500\}, \cdot^{\mathcal{I}_6})$  given by  
 $tim^{\mathcal{I}_6} = tim$        $G500^{\mathcal{I}_6} = G500$   
 $person^{\mathcal{I}_6} = \{tim\}$        $course^{\mathcal{I}_6} = \{G500\}$   
 $module^{\mathcal{I}_6} = \emptyset$   
 $enrolledOn^{\mathcal{I}_6} = \{(tim, G500)\}$        $consistOf^{\mathcal{I}_6} = \emptyset$

does not satisfy the TBox since

$$G500 \in course^{\overline{\mathcal{I}_6}} \text{ but } G500 \notin (\exists consistsOf.module)^{\overline{\mathcal{I}_6}} = \emptyset$$

- p.25

### Semantics of knowledge bases: Example (3)

TBox	$course \sqsubseteq \exists consistsOf.module$
ABox	$tim : person$ $tim : \neg module$ $G500 : course$ $G500 : \neg module$ $(tim, G500) : enrolledOn$

Then  $\mathcal{I}_7 = (\{tim, G500, C304\}, \cdot^{\mathcal{I}_7})$  given by  
 $tim^{\mathcal{I}_7} = tim$        $G500^{\mathcal{I}_7} = G500$   
 $person^{\mathcal{I}_7} = \{tim\}$        $course^{\mathcal{I}_7} = \{G500\}$   
 $module^{\mathcal{I}_7} = \{C304\}$   
 $enrolledOn^{\mathcal{I}_7} = \{(tim, G500)\}$        $consistOf^{\mathcal{I}_7} = \{(G500, C304)\}$

does satisfy this knowledge base

- p.26

### Semantics of knowledge bases (3)

- The examples illustrate that concept definitions put constraints on the interpretation of concept symbols and role symbols.
- They also illustrate that concept definitions put constraints on the domain as well, that is, we may have elements  $e$  in the domain of an interpretation without an object symbol in the ABox that denotes  $e$ .

- p.27

### Summary

- Overview on description logics
  - Trade-off between expressiveness and completeness/termination
  - Description logics as compromises with respect to this trade-off
  - Applications
- Syntax of  $\mathcal{ALC}$ 
  - Concepts
  - Concept definitions, concept and role assertions
  - ABox, TBox, knowledge bases

- p.28

## Knowledge Representation and Reasoning Part 1: Modal and Description Logics

Wiebe van der Hoek

### Lecture 14: Description logic (2)

### Last time ...

- Overview on description logics
  - Trade-off between expressiveness and completeness/termination
  - Description logics as compromises with respect to this trade-off
  - Applications
- Syntax of  $\mathcal{ALC}$ 
  - Concepts
  - Concept definitions, concept and role assertions
  - ABox, TBox, knowledge bases

- p.1

## Last time ...

- Semantics of  $\mathcal{ALC}$

- Terminological interpretations
- Lifting the interpretation function

$$(\forall R.C)^{\mathcal{I}} = \{x \mid \text{for every } y, (x, y) \in R^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}\}$$

$$(\exists R.C)^{\mathcal{I}} = \{x \mid \text{there exists } y, (x, y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$$

- Examples

## Semantics of knowledge bases: Example (4)

TBox	
ABox	tom : male jay : $\neg$ male

Then  $\mathcal{I}_8 = (\{tom, jay\}, \cdot^{\mathcal{I}_8})$  given by

$$tom^{\mathcal{I}_8} = jay \quad jay^{\mathcal{I}_8} = tom \quad male^{\mathcal{I}_8} = \{jay\}$$

satisfies this knowledge base since

$$jay = tom^{\mathcal{I}_8} \in male^{\mathcal{I}_8} = \{jay\}$$

and

$$tom = jay^{\mathcal{I}_8} \in \neg male^{\mathcal{I}_8} = \{tom, jay\} \setminus \{jay\} = \{tom\}$$

(but we try to avoid such confusing interpretations)

- p.2

- p.3

## Semantics of knowledge bases: Example (5)

TBox	male $\doteq$ $\neg$ female
ABox	robin : male $\sqcup$ female

Then  $\mathcal{I}_9 = (\{robin\}, \cdot^{\mathcal{I}_9})$  given by

$$robin^{\mathcal{I}_9} = robin \quad male^{\mathcal{I}_9} = \{robin\} \quad female^{\mathcal{I}_9} = \emptyset$$

and  $\mathcal{I}_{10} = (\{robin\}, \cdot^{\mathcal{I}_{10}})$  given by

$$robin^{\mathcal{I}_{10}} = robin \quad male^{\mathcal{I}_{10}} = \emptyset \quad female^{\mathcal{I}_{10}} = \{robin\}$$

satisfy this knowledge base,

but not  $\mathcal{I}_{11} = (\{robin\}, \cdot^{\mathcal{I}_{11}})$  given by

$$robin^{\mathcal{I}_{11}} = robin \quad male^{\mathcal{I}_{11}} = \emptyset \quad female^{\mathcal{I}_{11}} = \emptyset$$

- p.4

- p.5

## Checking terminological interpretations

Let  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$  be a knowledge base and

$\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$  be an interpretation with finite  $\Delta$

To check whether  $\mathcal{I}$  satisfies  $\mathcal{K}$  we proceed as follows:

- For each concept assertion  $a : C$  in  $\mathcal{A}$ , compute  $C^{\mathcal{I}}$  using the definition of  $\cdot^{\mathcal{I}}$  and check that  $a^{\mathcal{I}}$  is in  $C^{\mathcal{I}}$ .
- For each concept definition  $A \sqsubseteq C$  in  $\mathcal{T}$ , compute  $C^{\mathcal{I}}$  and check that  $A^{\mathcal{I}}$  (which is explicitly given by  $\cdot^{\mathcal{I}}$ ) is a subset of  $C^{\mathcal{I}}$ .
- For each concept definition  $A \doteq C$  in  $\mathcal{T}$ , compute  $C^{\mathcal{I}}$  and check that  $A^{\mathcal{I}}$  (which is explicitly given by  $\cdot^{\mathcal{I}}$ ) is equal to  $C^{\mathcal{I}}$ .

Note that there is nothing to check for the role assertions in  $\mathcal{A}$ , but they are relevant for the computation of  $\cdot^{\mathcal{I}}$ .

## Checking terminological interpretations: Example

TBox	course $\sqsubseteq$ $\exists$ consistsOf.module
ABox	G500 : course    G500 : $\neg$ module

Let  $\mathcal{I}_{12} = (\{G500, G520, C304, C305\}, \cdot^{\mathcal{I}_{12}})$  be given by

$$G500^{\mathcal{I}_{12}} = G500$$

$$course^{\mathcal{I}_{12}} = \{G500, G520\} \quad module^{\mathcal{I}_{12}} = \{C304, C305\}$$

$$consistsOf^{\mathcal{I}_{12}} = \{(G500, C304), (G500, C305), (G520, C304)\}$$

Checking that  $\mathcal{I}_{12}$  satisfies this knowledge base:

$$1(a) \quad G500 = G500^{\mathcal{I}_{12}} \in course^{\mathcal{I}_{12}} = \{G500, G520\}$$

$$1(b) \quad G500 = G500^{\mathcal{I}_{12}} \in (\neg module)^{\mathcal{I}_{12}} = \{G500, G520\}$$

$$2(a) \quad \{G500, G520\} = course^{\mathcal{I}_{12}} \subseteq (\exists consistsOf.module)^{\mathcal{I}_{12}} \\ = \{G500, G520\}$$

- p.6

- p.7

## Interpretations and labelled directed graphs

- We can depict a terminological interpretation  $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$  over a signature  $(\mathbf{O}, \mathbf{C}, \mathbf{R})$  by a labelled directed graph

$$G = (V, E, \delta_V, \delta_E)$$

( $\delta_V$  and  $\delta_E$  are labelling functions for vertices and edges, respectively).

- Also, given a labelled directed graph  $G$  we can construct a terminological interpretation  $\mathcal{I}$  such that  $G$  depicts  $\mathcal{I}$ .
- Finally, we can also depict an ABox over  $(\mathbf{O}, \mathbf{C}, \mathbf{R})$  by a labelled directed graph (with a different labelling of nodes and edges), and vice versa.

## Construction of $G$ from $\mathcal{I}$

Construction of  $G = (V, E, \delta_V, \delta_E)$  from  $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ :

- The set of vertices  $V$  of  $G$  is identical to the domain  $\Delta$  of  $\mathcal{I}$ .
- A vertex  $a$  in  $V$  is labelled by each object symbol  $o$  with  $o^{\mathcal{I}} = a$  and each concept symbol  $A$  such that  $a \in A^{\mathcal{I}}$ , that is,  $\delta_V(a) = (\{o \in \mathbf{O} \mid o^{\mathcal{I}} = a\}, \{A \in \mathbf{C} \mid a \in A^{\mathcal{I}}\})$ .
- The set of edges  $E$  of  $G$  is identical to the union of all the  $R^{\mathcal{I}}$  for  $R \in \mathbf{R}$ , that is,  $E = \bigcup_{R \in \mathbf{R}} R^{\mathcal{I}}$ .  
So, there is an edge from vertex  $a$  to vertex  $b$  iff there is a role symbol  $R \in \mathbf{R}$  such that  $(a, b) \in R^{\mathcal{I}}$ .
- An edge from  $a$  to  $b$  is labelled by each role symbol  $R$  such that  $(a, b) \in R^{\mathcal{I}}$ , that is,  $\delta_E((a, b)) = \{R \in \mathbf{R} \mid (a, b) \in R^{\mathcal{I}}\}$ .

- p.8

- p.9

## Construction of $G$ from $\mathcal{I}$ : Example

Let  $\mathcal{I}_7 = (\{tim, G500, C304\}, \cdot^{\mathcal{I}_7})$  be given by

$$tim^{\mathcal{I}_7} = tim \quad G500^{\mathcal{I}_7} = G500$$

$$person^{\mathcal{I}_7} = \{tim\} \quad course^{\mathcal{I}_7} = \{G500\}$$

$$module^{\mathcal{I}_7} = \{C304\}$$

$$enrolledOn^{\mathcal{I}_7} = \{(tim, G500)\} \quad consistsOf^{\mathcal{I}_7} = \{(G500, C304)\}$$

Then  $\mathcal{I}_7$  is depicted by the following graph  $G_7$



- p.8

- p.9

## Construction of $\mathcal{I}$ from $G$

Construction of  $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$  from  $G = (V, E, \delta_V, \delta_E)$ :

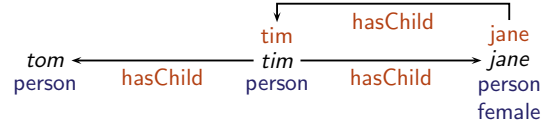
1. The domain  $\Delta$  is identical to the set of vertices  $V$  of  $G$ .
2. For each object symbol  $o \in \mathbf{O}$  there is exactly one vertex  $a$  labelled by  $o$ , that is,  $o \in \text{frst}(\delta_V(a))$ , and we define  $o^{\mathcal{I}} = a$ .
3. For each concept symbol  $A \in \mathbf{C}$  let  $A^{\mathcal{I}}$  be the set of all vertices labelled by  $A$ , that is,  $A^{\mathcal{I}} = \{a \in V \mid A \in \text{snd}(\delta_V(a))\}$ .
4. For each role symbol  $R \in \mathbf{R}$  let  $R^{\mathcal{I}}$  be the set of all pairs  $(a, b)$  such that there is an edge from  $a$  to  $b$  labelled by  $R$ , that is,  $R^{\mathcal{I}} = \{(a, b) \mid (a, b) \in E \text{ and } R \in \delta_E((a, b))\}$

where for any ordered pair  $(X, Y)$ ,  $\text{frst}(X, Y) = X$  and  $\text{snd}(X, Y) = Y$ .

- p.10

## Construction of $\mathcal{I}$ from $G$ : Example

Let the following labelled directed graph be given



Then the corresponding terminological interpretation

$\mathcal{I}_{13} = (\{tom, tim, jane\}, \cdot^{\mathcal{I}_{13}})$  is given by

$$\begin{aligned} \text{tim}^{\mathcal{I}_{13}} &= \text{tim} & \text{jane}^{\mathcal{I}_{13}} &= \text{jane} \\ \text{person}^{\mathcal{I}_{13}} &= \{\text{tom}, \text{tim}, \text{jane}\} & \text{female}^{\mathcal{I}_{13}} &= \{\text{jane}\} \\ \text{hasChild}^{\mathcal{I}_{13}} &= \{(\text{tim}, \text{tom}), (\text{tim}, \text{jane}), (\text{jane}, \text{tim})\} \end{aligned}$$

- p.11

## Exercise 9

Construct the corresponding terminological interpretation for the labelled directed graph below.



- p.12

## Exercise 9: Answer

Construct the corresponding terminological interpretation for the labelled directed graph below.



The corresponding terminological interpretation is

$\mathcal{I} = (\{jim, sue, tim, tom\}, \cdot^{\mathcal{I}})$  with

$$\begin{aligned} \text{jim}^{\mathcal{I}} &= \text{jim} & \text{sue}^{\mathcal{I}} &= \text{sue} \\ \text{person}^{\mathcal{I}} &= \{\text{jim}, \text{sue}, \text{tim}, \text{tom}\} \\ \text{hasFriend}^{\mathcal{I}} &= \{(\text{jim}, \text{sue}), (\text{tim}, \text{tom}), (\text{tom}, \text{tom})\} \end{aligned}$$

- p.13

## Construction of $G$ from an ABox

Construction of  $G = (V, E, \delta_V, \delta_E)$  from an ABox  $\mathcal{A}$ :

1. The set of vertices  $V$  of  $G$  is identical to the set of object symbols occurring in the ABox.
2. A vertex  $a$  in  $V$  is labelled by each concept  $C$  such that the concept assertion  $a : C$  occurs in the ABox, that is,  $\delta_V(a) = \{C \mid a : C \text{ occurs in } \mathcal{A}\}$ .
3. The set of edges  $E$  of  $G$  is the set of all pairs of object symbols  $(a, b)$  such that  $(a, b) : R$  occurs in the ABox for any role symbol  $R$ , that is,  $E = \bigcup_{R \in \mathbf{R}} \{(a, b) \mid (a, b) : R \text{ occurs in } \mathcal{A}\}$ .
4. An edge from  $a$  to  $b$  is labelled by each role symbol  $R$  such that  $(a, b) : R$  occurs in the ABox, that is,  $\delta_E((a, b)) = \{R \in \mathbf{R} \mid (a, b) : R \text{ occurs in } \mathcal{A}\}$ .

- p.14

## Construction of $G$ from an ABox: Example

Let an ABox be given by

ABox	robin : male $\sqcup$ female
	(robin, G500) : enrolledOn
	G500 : $\exists$ consistsOf.module

Then the corresponding labelled directed graph is



Note that vertices are object symbols (not elements of the domain of a terminological interpretation) and are labelled by concepts (not concept symbols).

- p.15

## Construction of an ABox from $G$

Construction of an ABox  $\mathcal{A}$  from  $G = (V, E, \delta_V, \delta_E)$ :

1. The set of object symbols of  $\mathcal{A}$  is identical to  $V$ , the set of concept symbols of  $\mathcal{A}$  is identical to the set of all concept symbols occurring in the concepts labelling the vertices of  $G$ , and the set of role symbols of  $\mathcal{A}$  is identical to the set of role symbols labelling edges of  $G$ .
2. For each element  $a$  of  $V$  and concept  $C$  labelling  $a$ ,  $\mathcal{A}$  contains a concept assertion  $a : C$ .  
For each edge from  $a$  to  $b$  in  $G$  labelled by a role symbol  $R$ ,  $\mathcal{A}$  contains a role assertion  $(a, b) : R$ .  
That is,  $\mathcal{A} = \{a : C \mid a \in V \text{ and } C \in \delta_V(a)\} \cup \{(a, b) : R \mid (a, b) \in E \text{ and } R \in \delta_E((a, b))\}$

- p.16

## Construction of an ABox from $G$ : Example

Let a labelled directed graph be given by



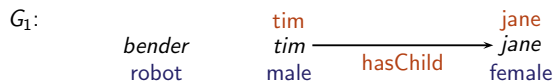
the corresponding ABox is given by

ABox	tom : person
	tim : $\exists$ hasChild.male
	blacky : $\neg$ person
	blacky : female
	(tom, blacky) : owns

- p.17

## Labelled directed graphs and interpretations

Consider the following graph  $G_1$  depicting a terminological interpretation  $\mathcal{I}_1$ :

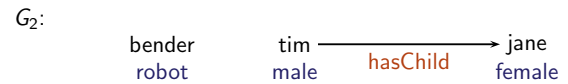


The absence of `male` and `female` as labels of `bender` means that `bender` is neither in `male` <sup>$\mathcal{I}_1$</sup>  nor `female` <sup>$\mathcal{I}_1$</sup> . Also the absence of an edge from `tim` to `bender` means that the pair  $(\text{tim}, \text{bender})$  is not in `hasChild` <sup>$\mathcal{I}_1$</sup>  (nor in the interpretation of any other role symbol).

- p.18

## Labelled directed graphs and ABoxes

In contrast, consider the following graph  $G_2$  depicting an ABox  $\mathcal{A}_2$  which is almost identical to  $G_1$ :



Let  $\mathcal{I}_2$  be an arbitrary terminological interpretation satisfying  $\Delta_2$ .

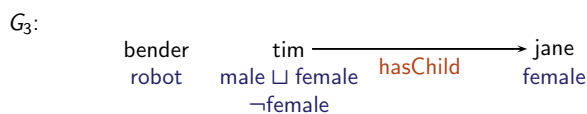
The absence of `male` and `female` as labels of `bender` means that the ABox does not contain any **explicit** information whether `bender` <sup>$\mathcal{I}_2$</sup>  has to be in `male` <sup>$\mathcal{I}_2$</sup>  and `female` <sup>$\mathcal{I}_2$</sup> . It only states that `bender` <sup>$\mathcal{I}_2$</sup>  has to be in `robot` <sup>$\mathcal{I}_2$</sup> .

Analogously, for the missing edge from `tim` to `bender`.

- p.19

## Explicit and implicit information in ABoxes

Consider the following graph  $G_3$  depicting an ABox  $\mathcal{A}_3$ :



Let  $\mathcal{I}_3$  be an arbitrary terminological interpretation satisfying  $\Delta_3$ .

The ABox **explicitly** states that `tim` <sup>$\mathcal{I}_3$</sup>  has to be an element of  $(\text{male} \sqcup \text{female})^{\mathcal{I}_3}$  and  $(\neg \text{female})^{\mathcal{I}_3}$ . Since `female` <sup>$\mathcal{I}_3$</sup>  and  $(\neg \text{female})^{\mathcal{I}_3}$  are disjoint, `tim` <sup>$\mathcal{I}_3$</sup>  has to be in `male` <sup>$\mathcal{I}_3$</sup> .

Thus, the ABox **implicitly** states that `tim` <sup>$\mathcal{I}_3$</sup>  is an element of `male` <sup>$\mathcal{I}_3$</sup> .

- p.20

## Summary

- Graph representations of
  - terminological interpretations
  - ABoxes
- Interpretation of absent information in graphs representing interpretations versus graphs representing ABoxes

- p.21

# Knowledge Representation and Reasoning Part 1: Modal and Description Logics

Wiebe van der Hoek

## Lecture 15: Description logic (3)

## Last time ...

- Graph representations of
  - terminological interpretations
  - ABoxes
- Interpretation of absent information in graphs representing interpretations versus graphs representing ABoxes

- p.1

## Inferential services (1)

Let  $\mathcal{K}$  be a knowledge base.

A knowledge base  $\mathcal{K}$  entails a concept definition, concept or role assertion  $\alpha$ , written  $\mathcal{K} \models \alpha$ , iff every terminological interpretation satisfying  $\mathcal{K}$  also satisfies  $\alpha$ .

The **entailment problem** is to decide, given a knowledge base  $\mathcal{K}$  and a concept definition, concept or role assertion  $\alpha$ , whether  $\mathcal{K}$  entails  $\alpha$ .

The entailment problem can be further divided into the following tasks:

- Subsumption of concepts: decide whether  $\emptyset \models C \sqsubseteq D$  holds for concepts  $C$  and  $D$  in which case  $D$  subsumes  $C$ ,  $C$  is subsumed by  $D$ , and  $C$  is more specific than  $D$

- p.2

## Inferential services (2)

- Subsumption of concepts with respect to a TBox  $\mathcal{T}$ : decide whether  $(\mathcal{T}, \emptyset) \models C \sqsubseteq D$  holds
- Equivalence of concepts (with respect to a TBox  $\mathcal{T}$ ): decide for two given concepts  $C$  and  $D$  whether  $C$  subsumes  $D$  (with respect to a TBox  $\mathcal{T}$ ) and  $D$  subsumes  $C$  (with respect to a TBox  $\mathcal{T}$ )
- Instance checking: decide whether a given knowledge base  $\mathcal{K}$  entails a given concept assertion  $a : C$

These tasks are also called **inferential services** that are provided by description logic systems

- p.3

### Inferential services (3)

Additional inferential services are the following:

- **Classification of a TBox  $\mathcal{T}$ :**  
decide for all concept symbols  $A$  and  $B$  occurring in  $\mathcal{T}$  whether  $A$  subsumes  $B$  or  $B$  subsumes  $A$  with respect to  $\mathcal{T}$ .
- **Coherence of a concept (with respect to a TBox  $\mathcal{T}$ ):**  
decide for a given concept  $C$  whether there is a terminological interpretation  $\mathcal{I}$  (satisfying  $\mathcal{T}$ ) such that  $C^{\mathcal{I}}$  is non-empty

- p.4

### Inferential services (4)

- **Consistency of an ABox  $\mathcal{A}$  with respect to a TBox  $\mathcal{T}$ :**  
decide whether the knowledge base  $(\mathcal{T}, \mathcal{A})$  is satisfiable
- **Realization:**  
compute for an object symbol  $a$  in a knowledge base  $\mathcal{K}$  the set of most specific (with respect to the subsumption relation) concept symbols  $A$  such that  $\mathcal{K} \models a : A$
- **Retrieval:**  
compute for a given concept  $C$  in a knowledge base  $\mathcal{K}$  those object symbols  $a$  such that  $\mathcal{K}$  entails  $a : C$

- p.5

### Taxonomy

- The subsumption relation is a reflexive and transitive relation on concept symbols in  $\mathcal{T}$ .
- The **taxonomy**  $T$  is the minimal binary relation such that its reflexive, transitive closure is identical to the inverse of the subsumption relation on symbols in  $\mathcal{T}$ .  
That is, for two distinct concept symbols  $A$  and  $B$ , the pair  $(A, B)$  is in the taxonomy  $T$  iff (i)  $B$  subsumes  $A$  and (ii) there is no concept symbol  $A'$  distinct to  $A$  and  $B$  such that  $B$  subsumes  $A'$  and  $A'$  subsumes  $A$ .
- A **taxonomy**  $T$  can be depicted by a **directed graph** whose vertices are concept symbols and there is an edge from concept symbol  $A$  to concept symbol  $B$  iff the pair  $(A, B)$  is in the taxonomy  $T$ .

- p.6

### Inferential services: Example

TBox	parent $\doteq$ person $\sqcap$ $\exists$ hasChild.person	
	father $\doteq$ parent $\sqcap$ male	
	grandParent $\doteq$ person $\sqcap$ $\exists$ hasChild.parent	
ABox	jim : person	jim : male
	tom : person	sue : person
	(jim, tom) : hasChild	(tom, sue) : hasChild

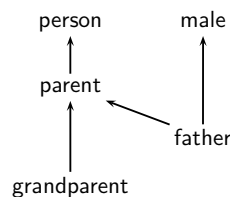
- **Subsumption of concepts:**
  - person subsumes person  $\sqcap$   $\exists$ hasChild.person
  - parent and male both subsume parent  $\sqcap$  male
- **Subsumption wrt. to a TBox:**
  - person subsumes parent (every parent is a person)
  - parent and male both subsume father

- p.7

### Inferential services: Example

TBox	parent $\doteq$ person $\sqcap$ $\exists$ hasChild.person	
	father $\doteq$ parent $\sqcap$ male	
	grandParent $\doteq$ person $\sqcap$ $\exists$ hasChild.parent	
ABox	jim : person	jim : male
	tom : person	sue : person
	(jim, tom) : hasChild	(tom, sue) : hasChild

- **Classification of the TBox:**



- p.7

### Inferential services: Example

TBox	parent $\doteq$ person $\sqcap$ $\exists$ hasChild.person	
	father $\doteq$ parent $\sqcap$ male	
	grandParent $\doteq$ person $\sqcap$ $\exists$ hasChild.parent	
ABox	jim : person	jim : male
	tom : person	sue : person
	(jim, tom) : hasChild	(tom, sue) : hasChild

- **Instance checking:**
  - sue is an instance of person
  - tom is an instance of person, parent
  - jim is an instance of male, person, parent, father, and grandparent

- p.7

### Inferential services: Example

TBox	parent $\doteq$ person $\sqcap$ $\exists$ hasChild.person	
	father $\doteq$ parent $\sqcap$ male	
	grandParent $\doteq$ person $\sqcap$ $\exists$ hasChild.parent	
ABox	jim : person	jim : male
	tom : person	sue : person
	(jim, tom) : hasChild	(tom, sue) : hasChild

- **Realization:**  
The most specific concept symbol(s) we can attribute to
  - sue is **person**
  - tom is **parent** (which is more specific than person)
  - jim are **grandparent** (which is more specific than parent and person) and **father** (which is more specific than male)

- p.7

### Inferential services: Example

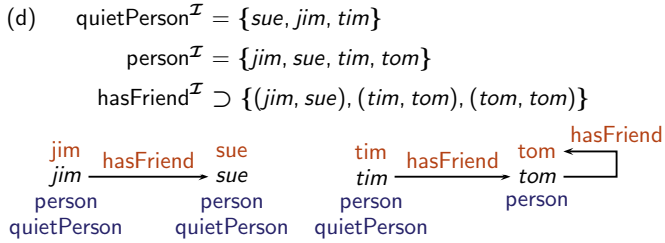
TBox	parent $\doteq$ person $\sqcap$ $\exists$ hasChild.person	
	father $\doteq$ parent $\sqcap$ male	
	grandParent $\doteq$ person $\sqcap$ $\exists$ hasChild.parent	
ABox	jim : person	jim : male
	tom : person	sue : person
	(jim, tom) : hasChild	(tom, sue) : hasChild

- **Retrieval:**  
The object symbols which are instances of
  - person are: **sue, tom, jim**
  - male are: **jim**
  - parent are: **tom, jim**
  - father are: **jim**
  - grandparent are: **jim**

- p.7



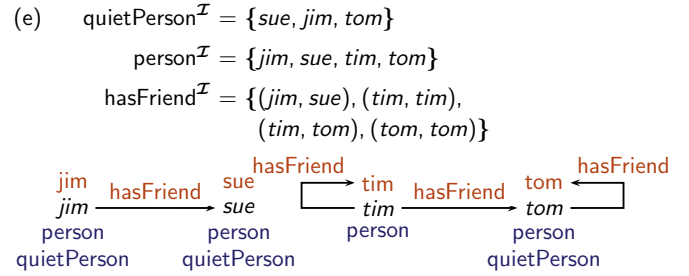
### Exercise 10: Answer (6)



There is no interpretation with  $\text{quietPerson}^{\mathcal{I}} = \{\text{sue}, \text{jim}, \text{tim}\}$  that satisfies the knowledge base, since  $\text{tim} \in \text{quietPerson}^{\mathcal{I}}$  implies any element related to  $\text{tim}$  via  $\text{hasFriend}^{\mathcal{I}}$  has to be in  $\text{quietPerson}^{\mathcal{I}}$ . Obviously,  $\text{tom}$  violates this constraint. (Note  $\text{tim}^{\mathcal{I}} = \text{tim} \neq \text{tom} = \text{tom}^{\mathcal{I}}$  by the unique name assumption)

- p.14

### Exercise 10: Answer (7)

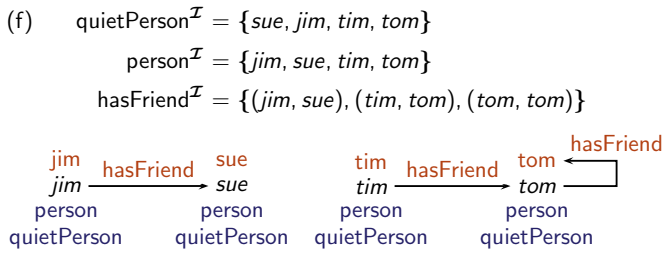


Does satisfy the knowledge base,

since except for  $\text{tim}$  all other elements of the domain are only related via  $\text{hasFriend}^{\mathcal{I}}$  to elements in  $\text{quietPerson}^{\mathcal{I}}$ ;  $\text{tim}$ , on the other hand, is related via  $\text{hasFriend}^{\mathcal{I}}$  to an element which is not in  $\text{quietPerson}^{\mathcal{I}}$ , namely, himself.

- p.15

### Exercise 10: Answer (8)



Does satisfy the knowledge base,

since every element  $a$  of the domain is only related via  $\text{hasFriend}^{\mathcal{I}}$  to elements of the domain that are in  $\text{quietPerson}^{\mathcal{I}}$ .

- p.16

### Reduction to satisfiability (1)

Theorem 15.1

Let  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$  be knowledge base over  $\Sigma = (\mathbf{O}, \mathbf{C}, \mathbf{R})$ .

$C$  and  $D$  be concepts over  $\Sigma$

$a$  be an object symbol in  $\mathcal{A}$

- Subsumption of concepts with respect to  $\mathcal{T}$ :  
 $C$  subsumes  $D$  wrt.  $\mathcal{T}$  iff  $(\mathcal{T}, \{x : (C \sqcap \neg D)\})$  is unsatisfiable for some arbitrary object symbol  $x$
- Instance checking:  
 $\mathcal{K}$  entails  $a : C$  iff  $(\mathcal{T}, \mathcal{A} \cup \{a : \neg C\})$  is unsatisfiable
- Coherence of a concept with respect to  $\mathcal{T}$ :  
 $C$  is coherent wrt.  $\mathcal{T}$  iff  $(\mathcal{T}, \{x : C\})$  is satisfiable for some arbitrary object symbol  $x$

- p.17

### Reduction to satisfiability (2)

- Theorem 15.1 tells us that the inferential services
  - Subsumption of concepts (with respect to a TBox)
  - Instance checking
  - Coherence of a concept (with respect to a TBox)
 can be provided by using the inferential service of deciding the
  - Consistency of an ABox with respect to a TBox
- What about the remaining inferential services?

- p.18

### Equivalence of concepts

- To decide whether two given concepts  $C$  and  $D$  are equivalent, we simply have to perform two subsumption test:
  - test whether  $C$  subsumes  $D$  (with respect to a TBox  $\mathcal{T}$ ) and
  - test whether  $D$  subsumes  $C$  (with respect to a TBox  $\mathcal{T}$ ) $C$  and  $D$  are equivalent iff both tests succeed.

The two subsumption tests can in turn be reduced to two consistency test using Theorem 15.1.

- p.19

### Classification of a TBox

- To classify the concepts symbols of a TBox  $\mathcal{T}$ , we check for all pairs  $(A, B)$  of distinct concept symbols occurring in  $\mathcal{T}$  whether  $A$  subsumes  $B$ .  
 (There are better algorithms which reduce the number of tests required, but this is not our concern here)

Each subsumption test can be reduced to a consistency test using Theorem 15.1.

- p.20

### Realization (1)

To compute for an object symbol  $a$  in a knowledge base  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$  the set of most specific concept symbols  $A$  such that  $\mathcal{K} \models a : A$ , we can proceed as follows:

1. Determine the set  $\mathbf{C}_{\mathcal{K}}$  of all concept symbols occurring in  $\mathcal{K}$ .
2. Determine the set  $\mathcal{C}_a$  of all concept symbols  $A$  in  $\mathbf{C}_{\mathcal{K}}$  such that  $\mathcal{K} \models a : A$ , that is,  $\mathcal{C}_a = \{A \in \mathbf{C}_{\mathcal{K}} \mid \mathcal{K} \models a : A\}$  by performing instance checks for each concept symbol.
3. For each pair  $(A, B)$  of distinct concept symbols in  $\mathcal{C}_a$ , test whether  $A$  subsumes  $B$  with respect to the TBox  $\mathcal{T}$ . If this is the case, delete  $A$  from  $\mathcal{C}_a$ .

The set  $\mathcal{C}_a$  remaining after the last step is the set of most specific concept symbols  $A$  such that  $\mathcal{K} \models a : A$ .

- p.21

## Realization (2)

This procedure reduces the inferential service of realization to

- instance checking and
- subsumption of concepts with respect to a TBox

Each of these inferential services can in turn be reduced to consistency testing using Theorem 15.1.

- p.22

## Retrieval (1)

To compute for a given concept  $C$  in a knowledge base  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$  those object symbols  $a$  such that  $\mathcal{K}$  entails  $a : C$ , we proceed as follows

1. Determine the set  $O_{\mathcal{A}}$  of object symbols occurring in the ABox  $\mathcal{A}$  by inspecting each of the concept and role assertions in  $\mathcal{A}$
2. Determine the set  $O_C$  defined by

$$O_C = \{a \in O_{\mathcal{A}} \mid \mathcal{K} \models a : C\}$$

by performing an instance check  $\mathcal{K} \models a : C$  for each object symbol  $a$  in  $O_{\mathcal{A}}$ .

The set  $O_C$  is the set of object symbols we are looking for.

- p.23

## Retrieval (2)

This procedure reduces the inferential service of retrieval to

- instance checking

The inferential service of instance checking can in turn be reduced to consistency testing using Theorem 15.1.

- p.24

## Inferential services: Summary

All inferential services

- Subsumption of concepts (with respect to a TBox)
- Equivalence of concepts (with respect to a TBox)
- Instance checking
- Classification of a TBox
- Coherence of a concept (with respect to a TBox)
- Realization
- Retrieval

can be realised by

- Consistency of an ABox (with respect to a TBox)

using an auxiliary procedure for some of the services.

Thus, in the following we focus on this inferential service alone.

- p.25

# Knowledge Representation and Reasoning Part 1: Modal and Description Logics

Wiebe van der Hoek

## Lecture 16: Description logic (4)

0-1

## Last time ...

- Inferential services of a description logic system
  - Subsumption of concepts (with respect to a TBox)
  - Equivalence of concepts (with respect to a TBox)
  - Instance checking
  - Classification of a TBox
  - Coherence of a concept (with respect to a TBox)
  - Consistency of an ABox (with respect to a TBox)
  - Realization
  - Retrieval
- All inferential services can be reduced to  
Consistency of an ABox (with respect to a TBox)

- p.1

## Consistency of an ABox (with respect to a TBox)

- There are various approaches that can be used to test the consistency of an ABox with respect to a TBox.
- We will first focus on a simple, so called tableaux-based, approach that applies only to acyclic knowledge bases and proceeds in five steps:
  1. Elimination of primitive concept definitions from the TBox
  2. Expansion of the TBox
  3. Elimination of defined concepts from the ABox
  4. Transforming the ABox into negation normal form
  5. Application of completion rules to the ABox

- p.2

## Acyclic knowledge bases (1)

- We say, a concept symbol  $A$  uses a concept symbol  $B$  in a TBox  $\mathcal{T}$  directly iff  $\mathcal{T}$  contains a concept definition of the form  $A \doteq C$  or  $A \sqsubseteq C$  such that  $B$  occurs in  $C$ .
- A concept symbol  $A_0$  uses  $A_n$  iff there is a sequence of symbols  $A_0, \dots, A_n$  such that  $A_i$  uses  $A_{i+1}$  directly, for every  $i$ ,  $1 \leq i \leq n-1$ .
- A knowledge base  $\mathcal{K}$  contains a terminological cycle iff some concept symbol uses itself in the TBox of  $\mathcal{K}$ .

- p.3

## Acyclic knowledge bases (2)

An acyclic knowledge base is a knowledge base  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$  that satisfies the following two constraints:

1. A concept symbol may occur at most once on the left-hand side of a concept definition in  $\mathcal{T}$ , and
2.  $\mathcal{K}$  contains no terminological cycles.

- p.4

## Terminological cycles: Examples

- Consider the following concept definition:

$$\text{quietPerson} \doteq \text{person} \sqcap \forall \text{hasFriend}.\text{quietPerson}$$

quietPerson directly uses person and quietPerson.

So, quietPerson uses itself and a knowledge base containing this concept definition contains a terminological cycle.

- Consider the following two concept definitions:

$$\text{formalApproach} \sqsubseteq \text{logicBasedApproach}$$

$$\text{logicBasedApproach} \sqsubseteq \text{formalApproach}$$

Here formalApproach directly uses logicBasedApproach which in turn directly uses formalApproach.

So, formalApproach uses itself.

- p.5

## Elimination of primitive concept definitions

Let  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$  be an acyclic knowledge base.

We can transform  $\mathcal{K}$  into an acyclic knowledge base  $\mathcal{K}^*$  not containing any primitive concept definitions in the following way:

- Delete any concept definition of the form  $A \sqsubseteq \top$  from  $\mathcal{T}$ .
- Replace any remaining concept definition of the form

$$A \sqsubseteq C$$

by

$$A \doteq C \sqcap A^*$$

where  $A^*$  is a new concept symbol uniquely associated with  $A$  that does not occur in  $\mathcal{K}$ .  $A^*$  is the primitive component of  $A$ .

- p.6

## Elimination of primitive concept definitions: Example

Consider the following TBox:

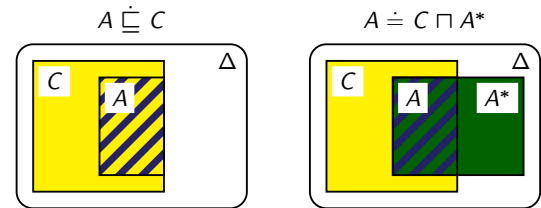
TBox	male $\sqsubseteq \neg \text{female}$
	parent $\doteq \text{person} \sqcap \exists \text{hasChild}.\text{person}$
	happyParent $\sqsubseteq \text{parent} \sqcap \forall \text{hasChild}.\text{happy}$
	happyStudent $\doteq \text{student} \sqcap \text{happy}$

After elimination of primitive concept definitions we obtain:

TBox	male $\doteq \neg \text{female} \sqcap \text{male}^*$
	parent $\doteq \text{person} \sqcap \exists \text{hasChild}.\text{person}$
	happyParent $\doteq \text{parent} \sqcap \forall \text{hasChild}.\text{happy} \sqcap \text{happyParent}^*$
	happyStudent $\doteq \text{student} \sqcap \text{happy}$

- p.7

## Elimination of primitive concept definitions: Intuition



- If a terminological interpretation  $\mathcal{I}_1 = (\Delta_1, \cdot^{\mathcal{I}_1})$  satisfies  $A \sqsubseteq C$ , then  $A^{\mathcal{I}_1} \subseteq C^{\mathcal{I}_1}$ .
- If a terminological interpretation  $\mathcal{I}_2 = (\Delta_2, \cdot^{\mathcal{I}_2})$  satisfies  $A \doteq C \sqcap A^*$ , then  $A^{\mathcal{I}_2} = C^{\mathcal{I}_2} \cap A^{*\mathcal{I}_2}$ .

For the transformation from  $A \sqsubseteq C$  to  $A \doteq C \sqcap A^*$  to be admissible,  $A \sqsubseteq C$  has to be satisfiable iff  $A \doteq C \sqcap A^*$  is satisfiable.

- p.8

## Elimination of primitive concept definitions: Lemma (1)

### Lemma 16.1

$A \sqsubseteq C$  is satisfiable iff  $A \doteq C \sqcap A^*$  is satisfiable

Proof

- Let  $\mathcal{I}_1 = (\Delta, \cdot^{\mathcal{I}_1})$  be an interpretation over  $(\mathbf{O}, \mathbf{C}, \mathbf{R})$  satisfying  $A \sqsubseteq C$ . Define  $\mathcal{I}_2 = (\Delta, \cdot^{\mathcal{I}_2})$  over  $(\mathbf{O}, \mathbf{C} \cup \{A^*\}, \mathbf{R})$  by  $B^{\mathcal{I}_2} = B^{\mathcal{I}_1}$  for every  $B \in \mathbf{C}$  and  $A^{*\mathcal{I}_2} = A^{\mathcal{I}_1}$ . Since  $C$  does not contain  $A^*$ ,  $C^{\mathcal{I}_2} = C^{\mathcal{I}_1}$ . Since  $\mathcal{I}_1$  satisfies  $A \sqsubseteq C$  and  $A^{\mathcal{I}_2} = A^{\mathcal{I}_1}$ ,  $A^{\mathcal{I}_2} \subseteq C^{\mathcal{I}_2}$ . So,  $C^{\mathcal{I}_2} \cap A^{\mathcal{I}_2} = A^{\mathcal{I}_2} = C^{\mathcal{I}_2} \cap A^{*\mathcal{I}_2} = (C \sqcap A^*)^{\mathcal{I}_2}$ . Thus,  $(\Delta, \cdot^{\mathcal{I}_2})$  satisfies  $A \doteq C \sqcap A^*$ .

- p.9

## Elimination of primitive concept definitions: Lemma (2)

Proof (continued)

- Let  $\mathcal{I}_2 = (\Delta, \cdot^{\mathcal{I}_2})$  over  $(\mathbf{O}, \mathbf{C} \cup \{A^*\}, \mathbf{R})$  satisfying  $A \doteq C \sqcap A^*$ . Define  $\mathcal{I}_1 = (\Delta, \cdot^{\mathcal{I}_1})$  over  $(\mathbf{O}, \mathbf{C}, \mathbf{R})$  by  $B^{\mathcal{I}_1} = B^{\mathcal{I}_2}$  for every  $B \in \mathbf{C}$ . Since  $\mathcal{I}_2$  satisfies  $A \doteq C \sqcap A^*$ ,  $A^{\mathcal{I}_2} \subseteq C^{\mathcal{I}_2}$  and  $A^{\mathcal{I}_2} \subseteq A^{*\mathcal{I}_2}$ . Since  $C$  does not contain  $A^*$  and by definition of  $\mathcal{I}_1$ ,  $A^{\mathcal{I}_2} = A^{\mathcal{I}_1} \subseteq C^{\mathcal{I}_1} = C^{\mathcal{I}_2}$ . Thus,  $\mathcal{I}_1$  satisfies  $A \sqsubseteq C$ .

- p.10

## Elimination of primitive concept definitions: Theorem

### Theorem 16.1

Let  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$  be an acyclic knowledge base.  
Then  $\mathcal{K}^*$  is satisfiable iff  $\mathcal{K}$  is satisfiable.

- p.11

## Elimination of primitive concept definitions: Proof (1)

Proof

We show by induction on the number  $n$  of primitive concept definitions that are eliminated that if  $\mathcal{K}_n$  is obtained from  $\mathcal{K}$  by an  $n$  eliminations then  $\mathcal{K}_n$  is satisfiable iff  $\mathcal{K}$  is satisfiable. Since  $\mathcal{K}^*$  is obtained from  $\mathcal{K}$  by a finite number of such eliminations, the result follows.

Induction base:

$\mathcal{K}$  contains no primitive concept definition and consequently, no primitive concept definition is eliminated. Then  $\mathcal{K}_0 = \mathcal{K}$  and obviously,  $\mathcal{K}_0$  is satisfiable iff  $\mathcal{K}$  is satisfiable.

- p.12

## Elimination of primitive concept definitions: Proof (2)

Induction hypothesis:

If  $\mathcal{K}_n$  is obtained from  $\mathcal{K}$  by eliminating  $n$  primitive concept definitions then  $\mathcal{K}_n$  is satisfiable iff  $\mathcal{K}$  is satisfiable.

Induction step:

Show that the induction hypothesis implies that if  $\mathcal{K}_{n+1}$  is obtained from  $\mathcal{K}$  by eliminating  $n + 1$  primitive concept definitions, then  $\mathcal{K}_{n+1}$  is satisfiable iff  $\mathcal{K}$  is satisfiable.  $\mathcal{K}_{n+1}$  is obtained from  $\mathcal{K}_n$  by eliminating one particular primitive concept definition  $A \sqsubseteq C$  and replacing it by  $A \sqsubseteq C \sqcap A^*$ . By Lemma 16.1,  $\mathcal{K}_{n+1}$  is satisfiable iff  $\mathcal{K}_n$  is satisfiable. By induction hypothesis,  $\mathcal{K}_n$  is satisfiable iff  $\mathcal{K}$  is satisfiable. Both together imply that  $\mathcal{K}_{n+1}$  is satisfiable iff  $\mathcal{K}$  is satisfiable.

- p.13

## Expansion of a TBox

- Let  $\mathcal{T}$  be an acyclic TBox without primitive concept definitions
- The expansion of  $\mathcal{T}$  is formed by the following process of unfolding:
  - For every concept definition  $A \sqsubseteq C$  in  $\mathcal{T}$ , substitute every occurrence of the concept symbol  $A$  on the right-hand side of any concept definition in  $\mathcal{T}$  by  $C$ .
- An acyclic TBox  $\mathcal{T}$  without primitive concept definitions is expanded iff no concept symbol occurring on the left-hand side of a concept definition in  $\mathcal{T}$  occurs on the right-hand side of any concept definition in  $\mathcal{T}$ .

- p.14

## Expansion of a TBox: Example (1)

We start with the following TBox:

TBox	male $\sqsubseteq$ $\neg$ female $\sqcap$ male*
	parent $\sqsubseteq$ person $\sqcap$ $\exists$ hasChild.person
	father $\sqsubseteq$ parent $\sqcap$ male
	grandParent $\sqsubseteq$ person $\sqcap$ $\exists$ hasChild.parent

After unfolding occurrences of male we obtain:

TBox	male $\sqsubseteq$ $\neg$ female $\sqcap$ male*
	parent $\sqsubseteq$ person $\sqcap$ $\exists$ hasChild.person
	father $\sqsubseteq$ parent $\sqcap$ ( $\neg$ female $\sqcap$ male*)
	grandParent $\sqsubseteq$ person $\sqcap$ $\exists$ hasChild.parent

- p.15

## Expansion of a TBox: Example (1)

So, we now have:

TBox	male $\sqsubseteq$ $\neg$ female $\sqcap$ male*
	parent $\sqsubseteq$ person $\sqcap$ $\exists$ hasChild.person
	father $\sqsubseteq$ parent $\sqcap$ ( $\neg$ female $\sqcap$ male*)
	grandParent $\sqsubseteq$ person $\sqcap$ $\exists$ hasChild.parent

After unfolding occurrences of parent we obtain:

TBox	male $\sqsubseteq$ $\neg$ female $\sqcap$ male*
	parent $\sqsubseteq$ person $\sqcap$ $\exists$ hasChild.person
	father $\sqsubseteq$ (person $\sqcap$ $\exists$ hasChild.person) $\sqcap$ ( $\neg$ female $\sqcap$ male*)
	grandParent $\sqsubseteq$ person $\sqcap$ $\exists$ hasChild.(person $\sqcap$ $\exists$ hasChild.person)

- p.15

## Expansion of a TBox: Example (1)

So, we now have:

TBox	male $\sqsubseteq$ $\neg$ female $\sqcap$ male*
	parent $\sqsubseteq$ person $\sqcap$ $\exists$ hasChild.person
	father $\sqsubseteq$ (person $\sqcap$ $\exists$ hasChild.person) $\sqcap$ ( $\neg$ female $\sqcap$ male*)
	grandParent $\sqsubseteq$ person $\sqcap$ $\exists$ hasChild.(person $\sqcap$ $\exists$ hasChild.person)

father and grandparent do not occur on the right-hand side of any concept definition. Thus, unfolding occurrences of father and grandparent on the right-hand sides of concept definitions does not change the TBox.

- p.15

## Expansion of a TBox: Example (1)

The expanded TBox is:

TBox	male $\sqsubseteq$ $\neg$ female $\sqcap$ male*
	parent $\sqsubseteq$ person $\sqcap$ $\exists$ hasChild.person
	father $\sqsubseteq$ (person $\sqcap$ $\exists$ hasChild.person) $\sqcap$ ( $\neg$ female $\sqcap$ male*)
	grandParent $\sqsubseteq$ person $\sqcap$ $\exists$ hasChild.(person $\sqcap$ $\exists$ hasChild.person)

- p.15

## Expansion of a TBox: Example (2)

Consider the following TBox:

TBox	male $\doteq$ $\neg$ female $\sqcap$ male*
	parent $\doteq$ person $\sqcap$ $\exists$ hasChild.person
	happyParent $\doteq$ parent $\sqcap$ $\forall$ hasChild.happy $\sqcap$ happyParent*
	happyStudent $\doteq$ student $\sqcap$ happy

Only the defined concept `parent` occurs on the right-hand side of a concept definition in this TBox. Unfolding `parent` results in:

TBox	male $\doteq$ $\neg$ female $\sqcap$ male*
	parent $\doteq$ person $\sqcap$ $\exists$ hasChild.person
	happyParent $\doteq$ (person $\sqcap$ $\exists$ hasChild.person) $\sqcap$ $\forall$ hasChild.happy $\sqcap$ happyParent*
	happyStudent $\doteq$ student $\sqcap$ happy

- p.16

## Exercise 11

Let the TBox  $\mathcal{T}$  be given by the following concept definitions:

mscCourse	$\doteq$ course $\sqcap$ $\exists$ consistsOf.mscMods
mscStudent	$\doteq$ student $\sqcap$ $\exists$ enrolledOn.mscCourse
phdStudent	$\doteq$ student $\sqcap$ $\exists$ studiesFor.phdDegree
pgStudent	$\doteq$ mscStudent $\sqcup$ phdStudent

Give the expansion of  $\mathcal{T}$ .

- p.17

## Exercise 11: Answer

mscCourse	$\doteq$ course $\sqcap$ $\exists$ consistsOf.mscMods
mscStudent	$\doteq$ student $\sqcap$ $\exists$ enrolledOn.mscCourse
phdStudent	$\doteq$ student $\sqcap$ $\exists$ studiesFor.phdDegree
pgStudent	$\doteq$ mscStudent $\sqcup$ phdStudent

The expansion of  $\mathcal{T}$  is given by:

mscCourse	$\doteq$ course $\sqcap$ $\exists$ consistsOf.mscMods
mscStudent	$\doteq$ student $\sqcap$ $\exists$ enrolledOn.(course $\sqcap$ $\exists$ consistsOf.mscMods)
phdStudent	$\doteq$ student $\sqcap$ $\exists$ studiesFor.phdDegree
pgStudent	$\doteq$ (student $\sqcap$ $\exists$ enrolledOn.(course $\sqcap$ $\exists$ consistsOf.mscMods)) $\sqcup$ (student $\sqcap$ $\exists$ studiesFor.phdDegree)

- p.18

## Exercise 12

Let the TBox  $\mathcal{T}$  be given by the following concept definitions:

male	$\doteq$ $\neg$ female $\sqcap$ male*
parent	$\doteq$ mother $\sqcup$ father
happyParent	$\doteq$ parent $\sqcap$ $\forall$ hasChild.happy $\sqcap$ happyParent*
grandParent	$\doteq$ parent $\sqcap$ $\exists$ hasChild.parent

Give the expansion of  $\mathcal{T}$ .

- p.19

## Exercise 12: Answer

Let the TBox  $\mathcal{T}$  be given by the following concept definitions:

male	$\doteq$ $\neg$ female $\sqcap$ male*
parent	$\doteq$ mother $\sqcup$ father
happyParent	$\doteq$ parent $\sqcap$ $\forall$ hasChild.happy $\sqcap$ happyParent*
grandParent	$\doteq$ parent $\sqcap$ $\exists$ hasChild.parent

The expansion of  $\mathcal{T}$  is given by:

male	$\doteq$ $\neg$ female $\sqcap$ male*
parent	$\doteq$ mother $\sqcup$ father
happyParent	$\doteq$ (mother $\sqcup$ father) $\sqcap$ $\forall$ hasChild.happy $\sqcap$ happyParent*
grandParent	$\doteq$ (mother $\sqcup$ father) $\sqcap$ $\exists$ hasChild.(mother $\sqcup$ father)

- p.20

## Expansion of a TBox: Lemma (1)

### Lemma 16.2

Let  $\mathcal{T}$  be an acyclic TBox  $\mathcal{T}$ .

1. The process of unfolding the TBox  $\mathcal{T}$  terminates.
2. The result of the process of unfolding a unique TBox.
3. The result of the process of unfolding the TBox  $\mathcal{T}$  is an expanded TBox.

### Proof

The process of unfolding *terminates* since in an acyclic TBox  $\mathcal{T}$  each unfolding step which replaces a concept symbol  $A$  by  $C$  reduces the number of occurrences of  $A$  in  $\mathcal{T}$ . Eventually the number of occurrences of any defined concept symbol on the right-hand side of concept definitions will be zero. Therefore, the resulting TBox is expanded.

- p.21

## Expansion of a TBox: Lemma (2)

### Lemma 16.3

Let  $\mathcal{T}_0 = \{A \doteq C, B \doteq D\} \cup \mathcal{T}'$  be a TBox over a signature  $\Sigma$ .

Let  $B \doteq D$  be a concept definition such that the concept  $D$  contains an occurrence of  $A$ .

Let  $E$  be the concept obtained by replacing the occurrence of  $A$  by  $C$ .

Then  $\mathcal{T}_0$  is satisfiable iff  $\{A \doteq C, B \doteq E\} \cup \mathcal{T}'$  is satisfiable.

### Proof

Let  $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$  be a terminological interpretation of the signature  $\Sigma$ . We show that  $\mathcal{I}$  satisfies  $\mathcal{T}_0$  iff  $\mathcal{I}$  satisfies  $\mathcal{T}_1 = \{A \doteq C, B \doteq E\} \cup \mathcal{T}'$ . To this end it is sufficient to show that  $D^{\mathcal{I}} = E^{\mathcal{I}}$ , which can be done by induction on the structure of  $D$ .

- p.22

## Expansion of a TBox: Theorem

### Theorem 16.2

Let  $\mathcal{T}$  be an acyclic TBox without primitive concept definitions over a signature  $\Sigma$  and let  $\mathcal{T}'$  be the expansion of  $\mathcal{T}$  obtained by the process of unfolding. Then  $\mathcal{T}$  is satisfiable iff  $\mathcal{T}'$  is satisfiable.

### Proof

Let  $\mathcal{I}$  be a terminological interpretation over  $\Sigma$ . Let  $\mathcal{I}_n$  be obtained from  $\mathcal{I}$  by  $n$  unfolding steps, that is,  $n$  steps in which one occurrence of a defined concept symbol  $A$  is replaced by a concept  $C$  based on some concept definition  $A \doteq C$  in  $\mathcal{I}$ . We can show by induction on  $n$  that  $\mathcal{I}$  satisfies  $\mathcal{T}$  iff  $\mathcal{I}$  satisfies  $\mathcal{I}_n$ . Since after some finite number of unfolding steps we obtain some TBox  $\mathcal{I}_k$  such that  $\mathcal{I}_k = \mathcal{T}'$ , the desired result follows.

- p.23

## Elimination of defined concepts from an ABox (1)

- Let  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$  be a knowledge base such that  $\mathcal{T}$  is an acyclic TBox without primitive concept definitions.
- The **expansion** of  $\mathcal{K}$  is formed in two steps:
  1. Compute the **expansion**  $\mathcal{T}'$  of  $\mathcal{T}$
  2. For every concept definition  $A \doteq C$  in  $\mathcal{T}'$ , substitute every occurrence of the concept symbol  $A$  in any concept assertion in  $\mathcal{A}$  by  $C$ .  
The resulting ABox  $\mathcal{A}'$  is the **expansion** of  $\mathcal{A}$ .
- A knowledge base  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$  is **expanded** iff (i)  $\mathcal{T}$  is an expanded TBox and (ii) no concept assertion in  $\mathcal{A}$  contains a concept symbol occurring on the left-hand side of any concept definition in  $\mathcal{T}$ .

– p.24

## Expansion of a knowledge base: Example

We start with the following knowledge base  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ :

TBox	male $\doteq$ $\neg$ female $\sqcap$ male*
	parent $\doteq$ person $\sqcap$ $\exists$ hasChild.person
	happyParent $\doteq$ parent $\sqcap$ $\forall$ hasChild.happy $\sqcap$ happyParent*
	happyStudent $\doteq$ student $\sqcap$ happy
ABox	tim : happyParent
	sue : student            (tim, sue) : hasChild
	sue : $\neg$ happyStudent

– p.25

## Expansion of a knowledge base: Example

We have already determined that the **expansion** of the TBox

TBox	male $\doteq$ $\neg$ female $\sqcap$ male*
	parent $\doteq$ person $\sqcap$ $\exists$ hasChild.person
	happyParent $\doteq$ parent $\sqcap$ $\forall$ hasChild.happy $\sqcap$ happyParent*
	happyStudent $\doteq$ student $\sqcap$ happy

is given by

TBox	male $\doteq$ $\neg$ female $\sqcap$ male*
	parent $\doteq$ person $\sqcap$ $\exists$ hasChild.person
	happyParent $\doteq$ (person $\sqcap$ $\exists$ hasChild.person) $\sqcap$ $\forall$ hasChild.happy $\sqcap$ happyParent*
	happyStudent $\doteq$ student $\sqcap$ happy

– p.26

## Expansion of a knowledge base: Example

So, for the second step of the expansion of  $\mathcal{K}$  we start with

TBox	male $\doteq$ $\neg$ female $\sqcap$ male*
	parent $\doteq$ person $\sqcap$ $\exists$ hasChild.person
	happyParent $\doteq$ (person $\sqcap$ $\exists$ hasChild.person) $\sqcap$ $\forall$ hasChild.happy $\sqcap$ happyParent*
	happyStudent $\doteq$ student $\sqcap$ happy
ABox	tim : happyParent
	sue : student            (tim, sue) : hasChild
	sue : $\neg$ happyStudent

– p.27

## Expansion of a knowledge base: Example

Replacing **happyParent** in the ABox results in:

TBox	male $\doteq$ $\neg$ female $\sqcap$ male*
	parent $\doteq$ person $\sqcap$ $\exists$ hasChild.person
	happyParent $\doteq$ (person $\sqcap$ $\exists$ hasChild.person) $\sqcap$ $\forall$ hasChild.happy $\sqcap$ happyParent*
	happyStudent $\doteq$ student $\sqcap$ happy
ABox	tim : (person $\sqcap$ $\exists$ hasChild.person) $\sqcap$ $\forall$ hasChild.happy $\sqcap$ happyParent*
	sue : student
	sue : student            (tim, sue) : hasChild
	sue : $\neg$ happyStudent

– p.28

## Expansion of a knowledge base: Example

Replacing **happyStudent** in the ABox results in:

TBox	male $\doteq$ $\neg$ female $\sqcap$ male*
	parent $\doteq$ person $\sqcap$ $\exists$ hasChild.person
	happyParent $\doteq$ (person $\sqcap$ $\exists$ hasChild.person) $\sqcap$ $\forall$ hasChild.happy $\sqcap$ happyParent*
	happyStudent $\doteq$ student $\sqcap$ happy
ABox	tim : (person $\sqcap$ $\exists$ hasChild.person) $\sqcap$ $\forall$ hasChild.happy $\sqcap$ happyParent*
	sue : student            (tim, sue) : hasChild
	sue : $\neg$ (student $\sqcap$ happy)

No further unfolding steps can be performed. So, the knowledge base above is the **expansion** of  $\mathcal{K}$ .

– p.29

## Expansion of a knowledge base: Lemma

### Lemma 16.4

Let  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$  be a knowledge base such that  $\mathcal{T}$  is an acyclic TBox without primitive concept definitions.

1. The process of computing the expansion of  $\mathcal{K}$  **terminates**.
2. The expansion of  $\mathcal{K}$  is a **unique** knowledge base.
3. The expansion of  $\mathcal{K}$  is an **expanded** knowledge base.

**Proof**

Along the lines of the proof of Lemma 16.2.

– p.30

## Expansion of a knowledge base: Theorem

### Theorem 16.3

Let  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$  be a knowledge base such that  $\mathcal{T}$  is an acyclic TBox without primitive concept definitions over a signature  $\Sigma$  and let  $\mathcal{K}'$  be the expansion of  $\mathcal{K}$ .

Then  $\mathcal{K}$  is **satisfiable** iff  $\mathcal{K}'$  is **satisfiable** iff  $\mathcal{A}$  is **satisfiable**.

**Proof**

Along the lines of the proof of Theorem 16.2

– p.31

## Expansion: Summary

- Our aim is to present a simple approach to solving the problem of  
Consistency of an ABox (with respect to a TBox)
- So far we have seen three steps of this approach
  1. Elimination of primitive concept definitions from the TBox
  2. Expansion of the TBox
  3. Elimination of defined concepts from the ABox
- By Theorem 16.3 we can forget the TBox of a knowledge base after the third step of this approach.

- p.32

## Summary

We have considered the first three steps of a simple procedure for solving the problem of

Consistency of an ABox (with respect to a TBox)

which proceeds in five steps:

1. Elimination of primitive concept definitions from the TBox
2. Expansion of the TBox
3. Elimination of defined concepts from the ABox
4. Transforming the ABox into negation normal form
5. Application of completion rules to the ABox

- p.33

# Knowledge Representation and Reasoning Part 1: Modal and Description Logics

Wiebe van der Hoek

## Lecture 17: Description logic (5)

## Last time...

We have considered the first three steps of a simple procedure for solving the problem of

Consistency of an ABox (with respect to a TBox)

which proceeds in five steps:

1. Elimination of primitive concept definitions from the TBox
2. Expansion of the TBox
3. Elimination of defined concepts from the ABox
4. Transforming the ABox into negation normal form
5. Application of completion rules to the ABox

- p.1

## Negation normal form (1)

- Let  $C$  be an arbitrary concept of  $\mathcal{ALC}$ .  
Then  $C$  is in negation normal form iff for any subconcept  $\neg D$  of  $C$ ,  $D$  is a concept symbol.
- The negation normal form of a concept  $C$  can be computed by applying the function  $\text{nnf}$  to  $C$  where  $\text{nnf}$  is inductively defined as follows:

$$\begin{aligned}\text{nnf}(\neg T) &= \perp \\ \text{nnf}(\neg \perp) &= T \\ \text{nnf}(A) &= A \quad \text{for any concept symbol } A \\ \text{nnf}(\neg A) &= \neg A \quad \text{for any concept symbol } A \\ \text{nnf}(\neg\neg C) &= \text{nnf}(C)\end{aligned}$$

- p.2

## Negation normal form (2)

$$\begin{aligned}\text{nnf}(\neg(C \sqcup D)) &= \text{nnf}(\neg C) \sqcap \text{nnf}(\neg D) \\ \text{nnf}(C \sqcup D) &= \text{nnf}(C) \sqcup \text{nnf}(D) \\ \text{nnf}(\neg(C \sqcap D)) &= \text{nnf}(\neg C) \sqcup \text{nnf}(\neg D) \\ \text{nnf}(C \sqcap D) &= \text{nnf}(C) \sqcap \text{nnf}(D) \\ \text{nnf}(\neg\forall R.C) &= \exists R. \text{nnf}(\neg C) \\ \text{nnf}(\forall R.C) &= \forall R. \text{nnf}(C) \\ \text{nnf}(\neg\exists R.C) &= \forall R. \text{nnf}(\neg C) \\ \text{nnf}(\exists R.C) &= \exists R. \text{nnf}(C)\end{aligned}$$

- p.3

## Negation normal form (3)

- Alternatively, we can describe  $\text{nnf}$  by a simple rewrite relation on concepts defined by the following rewrite rules:

$$\begin{aligned}\neg\neg C &\rightarrow_{\text{nnf}} C \\ \neg T &\rightarrow_{\text{nnf}} \perp & \neg\perp &\rightarrow_{\text{nnf}} T \\ \neg(C \sqcup D) &\rightarrow_{\text{nnf}} \neg C \sqcap \neg D & \neg(C \sqcap D) &\rightarrow_{\text{nnf}} \neg C \sqcup \neg D \\ \neg\forall R.C &\rightarrow_{\text{nnf}} \exists R.\neg C & \neg\exists R.C &\rightarrow_{\text{nnf}} \forall R.\neg C\end{aligned}$$

Intuitively, the rules tell you to replace any occurrence of the left-hand side of a rule by the corresponding occurrence of the right-hand side of the rule wherever it occurs, that is even if the occurrence is deep inside a concept.

- p.4

## Negation normal form (4)

- Then  $\text{nnf}(C)$  denotes the normal form of  $C$  under this rewrite relation, that is, the concept obtained from  $C$  after no further application of the rules to  $C$  is possible.
- A knowledge base  $\mathcal{K}$  is in negation normal form iff every concept  $C$  in  $\mathcal{K}$  is in negation normal form
- Similarly, an ABox  $\mathcal{A}$  is in negation normal form iff every concept  $C$  in  $\mathcal{A}$  is in negation normal form
- The fourth step of our procedure for solving the problem of  
Consistency of an ABox (with respect to a TBox)  
consists of transforming the ABox of a knowledge base into negation normal form

- p.5

## Negation normal form: Example (1)

Consider the concept  $\neg(\text{student} \sqcap \text{happy})$ .

Then

$$\begin{aligned} \text{nnf}(\neg(\text{student} \sqcap \text{happy})) \\ = \neg\text{student} \sqcup \neg\text{happy} \end{aligned}$$

Consider the concept  $\neg(\text{mother} \sqcup (\neg\text{female} \sqcap \exists\text{hasChild.person}))$ .

Then

$$\begin{aligned} \text{nnf}(\neg(\text{mother} \sqcup (\neg\text{female} \sqcap \exists\text{hasChild.person}))) \\ = \text{nnf}(\neg\text{mother}) \sqcap \text{nnf}(\neg(\neg\text{female} \sqcap \exists\text{hasChild.person})) \\ = \neg\text{mother} \sqcap \text{nnf}(\neg(\neg\text{female} \sqcap \exists\text{hasChild.person})) \\ = \neg\text{mother} \sqcap (\text{nnf}(\neg\neg\text{female}) \sqcup \text{nnf}(\neg\exists\text{hasChild.person})) \\ = \neg\text{mother} \sqcap (\text{female} \sqcup \text{nnf}(\neg\exists\text{hasChild.person})) \\ = \neg\text{mother} \sqcap (\text{female} \sqcup \forall\text{hasChild.nnf}(\neg\text{person})) \\ = \neg\text{mother} \sqcap (\text{female} \sqcup \forall\text{hasChild.}\neg\text{person}) \end{aligned}$$

- p.6

## Negation normal form: Example (1)

Consider the concept  $\neg(\text{student} \sqcap \text{happy})$ .

Then

$$\begin{aligned} (\neg(\text{student} \sqcap \text{happy})) \\ \rightarrow_{\text{nnf}} \neg\text{student} \sqcup \neg\text{happy} \end{aligned}$$

Consider the concept  $\neg(\text{mother} \sqcup (\neg\text{female} \sqcap \exists\text{hasChild.person}))$ .

Then

$$\begin{aligned} \neg(\text{mother} \sqcup (\neg\text{female} \sqcap \exists\text{hasChild.person})) \\ \rightarrow_{\text{nnf}} \neg\text{mother} \sqcap \neg(\neg\text{female} \sqcap \exists\text{hasChild.person}) \\ \rightarrow_{\text{nnf}} \neg\text{mother} \sqcap ((\neg\neg\text{female}) \sqcup (\neg\exists\text{hasChild.person})) \\ \rightarrow_{\text{nnf}} \neg\text{mother} \sqcap (\text{female} \sqcup (\neg\exists\text{hasChild.person})) \\ \rightarrow_{\text{nnf}} \neg\text{mother} \sqcap (\text{female} \sqcup \forall\text{hasChild.}\neg\text{person}) \end{aligned}$$

- p.7

## Exercise 13

Let the knowledge base  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$  be given by

TBox	$\text{cpu} \sqsubseteq \text{amd} \sqcup \text{intel}$	$\text{pc} \doteq \exists\text{hasPart.cpu}$
	$\text{pcLab} \doteq \text{room} \sqcap \forall\text{hasEquip.pc}$	
ABox	$\text{t10} : \neg\text{pcLab}$	$\text{t10} : \text{room}$

Eliminate all primitive concept definitions, expand the knowledge base, and transform all concepts into negation normal form.

- p.8

## Exercise 13: Answer

TBox	$\text{cpu} \sqsubseteq \text{amd} \sqcup \text{intel}$	$\text{pc} \doteq \exists\text{hasPart.cpu}$
	$\text{pcLab} \doteq \text{room} \sqcap \forall\text{hasEquip.pc}$	
ABox	$\text{t10} : \neg\text{pcLab}$	$\text{t10} : \text{room}$

Eliminate all primitive concept definitions, expand the knowledge base, and transform all concepts into negation normal form.

TBox	$\text{cpu} \doteq (\text{amd} \sqcup \text{intel}) \sqcap \text{cpu}^*$
	$\text{pc} \doteq \exists\text{hasPart}.\text{((amd} \sqcup \text{intel}) \sqcap \text{cpu}^*)$
	$\text{pcLab} \doteq \text{room} \sqcap \forall\text{hasEquip}.\text{(\exists\text{hasPart}.\text{((amd} \sqcup \text{intel}) \sqcap \text{cpu}^*)})}$
ABox	$\text{t10} : \neg\text{room} \sqcup \exists\text{hasEquip}.\forall\text{hasPart}.\text{((}\neg\text{amd} \sqcap \neg\text{intel}) \sqcup \neg\text{cpu}^*)}$
	$\text{t10} : \text{room}$

- p.9

## Tableaux calculi (1)

- The final steps in our procedure for solving the problem of Consistency of an ABox (with respect to a TBox) consist of the application of completion rules to the ABox
- The completion rules can be seen as inference rules of a tableaux calculus

- p.10

## Tableaux calculi (2)

- A tableaux calculus is a formal proof procedure, existing in many varieties and for several logics, but always with certain characteristics:
  - It is a refutation system: Given an initial constraint system or tableau  $\Delta$ , it tries to show that  $\Delta$  is unsatisfiable.
  - It proceeds by breaking down  $\Delta$  into several tableaux  $\Delta_1, \dots, \Delta_n$  such that  $\Delta$  is unsatisfiable iff all of the  $\Delta_i$ ,  $1 \leq i \leq n$ , are unsatisfiable (tableau expansion stage.)
  - Finally, there are rules for closing a tableau: unsatisfiability conditions based on the syntactical form of a tableau.

- p.11

## Tableaux calculi (3)

- In our procedure for solving the problem of Consistency of an ABox (with respect to a TBox)
  - our initial constraint system is simply the ABox  $\mathcal{A}^*$
  - given a constraint system  $\Delta$  our completion rules will generate one or more constraint systems  $\Delta_i$ ,  $1 \leq i \leq n$
  - the completion rules are intended to preserve satisfiability, that is, if we apply a completion rule to  $\Delta$ , then  $\Delta$  is unsatisfiable iff all the resulting constraint systems are unsatisfiable

- p.12

## Tableaux calculi: Keeping track

- Assume the following:
  - We start with a constraint system  $\Delta_1$  and by applying a completion rule we get  $\Delta_2$  and  $\Delta_3$
  - By applying a completion rule to  $\Delta_2$  we get  $\Delta_4$  and  $\Delta_5$
  - Finally, by applying a completion rule to  $\Delta_3$  we get  $\Delta_6$ ,  $\Delta_7$ , and  $\Delta_8$
- Question: How do we keep track of what is going on here?
- Answer: Either by trees or sets of sequences of constraint systems

- p.13

### Tableaux calculi: Keeping track (trees)

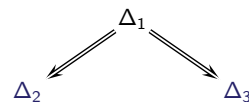
We start with the constraint system  $\Delta_1$ :

$$\Delta_1$$

- p.14

### Tableaux calculi: Keeping track (trees)

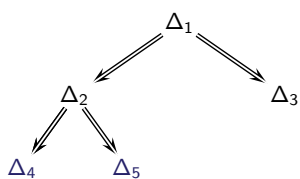
Then we obtain  $\Delta_2$  and  $\Delta_3$  from  $\Delta_1$ :



- p.14

### Tableaux calculi: Keeping track (trees)

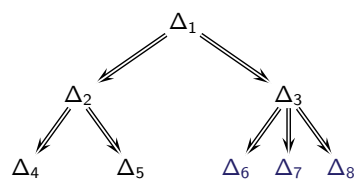
In the next step we obtain  $\Delta_4$  and  $\Delta_5$  from  $\Delta_2$ :



- p.14

### Tableaux calculi: Keeping track (trees)

Finally, we obtain  $\Delta_6$ ,  $\Delta_7$  and  $\Delta_8$  from  $\Delta_3$ :



- p.14

### Tableaux calculi: Keeping track (sets of sequences)

We start with the constraint system  $\Delta_1$ :

$$\Delta_1$$

- p.15

### Tableaux calculi: Keeping track (sets of sequences)

Then we obtain  $\Delta_2$  and  $\Delta_3$  from  $\Delta_1$ :

$$\Delta_1 \rightarrow \begin{matrix} \Delta_1 \Rightarrow \Delta_2 \\ \Delta_1 \Rightarrow \Delta_3 \end{matrix}$$

- p.15

### Tableaux calculi: Keeping track (sets of sequences)

In the next step we obtain  $\Delta_4$  and  $\Delta_5$  from  $\Delta_2$ :

$$\Delta_1 \rightarrow \begin{matrix} \Delta_1 \Rightarrow \Delta_2 \\ \Delta_1 \Rightarrow \Delta_3 \end{matrix} \rightarrow \begin{matrix} \Delta_1 \Rightarrow \Delta_2, & \Delta_2 \Rightarrow \Delta_4 \\ & \Delta_2 \Rightarrow \Delta_5 \\ \Delta_1 \Rightarrow \Delta_3 \end{matrix}$$

- p.15

### Tableaux calculi: Keeping track (sets of sequences)

Finally, we obtain  $\Delta_6$ ,  $\Delta_7$  and  $\Delta_8$  from  $\Delta_3$ :

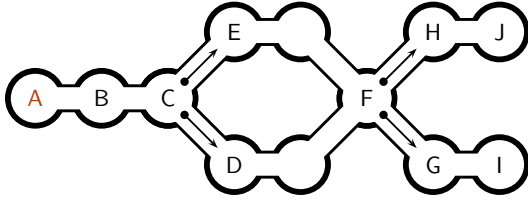
$$\Delta_1 \rightarrow \begin{matrix} \Delta_1 \Rightarrow \Delta_2 \\ \Delta_1 \Rightarrow \Delta_3 \end{matrix} \rightarrow \begin{matrix} \Delta_1 \Rightarrow \Delta_2, & \Delta_2 \Rightarrow \Delta_4 \\ & \Delta_2 \Rightarrow \Delta_5 \\ \Delta_1 \Rightarrow \Delta_3 \end{matrix} \downarrow \begin{matrix} \Delta_1 \Rightarrow \Delta_2, & \Delta_2 \Rightarrow \Delta_4 \\ & \Delta_2 \Rightarrow \Delta_5 \\ \Delta_1 \Rightarrow \Delta_3, & \Delta_3 \Rightarrow \Delta_6 \\ & \Delta_3 \Rightarrow \Delta_7 \\ & \Delta_3 \Rightarrow \Delta_8 \end{matrix}$$

- p.15



### Deterministic and non-deterministic rules (6)

Suppose you are watching a person at point **A** in the labyrinth below, who wants to get to point **K** (which doesn't exist), he can only go forward or backtrack, and he does not have the overhead view we have.

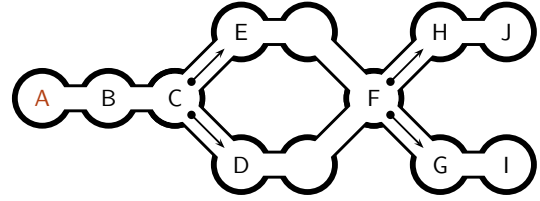


What happens if the person correctly distinguishes between deterministic steps, don't care non-deterministic and don't know non-deterministic steps?

- p.21

### Deterministic and non-deterministic rules (6)

Suppose you are watching a person at point **A** in the labyrinth below, who wants to get to point **K** (which doesn't exist), he can only go forward or backtrack, and he does not have the overhead view we have.

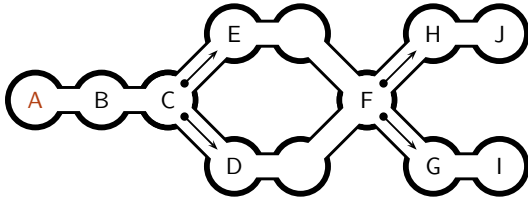


What happens if the person has to assume that all the choices he makes are don't know non-deterministic?

- p.21

### Deterministic and non-deterministic rules (6)

Suppose you are watching a person at point **A** in the labyrinth below, who wants to get to point **K** (which doesn't exist), he can only go forward or backtrack, and he does not have the overhead view we have.



What happens if the person has to assume that underlying each step is a don't know non-deterministic choice?

- p.21

### Summary

- Negation normal form computation
- Tableaux calculus for deciding the consistency of ABoxes (overview)
- Deterministic and non-deterministic rules

- p.22

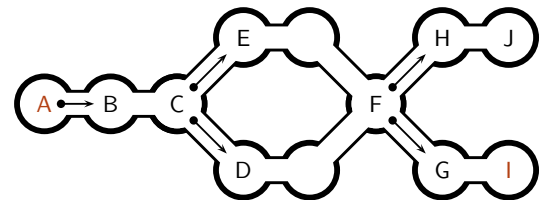
## Knowledge Representation and Reasoning Part 1: Modal and Description Logics

Wiebe van der Hoek

### Lecture 18: Description logic (6)

### Last time ...

- Tableaux calculus for deciding the consistency of ABoxes
- Deterministic and non-deterministic rules



The step from **A** is deterministic.  
The step from **C** is don't care non-deterministic.  
The step from **F** is don't know non-deterministic.

- Completion rules

- p.1

### Completion rules: The AND rule (1)

- The AND rule takes the following form:  

$$\Delta \Rightarrow_{\cap} \Delta \cup \{a : C, a : D\}$$
 if  $a : (C \sqcap D)$  is in  $\Delta$ ,  $a : C$  and  $a : D$  are not both in  $\Delta$ .  
 and we say that the AND rule is applied to  $a : (C \sqcap D)$ .
- Intuition: Let  $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$  be a terminological interpretation. Then  $\mathcal{I}$  satisfies  $\{a : (C \sqcap D)\}$   
 iff  $a^{\mathcal{I}} \in (C \sqcap D)^{\mathcal{I}}$   
 iff  $a^{\mathcal{I}} \in C^{\mathcal{I}}$  and  $a^{\mathcal{I}} \in D^{\mathcal{I}}$   
 iff  $\mathcal{I}$  satisfies  $\{a : C, a : D\}$ .  
 Thus,  $\mathcal{I}$  satisfies  $\Delta \cup \{a : (C \sqcap D)\}$   
 iff  $\mathcal{I}$  satisfies  $\Delta \cup \{a : C, a : D, a : (C \sqcap D)\}$ .

- p.2

### Completion rules: The AND rule (2)

- The side condition 'if  $a : C$  and  $a : D$  are not both in  $\Delta$ ' ensures that the rule is applied at most once for  $a : (C \sqcap D)$ .
- The AND rule is don't care non-deterministic, that is, given a constraint system

$$\Delta = \{a : (C \sqcap D), b : (E \sqcap F)\} \cup \Delta'$$

such that neither both  $a : C$  and  $a : D$  nor both  $b : E$  and  $b : F$  are in  $\Delta$ , it makes no difference whether we apply the rule first to  $a : (C \sqcap D)$  or to  $b : (E \sqcap F)$ .



- p.3

## Understanding completion rules (1)

Question: What do we mean by

$$\Delta \Rightarrow_{\sqcap} \Delta \cup \{a : C, a : D\}$$

if  $a : (C \sqcap D)$  is in  $\Delta$ ,  $a : C$  and  $a : D$  are not both in  $\Delta$ .

There are two views:

1. This is a rule schema standing for all instances of
 
$$\Delta \Rightarrow_{\sqcap} \Delta \cup \{a : C, a : D\}$$
 for all possible constraint systems  $\Delta$ , object symbols  $a$ , and concepts  $C, D$ , satisfying the side condition that  $a : (C \sqcap D)$  is in  $\Delta$ ,  $a : C$  and  $a : D$  are not both in  $\Delta$ .

- p.4

## Understanding completion rules (2)

- Under this view, for example,
 
$$\{b1 : car \sqcap bmw, t1 : car \sqcap audi\} \Rightarrow_{\sqcap}^1$$

$$\{b1 : car, b1 : bmw, b1 : car \sqcap bmw, t1 : car \sqcap audi\}$$

$$\{b1 : car \sqcap bmw, t1 : car \sqcap audi\} \Rightarrow_{\sqcap}^2$$

$$\{t1 : car, t1 : audi, b1 : car \sqcap bmw, t1 : car \sqcap audi\}$$
 are two distinct instances of this rule schema.
- Under this view, each instance of the AND rule is a deterministic rule, since it has exactly one result. However, the choice which instance of the AND rule we apply is don't care non-deterministic. For example, both rule instances above are applicable to  $\{b1 : car \sqcap bmw, t1 : car \sqcap audi\}$ , and we get different, though deterministic, results depending on which rule instance we apply.

- p.5

## Understanding completion rules (3)

Question: What do we mean by

$$\Delta \Rightarrow_{\sqcap} \Delta \cup \{a : C, a : D\}$$

if  $a : (C \sqcap D)$  is in  $\Delta$ ,  $a : C$  and  $a : D$  are not both in  $\Delta$ .

There are two views:

2. This is a meta-level rewrite rule with variables  $\Delta$  for constraint systems,  $a$  for object symbols, and  $C, D$  for concepts. In an application of this rule
  - a. the meta-level variables on the left-hand side of the rule are matched with a particular constraint system, a particular object symbol, and particular concepts;
  - b. it is checked whether the side condition holds;
  - c. the instance of the left-hand side is replaced by the corresponding instance of the right-hand side of the rule.

- p.6

## Understanding completion rules (4)

- Under this view, for example, there are two ways to match the variables  $\Delta, a, C$  and  $D$  onto the constraint system  $\{b1 : car \sqcap bmw, t1 : car \sqcap audi\}$ :
  - ▶  $\Delta = \{b1 : car \sqcap bmw, t1 : car \sqcap audi\}$ ,  
 $a = b1, \quad C = car, \quad D = bmw$
  - ▶  $\Delta = \{b1 : car \sqcap bmw, t1 : car \sqcap audi\}$ ,  
 $a = t1, \quad C = car, \quad D = audi$
 and depending on which matching substitution we use, the application will either result in  $\{b1 : car, b1 : bmw, b1 : car \sqcap bmw, t1 : car \sqcap audi\}$  or  $\{t1 : car, t1 : audi, b1 : car \sqcap bmw, t1 : car \sqcap audi\}$ . Thus, the rule is non-deterministic, and we can show that it is don't care non-deterministic.

- p.7

## The AND rule: Example (1)

- Consider the constraint system
 
$$\Delta_1 = \{tim : (person \sqcap \exists hasChild.person) \sqcap (\forall hasChild.happy \sqcap happyParent^*),$$

$$sue : student, (tim, sue) : hasChild,$$

$$sue : \neg student \sqcup \neg happy\}$$
- By one application of the AND rule we obtain
 
$$\Delta_2 = \{tim : (person \sqcap \exists hasChild.person),$$

$$tim : (\forall hasChild.happy \sqcap happyParent^*)\} \cup \Delta_1,$$
 that is  $\Delta_1 \Rightarrow_{\sqcap} \Delta_2$ .

- p.8

## The AND rule: Example (2)

- Consider the constraint system
 
$$\Delta_2 = \{tim : (person \sqcap \exists hasChild.person),$$

$$tim : (\forall hasChild.happy \sqcap happyParent^*)\} \cup \Delta_1,$$
- We see that an application of the AND rule to  $\Delta_2$  can yield two different results depending on whether it is applied to the first or second constraint (concept assertion) we see above.
- We know that it does not matter which result we pick and that after two applications of the AND rule we obtain:
 
$$\Delta_4 = \{tim : person, tim : \exists hasChild.person,$$

$$tim : \forall hasChild.happy, tim : happyParent^*\} \cup \Delta_2,$$
 that is  $\Delta_2 \Rightarrow_{\sqcap} \Delta_3 \Rightarrow_{\sqcap} \Delta_4$ .

- p.9

## Completion rules: The OR rule (1)

- The OR rule takes the following form:
 
$$\Delta \Rightarrow_{\sqcup} \Delta \cup \{a : E\}$$
 if  $a : (C \sqcup D)$  is in  $\Delta$ , neither  $a : C$  nor  $a : D$  is in  $\Delta$  and  $E = C$  or  $E = D$ . and we say that the OR rule is applied to  $a : (C \sqcup D)$ .
- Intuition: Let  $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$  be a terminological interpretation. Then  $\mathcal{I}$  satisfies  $\{a : (C \sqcup D)\}$ 
 iff  $a^{\mathcal{I}} \in (C \sqcup D)^{\mathcal{I}}$   
 iff  $a^{\mathcal{I}} \in C^{\mathcal{I}}$  or  $a^{\mathcal{I}} \in D^{\mathcal{I}}$   
 iff  $\mathcal{I}$  satisfies  $\{a : C\}$  or  $\mathcal{I}$  satisfies  $\{a : D\}$ .
- Thus,  $\mathcal{I}$  satisfies  $\Delta \cup \{a : (C \sqcup D)\}$ 
 iff  $\mathcal{I}$  satisfies  $\Delta \cup \{a : C, a : (C \sqcup D)\}$  or  $\mathcal{I}$  satisfies  $\Delta \cup \{a : D, a : (C \sqcup D)\}$ .

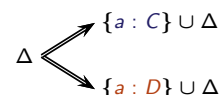
- p.10

## Completion rules: The OR rule (2)

- The side condition 'if neither  $a : C$  nor  $a : D$  is in  $\Delta$ ' ensures that the rule is applied at most once for  $a : (C \sqcup D)$ .
- The choice to which constraint the OR rule is applied is don't care non-deterministic, however, the choice which result we pick don't know non-deterministic, that is, given a constraint system

$$\Delta = \{a : (C \sqcup D)\} \cup \Delta'$$

such that neither  $a : C$  nor  $a : D$  is in  $\Delta$ , we may have to explore two different constraint systems,  $\{a : C\} \cup \Delta'$  and  $\{a : D\} \cup \Delta'$  since we do not know which one of these is the right one.



- p.11

## The OR rule: Example

- Reconsider the constraint system  
 $\Delta_4 = \{\text{sue} : \neg\text{student} \sqcup \neg\text{happy}, \dots\}$
- We see that an application of the OR rule to  $\Delta_4$  can yield two different results,
  - $\Delta_5 = \{\text{sue} : \neg\text{student}\} \cup \Delta_4$  or
  - $\Delta_6 = \{\text{sue} : \neg\text{happy}\} \cup \Delta_4$ ,but we don't know which one we have to pick.
- It follows from our discussion of **don't know non-determinism** that we can either work on both in parallel or focus on one of them first and, if unsuccessful, come back to the other one.

- p.12

## Exercise 13

Consider the constraint system

$$\Delta = \{\text{tom} : \text{student} \sqcap \exists\text{worksAt.company}, \\ \text{jim} : (\exists\text{studiesFor.phdDegree}) \sqcup (\exists\text{worksAt.university})\}$$

1. Can you apply the AND rule to  $\Delta$ ? If so, what is the result?
2. Can you apply the OR rule to  $\Delta$ ? If so, what is the result?
3. If you can apply both the AND rule and the OR rule to a constraint system, does it matter what you do first?

- p.13

## Exercise 13: Answer (1)

Consider the constraint system

$$\Delta = \{\text{tom} : \text{student} \sqcap \exists\text{worksAt.company}, \\ \text{jim} : (\exists\text{studiesFor.phdDegree}) \sqcup (\exists\text{worksAt.university})\}$$

1. Can you apply the AND rule to  $\Delta$ ? If so, what is the result?

Yes, and the result is

$$\Delta_1 = \{\text{tom} : \text{student}, \text{tom} : \exists\text{worksAt.company}\} \cup \Delta$$

That is  $\Delta \Rightarrow_{\sqcap} \Delta_1$

- p.14

## Exercise 13: Answer (2)

Consider the constraint system

$$\Delta = \{\text{tom} : \text{student} \sqcap \exists\text{worksAt.company}, \\ \text{jim} : (\exists\text{studiesFor.phdDegree}) \sqcup (\exists\text{worksAt.university})\}$$

2. Can you apply the OR rule to  $\Delta$ ? If so, what is the result?

Yes, and the result is either

$$\Delta_2 = \{\text{jim} : \exists\text{studiesFor.phdDegree}\} \cup \Delta$$

or

$$\Delta_3 = \{\text{jim} : \exists\text{worksAt.university}\} \cup \Delta$$

That is  $\Delta \Rightarrow_{\sqcup} \Delta_2$  or  $\Delta \Rightarrow_{\sqcup} \Delta_3$

- p.15

## Exercise 13: Answer (3)

Consider the constraint system

$$\Delta = \{\text{tom} : \text{student} \sqcap \exists\text{worksAt.company}, \\ \text{jim} : (\exists\text{studiesFor.phdDegree}) \sqcup (\exists\text{worksAt.university})\}$$

3. If you can apply both the AND rule and the OR rule to a constraint system, does it matter what you do first?

No, it doesn't:

- After an application of the AND rule, the constraint to which we can apply the OR rule is still there
- Analogously, after an application of the OR rule, the constraint to which we can apply the AND rule is still there

- p.16

## Completion rules: The SOME rule (1)

- The SOME rule takes the following form:

$$\Delta \Rightarrow_{\exists} \Delta \cup \{(a, b) : R, b : C\}$$

if  $a : \exists R.C$  is in  $\Delta$ , there is no  $d$  such that both  $(a, d) : R$  and  $d : C$  are in  $\Delta$ , and  $b$  is a new object symbol with respect to  $\Delta$ .

and we say that the SOME rule is applied to  $a : \exists R.C$ .

- The side condition 'if there is no  $d$  such that both  $(a, d) : R$  and  $d : C$  are in  $\Delta$ ' ensures that the rule is applied at most once for  $a : \exists R.C$ .
- The SOME rule is don't care non-deterministic.

- p.17

## Completion rules: The SOME rule (2)

- Intuition: Let  $\mathcal{I}_1 = (\Delta, \cdot\mathcal{I}_1)$  be a terminological interpretation over  $\Sigma = (\mathbf{O}, \mathbf{C}, \mathbf{R})$ .

Then,  $\mathcal{I}_1$  satisfies  $\{a : \exists R.C\}$

iff there exists a  $y \in \Delta$  such that

$$(\mathcal{I}_1^a, y) \in R^{\mathcal{I}_1} \text{ and } y \in C^{\mathcal{I}_1}$$

$$\text{iff } (\mathcal{I}_2^a, d^{\mathcal{I}_2}) \in R^{\mathcal{I}_2} \text{ and } d^{\mathcal{I}_2} \in C^{\mathcal{I}_2}$$

where  $\mathcal{I}_2 = (\Delta, \cdot\mathcal{I}_2)$  is a terminological interpretation over  $\Sigma = (\mathbf{O} \cup \{d\}, \mathbf{C}, \mathbf{R})$  with  $\cdot\mathcal{I}_2 = \cdot\mathcal{I}_1$  except  $d^{\mathcal{I}_2} = y$ .

iff  $\mathcal{I}_2$  satisfies  $\{(a, d) : R, d : C\}$

Thus,  $\mathcal{I}_1$  satisfies  $\Delta \cup \{a : \exists R.C\}$

iff  $\mathcal{I}_2$  satisfies  $\Delta \cup \{(a, d) : R, d : C, a : \exists R.C\}$

- p.18

## The SOME rule: Example

- Reconsider the constraint system

$$\Delta_6 = \{\text{tim} : \exists\text{hasChild.person}, \\ \text{sue} : \text{student}, (\text{tim}, \text{sue}) : \text{hasChild}, \dots\}$$

- To see whether we can apply the SOME rule to  $\text{tim} : \exists\text{hasChild.person}$ , we have to check whether there is some object symbol  $d$  with  $(\text{tim}, d) : \text{hasChild}$  and  $d : \text{person}$  in  $\Delta_6$ . There is the object symbol  $\text{sue}$  with  $(\text{tim}, \text{sue}) : \text{hasChild}$  in  $\Delta_6$ , but  $\text{sue} : \text{person}$  is not in  $\Delta_6$ .

So, the SOME rule is applicable to  $\text{tim} : \exists\text{hasChild.person}$ .

- To apply the SOME rule, we use a new object symbol  $\text{che}$  and obtain  $\Delta_7 = \{\text{tim} : \exists\text{hasChild.person}, \\ \text{sue} : \text{student}, (\text{tim}, \text{sue}) : \text{hasChild}, \\ \text{che} : \text{person}, (\text{tim}, \text{che}) : \text{hasChild}, \dots\}$ .

- p.19

## Completion rules: The ALL rule (1)

- The ALL rule takes the following form:
 
$$\Delta \Rightarrow_{\forall} \Delta \cup \{b : C\}$$
 if  $a : \forall R.C$  and  $(a, b) : R$  are in  $\Delta$  and  $b : C$  is not in  $\Delta$ .  
 and we say that the ALL rule is applied to  $a : \forall R.C$  and  $(a, b) : R$ .
- The side condition 'if  $b : C$  is not in  $\Delta$ ' ensures that the rule is applied at most once for the pair  $a : \forall R.C$  and  $(a, b) : R$ .

- p.20

## Completion rules: The ALL rule (2)

- Intuition: Let  $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$  be a terminological interpretation. Then,  $\mathcal{I}$  satisfies  $\{a : \forall R.C, (a, b) : R\}$  iff  $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$  and for every  $y \in \Delta$ ,  $(a^{\mathcal{I}}, y) \in R^{\mathcal{I}}$  implies  $y \in C^{\mathcal{I}}$  iff  $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ ,  $b^{\mathcal{I}} \in C^{\mathcal{I}}$ , and for every  $y \in \Delta$ ,  $(a^{\mathcal{I}}, y) \in R^{\mathcal{I}}$  implies  $y \in C^{\mathcal{I}}$  iff  $\mathcal{I}$  satisfies  $\{a : \forall R.C, (a, b) : R, b : C\}$
- Thus,  $\mathcal{I}$  satisfies  $\Delta \cup \{a : \forall R.C, (a, b) : R\}$  iff  $\mathcal{I}$  satisfies  $\Delta \cup \{a : \forall R.C, (a, b) : R, b : C\}$

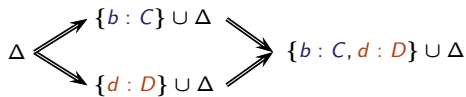
- p.21

## Completion rules: The ALL rule (3)

- The ALL rule is don't care non-deterministic, that is, given a constraint system

$$\Delta = \{a : \forall R.C, (a, b) : R, c : \forall S.D, (c, d) : S\} \cup \Delta'$$

such that neither  $b : C$  nor  $d : D$  are in  $\Delta$ , it makes no difference whether we apply the rule first to  $a : \forall R.C$  and  $(a, b) : R$  or to  $c : \forall S.D$  and  $(c, d) : S$ .



- p.22

## The ALL rule: Example

- Reconsider the constraint system  $\Delta_7 = \{\text{tim} : \forall \text{hasChild.happy}, (\text{tim}, \text{sue}) : \text{hasChild}, \dots\}$
- To see whether we can apply the ALL rule to  $\text{tim} : \forall \text{hasChild.happy}$  and  $(\text{tim}, \text{sue}) : \text{hasChild}$ , we have to check whether  $\text{sue} : \text{happy}$  is in  $\Delta_7$ . This is not case, so the ALL rule is applicable.
- As a result we obtain  $\Delta_8 = \{\text{tim} : \forall \text{hasChild.happy}, (\text{tim}, \text{sue}) : \text{hasChild}, \text{sue} : \text{happy}, \dots\}$

- p.23

## Completion rules: The clash rule (1)

- The clash rule takes the following form:
 
$$\Delta \Rightarrow_{\perp} \Delta \cup \{a : \perp\}$$
 if  $a : A$  and  $a : \neg A$  are in  $\Delta$ , where  $A$  is a concept symbol, and  $a : \perp$  is not in  $\Delta$ .  
 and we say that  $\Delta$  contains a clash (between  $a : A$  and  $a : \neg A$ ) and that the clash rule is applied to  $a : A$  and  $a : \neg A$ .
- The side condition 'if  $a : \perp$  is not in  $\Delta$ ' ensures that the rule is applied at most once for the pair  $a : A$  and  $a : \neg A$ .
- A constraint system containing a constraint  $a : \perp$  for any object symbol  $a$  is called *contradictory*.

- p.24

## Completion rules: The clash rule (2)

- Intuition:
  - The ABox  $\{a : A, a : \neg A\}$  is *unsatisfiable*, since for no terminological interpretation  $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$  we can have both  $a^{\mathcal{I}} \in A^{\mathcal{I}}$  and  $a^{\mathcal{I}} \in \neg A^{\mathcal{I}} = \Delta \setminus A^{\mathcal{I}}$ .
  - Also, the ABox  $\{a : \perp\}$  is *unsatisfiable*, since for no terminological interpretation  $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$  we can have  $a^{\mathcal{I}} \in \perp^{\mathcal{I}} = \emptyset$ .
- Thus,  $\mathcal{I}$  satisfies  $\Delta \cup \{a : A, a : \neg A\}$  iff  $\mathcal{I}$  satisfies  $\Delta \cup \{a : \perp\}$  since no  $\mathcal{I}$  will ever satisfy  $\Delta \cup \{a : A, a : \neg A\}$  or  $\Delta \cup \{a : \perp\}$ .

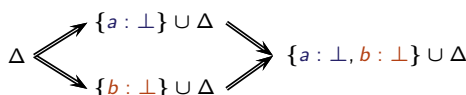
- p.25

## Completion rules: The clash rule (3)

- The clash rule is don't care non-deterministic, that is, given a constraint system

$$\Delta = \{a : A, a : \neg A, b : B, b : \neg B\} \cup \Delta'$$

such that neither  $a : \perp$  nor  $b : \perp$  are in  $\Delta$ , it makes no difference whether we apply the rule first to  $a : A$  and  $a : \neg A$  or to  $b : B$  and  $b : \neg B$ .



- p.26

## The clash rule: Example (1)

- Reconsider the constraint system  $\Delta_5$ , one of the constraint system we can obtain from  $\Delta_4$ :  $\Delta_5 = \{\text{sue} : \neg \text{student}, \text{sue} : \text{student}, \dots\}$
- By application of the clash rule to  $\text{sue} : \neg \text{student}$  and  $\text{sue} : \text{student}$  we obtain  $\Delta_9 = \{\text{sue} : \perp, \text{sue} : \neg \text{student}, \text{sue} : \text{student}, \dots\}$  which is evidently not satisfiable.

- p.27

## The clash rule: Example (2)

- Alternatively, we can consider the constraint system  $\Delta_8 = \{\text{sue} : \neg\text{happy}, \text{sue} : \text{happy}, \dots\}$ .
- By application of the clash rule to  $\text{sue} : \neg\text{happy}$  and  $\text{sue} : \text{happy}$  we obtain

$$\Delta_{10} = \{\text{sue} : \perp, \text{sue} : \neg\text{happy}, \text{sue} : \text{happy}, \dots\}$$

which is also evidently not satisfiable.

- Overall we get

$$\begin{aligned} \Delta_1 \Rightarrow_{\cap} \Delta_2 \Rightarrow_{\cap} \Delta_3 \Rightarrow_{\cap} \Delta_4 \Rightarrow_{\sqcup} \Delta_5 \Rightarrow_{\perp} \Delta_9 \\ \Delta_4 \Rightarrow_{\sqcup} \Delta_6 \Rightarrow_{\exists} \Delta_7 \Rightarrow_{\forall} \Delta_8 \Rightarrow_{\perp} \Delta_{10} \end{aligned}$$

Since beside  $\Delta_9$  and  $\Delta_{10}$  there are no further alternative constraint systems that we could consider, and both are obviously unsatisfiable, so is  $\Delta_1$  (as we will see later).

- p.28

## Summary

- Tableaux calculus for deciding the consistency of ABoxes
- Deterministic and non-deterministic rules
- Completion rules

- p.29

# Knowledge Representation and Reasoning Part 1: Modal and Description Logics

Wiebe van der Hoek

## Lecture 19: Description logic (7)

## Completion rules: Summary (1)

- The satisfiability of an ABox can be determined using the following completion rules:
  - $\Delta \Rightarrow_{\cap} \Delta \cup \{a : C, a : D\}$   
if  $a : (C \sqcap D)$  is in  $\Delta$ ,  $a : C$  and  $a : D$  are not both in  $\Delta$ .
  - $\Delta \Rightarrow_{\sqcup} \Delta \cup \{a : E\}$   
if  $a : (C \sqcup D)$  is in  $\Delta$ , neither  $a : C$  nor  $a : D$  is in  $\Delta$  and  $E = C$  or  $E = D$ .
  - $\Delta \Rightarrow_{\exists} \Delta \cup \{(a, b) : R, b : C\}$   
if  $a : \exists R.C$  is in  $\Delta$ , there is no  $d$  such that both  $(a, d) : R$  and  $d : C$  are in  $\Delta$ , and  $b$  is a new object symbol with respect to  $\Delta$ .

- p.1

## Completion rules: Summary (2)

- $\Delta \Rightarrow_{\forall} \Delta \cup \{b : C\}$   
if  $a : \forall R.C$  and  $(a, b) : R$  are in  $\Delta$  and  $b : C$  is not in  $\Delta$ .
- $\Delta \Rightarrow_{\perp} \Delta \cup \{a : \perp\}$   
if  $a : A$  and  $a : \neg A$  are in  $\Delta$ , where  $A$  is a concept symbol.
- In the following let  $\Rightarrow$  denote either one of  $\Rightarrow_{\cap}$ ,  $\Rightarrow_{\sqcup}$ ,  $\Rightarrow_{\exists}$ ,  $\Rightarrow_{\forall}$ , and  $\Rightarrow_{\perp}$ .

- p.2

## Derivations (1)

- A tableaux derivation from an ABox  $\mathcal{A}$  is any sequence of constraint systems  $\Delta_0, \Delta_1, \dots$  such that
    - $\Delta_0 = \mathcal{A}$ ,
    - $\Delta_i \Rightarrow \Delta_{i+1}$ , for every  $i, 0 \leq i$ ,
 which we also write as  $\Delta = \Delta_0 \Rightarrow \Delta_1 \Rightarrow \dots$ .
  - Any derivation can be constructed by
    - taking  $\Delta_0$  to be the ABox  $\mathcal{A}$ , and
    - to iteratively obtain  $\Delta_{i+1}$  from  $\Delta_i$  by application of one of the completion rules
- We call the application of one of the completion rules to  $\Delta_i$  a step of the derivation or derivation step.

- p.3

## Derivations (2)

- Note that, by definition, derivations could be finite as well as infinite sequences of constraint systems.
- In case a derivation is finite we can write it as  $\mathcal{A} = \Delta_0 \Rightarrow \Delta_1 \Rightarrow \dots \Rightarrow \Delta_n$ .
- A finite derivation  $\mathcal{A} = \Delta_0 \Rightarrow \Delta_1 \Rightarrow \dots \Rightarrow \Delta_n$  is complete iff there is no constraint system  $\Delta_{n+1}$  such that  $\Delta_n \Rightarrow \Delta_{n+1}$ . In this case, we also say that  $\Delta_n$  is complete, that the derivation terminates, and that the derivation is terminating.
- Note that this definition does not imply that there are any finite, complete derivations.

- p.4

## Properties of tableaux calculi

- In analogy to the properties soundness, completeness, and termination of a deductive system, we can define the properties for the refutation system we have just defined.
- A refutation system is terminating iff for every initial tableau  $\Delta$  any derivation from  $\Delta$  is terminating.
- A refutation system is sound iff for every initial tableau  $\Delta$ , if for every complete derivation  $\Delta = \Delta_1, \dots, \Delta_n$   $\Delta_n$  is contradictory then  $\Delta$  is unsatisfiable.
- A refutation system is complete iff for every initial tableau  $\Delta$ , if  $\Delta$  is unsatisfiable then for every complete derivation  $\Delta = \Delta_1, \dots, \Delta_n$   $\Delta_n$  is contradictory.

- p.5

## Termination of derivations: Theorem

### Theorem 19.1

Let  $\mathcal{A}$  be an ABox.

Any derivation from  $\mathcal{A}$  terminates.

- p.6

## Satisfiability of ABoxes: Theorem

### Theorem 19.2

Let  $\mathcal{A}$  be an ABox.

Then  $\mathcal{A}$  is **satisfiable** if and only if **there exists a complete derivation**

$$\mathcal{A} = \Delta_0 \Rightarrow \Delta_1 \Rightarrow \dots \Rightarrow \Delta_n$$

such that  $a : \perp$  is **not an element of  $\Delta_n$**  for any object symbol  $a$ .

- p.7

## Unsatisfiability of ABoxes: Corollary

### Corollary 19.1

Let  $\mathcal{A}$  be an ABox.

Then  $\mathcal{A}$  is **unsatisfiable** if and only if **for every complete derivation**

$$\mathcal{A} = \Delta_0 \Rightarrow \Delta_1 \Rightarrow \dots \Rightarrow \Delta_n$$

there exists an object symbol  $a$  such that  $a : \perp$  is an element of  $\Delta_n$ .

- p.8

## Unsatisfiability of ABoxes: Lemma

### Lemma 19.1

Let  $\mathcal{A}$  be an ABox.

Then  $\mathcal{A}$  is **unsatisfiable** if and only if **for every completed constraint system  $\Delta_n$  such that there exists a derivation**

$$\mathcal{A} = \Delta_0 \Rightarrow \Delta_1 \Rightarrow \dots \Rightarrow \Delta_n,$$

there exists an object symbol  $a$  such that  $a : \perp$  is an element of  $\Delta_n$ .

- p.9

## Unsatisfiability of ABoxes

- Suppose we want to show that an ABox  $\mathcal{A}$  is unsatisfiable.
- By Corollary 19.1 we have to consider all complete derivations from  $\Delta$ .
- An inspection of the completion rules reveals that for any derivation

$$\mathcal{A} = \Delta_0 \Rightarrow \dots \Rightarrow \Delta_i \Rightarrow \dots$$

if  $a : \perp$  is an element of  $\Delta_i$  (for some object symbol  $a$ ), then  $a : \perp$  is also an element of any constraint system  $\Delta_j$ ,  $j \geq i$ , occurring in the derivation.

- p.10

## Unsatisfiability of ABoxes

- Thus, if in the process of constructing a derivation

$$\mathcal{A} = \Delta_0 \Rightarrow \dots \Rightarrow \Delta_i \Rightarrow \dots$$

we obtain a  $\Delta_i$  such that  $a : \perp$  is an element of  $\Delta_i$ , we can stop even if the derivation is not yet complete, since for any complete derivation

$$\mathcal{A} = \Delta_0 \Rightarrow \dots \Rightarrow \Delta_i \Rightarrow \dots \Rightarrow \Delta_n$$

$\Delta_n$  will contain  $a : \perp$ .

- p.11

## Unsatisfiability of ABoxes: Example

ABox	tim : (person $\sqcap$ $\exists$ hasChild.person) $\sqcap$ $\forall$ hasChild.happy $\sqcap$ happyParent*
	sue : student (tim, sue) : hasChild
	sue : $\neg$ student $\sqcup$ $\neg$ happy

We can obtain the following derivation from the ABox above:

$$\Delta_1 = \{ \text{tim : (person } \sqcap \exists \text{hasChild.person) } \sqcap \\ (\forall \text{hasChild.happy } \sqcap \text{happyParent}^*), \\ \text{sue : student, (tim, sue) : hasChild,} \\ \text{sue : } \neg \text{student } \sqcup \neg \text{happy} \}$$

$$\Rightarrow_{\sqcup} \Delta_{11} = \{ \text{sue : } \neg \text{student} \} \cup \Delta_1$$

$$\Rightarrow_{\perp} \Delta_{12} = \{ \text{sue : } \perp \} \cup \Delta_{11}$$

Although  $\Delta_{12}$  is not complete, any constraint system derivable from  $\Delta_{12}$  will contain  $\text{sue : } \perp$ .

- p.12

## Exercise 14

ABox	c10 : room $\sqcap$ $\forall$ hasEquip.pc
	tv1 : $\neg$ pc
	(c10, tv1) : hasEquip

Give a tableau derivation which determines whether this ABox is satisfiable.

- p.13

## Exercise 14: Answer

<p>ABox</p> <p><math>c10 : \text{room} \sqcap \forall \text{hasEquip.pc}</math></p> <p><math>tv1 : \neg \text{pc}</math></p> <p><math>(c10, tv1) : \text{hasEquip}</math></p>
---

Give a tableau derivation which determines whether this ABox is satisfiable.

$$\Delta_1 = \{c10 : \text{room} \sqcap \forall \text{hasEquip.pc}, tv1 : \neg \text{pc}, (c10, tv1) : \text{hasEquip}\}$$

$$\Rightarrow_{\sqcap} \Delta_2 = \{c10 : \text{room}, c10 : \forall \text{hasEquip.pc}\} \cup \Delta_1$$

$$\Rightarrow_{\forall} \Delta_3 = \{tv1 : \text{pc}\} \cup \Delta_2$$

$$\Rightarrow_{\perp} \Delta_4 = \{tv1 : \perp\} \cup \Delta_3$$

Since  $\Delta_4$  contains  $tv1 : \perp$  and no other constraint system is derivable, the ABox above is unsatisfiable.

- p.14

## Complexity of $\mathcal{ALC}$ : Theorem

### Theorem 19.3

The problem of deciding whether an  $\mathcal{ALC}$ -ABox is satisfiable, that is the problem of deciding the

Consistency of an ABox,

is PSPACE-complete.

Recall that

$$P \subseteq NP \subseteq PSPACE \subseteq EXPTIME \subseteq NEXPTIME,$$

and that it is yet unknown whether these are strict inclusions or not (currently, the best guess is that they are.)

- p.15

## Complexity of $\mathcal{ALC}$ : Discussion (1)

- Our simple procedure based on the elimination of the TBox from a knowledge base  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$  and testing the satisfiability of the resulting ABox by using completion rules does not provide us with a PSPACE decision procedure:
  - ▶ the elimination of defined concepts can result in an ABox  $\mathcal{A}^*$  of exponential size in the size of the original ABox  $\mathcal{A}$ ;
  - ▶ the constraint systems derivable from  $\mathcal{A}^*$  can again be of exponential size in the size of  $\mathcal{A}^*$

- p.16

## Complexity of $\mathcal{ALC}$ : Discussion (2)

- To obtain a PSPACE decision procedure for deciding the consistency of an ABox  $\mathcal{A}$  wrt. a TBox  $\mathcal{T}$ , we have to use two techniques:
  - ▶ we have to use *lazy unfolding*, that is, concept definition are *unfolded on demand* during a derivation, not prior to it;
  - ▶ we have to make use of the *trace technique* which allows us to delete certain constraints from a constraint system once we are sure that they do not lead to the derivation of a *clash*.

This reduces the space requirements of the procedure to *polynomial space* in the size of the initial ABox  $\mathcal{A}$ .

- p.17

## The trace technique

- *Traces* are paths in the graph representation of an ABox (or constraint system)
- It can be shown that, in the case of  $\mathcal{ALC}$ , we can investigate these *traces independently*.  
To be able to do this we have to control applications of the SOME rule more tightly:
  - ▶ we apply the SOME rule only if no other rule is applicable;
  - ▶ if the SOME rule is applicable to more than one constraint, choose the constraint with the most recently generated object symbol;
  - ▶ we delete any constraints for an object symbol  $a$  once no further constraints concerning  $a$  can be generated and no clash involving  $a$  has been detected.

- p.18

## The trace technique: Example

Consider the ABox

$$\mathcal{A} = \{\text{tim} : \exists \text{owns.pc}, \text{tim} : \exists \text{owns.car}, \text{tim} : \forall \text{owns.fast}\}$$

Using the completion rules a 'standard derivation' from  $\mathcal{A}$  is the following:

$$\Delta_0 = \mathcal{A}$$

$$\Rightarrow_{\exists} \Delta_1 = \{(\text{tim}, \text{pc1}) : \text{owns}, \text{pc1} : \text{pc}\} \cup \Delta_0$$

$$\Rightarrow_{\exists} \Delta_2 = \{(\text{tim}, \text{car1}) : \text{owns}, \text{car1} : \text{car}\} \cup \Delta_1$$

$$\Rightarrow_{\forall} \Delta_3 = \{\text{pc1} : \text{fast}\} \cup \Delta_2$$

$$\Rightarrow_{\forall} \Delta_4 = \{\text{car1} : \text{fast}\} \cup \Delta_3$$

$\Delta_4$  is complete.

- p.19

## The trace technique: Example

$$\mathcal{A} = \{\text{tim} : \exists \text{owns.pc}, \text{tim} : \exists \text{owns.car}, \text{tim} : \forall \text{owns.fast}\}$$

Using the *trace technique* we obtain the following derivation:

$$\Delta_0 = \mathcal{A}$$

$$\Rightarrow_{\exists} \Delta_1 = \{(\text{tim}, \text{pc1}) : \text{owns}, \text{pc1} : \text{pc}\} \cup \Delta_0 \setminus \{\text{tim} : \exists \text{owns.pc}\}$$

SOME rule not applicable to  $\Delta_1$ , since the ALL rule is applicable

$$\Rightarrow_{\forall} \Delta_2 = \{\text{pc1} : \text{fast}\} \cup \Delta_1$$

no further constraints for  $\text{pc1}$ , delete all constraints for  $\text{pc1}$

$$\Rightarrow_{\text{del}} \Delta_3 = \Delta_0 \setminus \{\text{tim} : \exists \text{owns.pc}\}$$

$$\Rightarrow_{\exists} \Delta_4 = \{(\text{tim}, \text{car1}) : \text{owns}, \text{car1} : \text{car}\} \cup \Delta_3 \setminus \{\text{tim} : \exists \text{owns.car}\}$$

$$\Rightarrow_{\forall} \Delta_5 = \{\text{car1} : \text{fast}\} \cup \Delta_4$$

no further constraints for  $\text{car1}$ , delete all constraints for  $\text{car1}$

$$\Rightarrow_{\text{del}} \Delta_6 = \{\text{tim} : \forall \text{owns.fast}\}$$

$\Delta_6$  is complete. Note that we have used *less space* for this derivation compared to the 'standard derivation'.

- p.20

## Summary ...

- Derivation from an ABox  $\mathcal{A}$ 

$$\mathcal{A} = \Delta_0 \Rightarrow \Delta_1 \Rightarrow \dots \Rightarrow \Delta_n$$
- Termination
- Soundness  
If for every derivation from  $\mathcal{A}$ ,  $a : \perp$  in  $\Delta_n$  for some object symbol  $a$ , then  $\mathcal{A}$  is unsatisfiable
- Completeness  
If  $\mathcal{A}$  is unsatisfiable, then for every derivation from  $\mathcal{A}$ ,  $a : \perp$  in  $\Delta_n$  for some object symbol  $a$
- Complexity  
There is a procedure for testing the satisfiability of an ABox wrt. a TBox which requires only polynomial space
- Trace technique

- p.21

# Knowledge Representation and Reasoning Part 1: Modal and Description Logics

Wiebe van der Hoek

## Lecture 20: Description logic (8)

### Last time ...

- Derivation from an ABox  $\mathcal{A}$

$$\mathcal{A} = \Delta_0 \Rightarrow \Delta_1 \Rightarrow \dots \Rightarrow \Delta_n$$

- Termination

- Soundness

If for every derivation from  $\mathcal{A}$ ,  $a : \perp$  in  $\Delta_n$  for some object symbol  $a$ , then  $\mathcal{A}$  is unsatisfiable

- Completeness

If  $\mathcal{A}$  is unsatisfiable, then for every derivation from  $\mathcal{A}$ ,  $a : \perp$  in  $\Delta_n$  for some object symbol  $a$

- Complexity

There is a procedure for testing the satisfiability of an ABox wrt. a TBox which requires only polynomial space

- Trace technique

- p.1

### ALC and first-order logic (1)

- As in the case of basic modal logic K we can translate knowledge bases of the description logic **ALC** to first-order logic.
- To this end, given a **ALC**-signature  $\Sigma = (\mathbf{O}, \mathbf{C}, \mathbf{R})$ , with every object symbol  $a \in \mathbf{O}$  we associate a constant  $c_a$ , with every concept symbol  $A \in \mathbf{C}$  we associate a unary predicate symbol  $q_A$ , and with every role symbol  $R \in \mathbf{R}$  we associate a binary predicate symbol  $q_R$ .

object symbol  $a \in \mathbf{O} \rightsquigarrow$  constant  $c_a$

concept symbol  $A \in \mathbf{C} \rightsquigarrow$  unary predicate  $q_A$

role symbol  $R \in \mathbf{R} \rightsquigarrow$  binary predicate  $q_R$

- p.2

### Translation of concepts

We now define a function  $\tau$  that takes a **ALC**-concept and a variable symbol or constant of first-order logic as arguments and returns a first-order formula.

$$\tau(A, x) = q_A(x) \quad \text{for any concept symbol } A$$

$$\tau(\top, x) = \top$$

$$\tau(\perp, x) = \perp$$

$$\tau(\neg C, x) = \neg \tau(C, x)$$

$$\tau(C \sqcap D, x) = \tau(C, x) \wedge \tau(D, x)$$

$$\tau(C \sqcup D, x) = \tau(C, x) \vee \tau(D, x)$$

$$\tau(\forall R.C, x) = \forall y \cdot q_R(x, y) \rightarrow \tau(C, y) \quad \text{where } y \text{ is new}$$

$$\tau(\exists R.C, x) = \exists y \cdot q_R(x, y) \wedge \tau(C, y) \quad \text{first-order variable}$$

- p.3

### Translation of concepts: Example

Consider the concept  $(\forall \text{hasEquip.pc}) \sqcap (\exists \text{hasUse.}\neg \text{work})$ . We can compute the translation of this concept as follows.

$$\begin{aligned} & \tau((\forall \text{hasEquip.pc}) \sqcap (\exists \text{hasUse.}\neg \text{work}), x) \\ &= \tau(\forall \text{hasEquip.pc}, x) \wedge \tau(\exists \text{hasUse.}\neg \text{work}, x) \\ &= (\forall y \cdot q_{\text{hasEquip}}(x, y) \rightarrow \tau(\text{pc}, y)) \wedge \tau(\exists \text{hasUse.}\neg \text{work}, x) \\ &= (\forall y \cdot q_{\text{hasEquip}}(x, y) \rightarrow q_{\text{pc}}(y)) \wedge \tau(\exists \text{hasUse.}\neg \text{work}, x) \\ &= (\forall y \cdot q_{\text{hasEquip}}(x, y) \rightarrow q_{\text{pc}}(y)) \wedge \\ & \quad (\exists z \cdot q_{\text{hasUse}}(x, z) \wedge \tau(\neg \text{work}, z)) \\ &= (\forall y \cdot q_{\text{hasEquip}}(x, y) \rightarrow q_{\text{pc}}(y)) \wedge \\ & \quad (\exists z \cdot q_{\text{hasUse}}(x, z) \wedge \neg \tau(\text{work}, z)) \\ &= (\forall y \cdot q_{\text{hasEquip}}(x, y) \rightarrow q_{\text{pc}}(y)) \wedge \\ & \quad (\exists z \cdot q_{\text{hasUse}}(x, z) \wedge \neg q_{\text{work}}(z)) \end{aligned}$$

- p.4

### Translation of knowledge bases

We now extend the function  $\tau$  to a mapping from **ALC**-knowledge bases  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$  to conjunctions of first-order formulae.

$$\tau(A \sqsubseteq C) = (\forall x \cdot \tau(A, x) \rightarrow \tau(C, x))$$

$$\tau(A \doteq C) = (\forall x \cdot \tau(A, x) \leftrightarrow \tau(C, x))$$

$$\tau(a : C) = \tau(C, c_a)$$

$$\tau((a, b) : R) = q_R(c_a, c_b)$$

$$\begin{aligned} \tau(\mathcal{K}) &= (\bigwedge_{\alpha \in \mathcal{T}} \tau(\alpha)) \wedge (\bigwedge_{\alpha \in \mathcal{A}} \tau(\alpha)) \\ & \quad \wedge (\bigwedge_{\text{distinct } a, b \text{ occurring in } \mathcal{A}} c_a \neq c_b) \end{aligned}$$

The blue part of the translation of  $\mathcal{K}$  reflects the unique name assumption on the first-order level and is optional.

- p.5

### Translation of knowledge bases: Example

Consider the knowledge base  $\mathcal{K}$  below.

TBox	cpu $\sqsubseteq$ amd $\sqcup$ intel	
	pc $\doteq$ $\exists$ hasPart.cpu	
	pcLab $\doteq$ room $\sqcap$ $\forall$ hasEquip.pc	
ABox	t10 : $\neg$ pcLab	t10 : room
	t11 : room	

The translation  $\tau(\mathcal{K})$  is the following first-order formula:

$$\begin{aligned} & (\forall x \cdot q_{\text{cpu}}(x) \rightarrow q_{\text{amd}}(x) \vee q_{\text{intel}}(x)) \\ & \wedge (\forall x \cdot q_{\text{pc}}(x) \leftrightarrow (\exists y \cdot q_{\text{hasPart}}(x, y) \wedge q_{\text{cpu}}(y))) \\ & \wedge (\forall x \cdot q_{\text{pcLab}}(x) \leftrightarrow (q_{\text{room}}(x) \wedge (\forall y \cdot q_{\text{hasEquip}}(x, y) \rightarrow q_{\text{pc}}(y)))) \\ & \wedge \neg q_{\text{pcLab}}(c_{t10}) \wedge q_{\text{room}}(c_{t10}) \wedge q_{\text{room}}(c_{t11}) \\ & \wedge (c_{t10} \neq c_{t11}) \end{aligned}$$

- p.6

### Translation of knowledge bases: Theorem

#### Theorem 20.1

Let  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$  be a knowledge base (the TBox  $\mathcal{T}$  does not need to be acyclic).

Then  $\mathcal{K}$  is satisfiable iff  $\tau(\mathcal{K})$  is satisfiable in first-order logic.

#### Proof

The proof proceeds by showing that any model for  $\mathcal{K}$  can be transformed into a first-order interpretation satisfying  $\tau(\mathcal{K})$  and vice versa. To see that the inequalities in  $\tau(\mathcal{K})$  which reflect the unique name assumption in knowledge bases is not a necessary part of the translation, note that we can restrict ourselves to **Herbrand models** of  $\tau(\mathcal{K})$ , which by definition satisfy the unique name assumption even if not enforced by  $\tau(\mathcal{K})$ .

- p.7

## Translation to first-order logic: Conclusion

- As in the case of modal logic, once a knowledge base has been translated to first-order logic, the satisfiability of the first-order formulae we obtain can be determined by a number of means.
- We can again use **first-order resolution**, a sound and complete deductive system for first-order logic, and particular refinements of first-order resolution will always terminate on translated knowledge bases.

This provides us with an alternative **decision procedure** for the problem of deciding the

Consistency of an ABox wrt. a TBox

and, consequently, for all the other inferential services we expect of a description logic system.

- p.8

## Exercise 15

ABox     It7 :  $\forall \text{takeInto.}(\neg \text{foot} \sqcap \neg \text{drink})$   
           us1 : (drink  $\sqcup$  food)  
           (It7, us1) : takeInto

Give a tableau derivation which determines whether this ABox is satisfiable.

- p.9

## Exercise 15: Answer

ABox     It7 :  $\forall \text{takeInto.}(\neg \text{food} \sqcap \neg \text{drink})$   
           (It7, us1) : takeInto     us1 : (drink  $\sqcup$  food)

Is this ABox is satisfiable?

$$\Delta_1 = \{\text{It7} : \forall \text{takeInto.}(\neg \text{food} \sqcap \neg \text{drink}), (\text{It7}, \text{us1}) : \text{takeInto}, \text{us1} : (\text{drink} \sqcup \text{food})\}$$

$$\Rightarrow_{\forall} \Delta_2 = \{\text{us1} : (\neg \text{foot} \sqcap \neg \text{drink})\} \cup \Delta_1$$

$$\Rightarrow_{\sqcap} \Delta_3 = \{\text{us1} : \neg \text{foot}, \text{us1} : \neg \text{drink}\} \cup \Delta_2$$

$$\Rightarrow_{\sqcup} \Delta_4 = \{\text{us1} : \text{drink}\} \cup \Delta_3$$

$$\Rightarrow_{\perp} \Delta_5 = \{\text{us1} : \perp\} \cup \Delta_4$$

$\Delta_5$  is contradictory, but the step from  $\Delta_3$  to  $\Delta_4$  was don't know non-deterministic. Thus, we have to look at the alternatives to  $\Delta_4$ .

- p.10

## Exercise 15: Answer

ABox     It7 :  $\forall \text{takeInto.}(\neg \text{food} \sqcap \neg \text{drink})$   
           (It7, us1) : takeInto     us1 : (drink  $\sqcup$  food)

Is this ABox is satisfiable?

$$\Delta_1 = \{\text{It7} : \forall \text{takeInto.}(\neg \text{food} \sqcap \neg \text{drink}), (\text{It7}, \text{us1}) : \text{takeInto}, \text{us1} : (\text{drink} \sqcup \text{food})\}$$

$$\Rightarrow_{\forall} \Delta_2 = \{\text{us1} : (\neg \text{foot} \sqcap \neg \text{drink})\} \cup \Delta_1$$

$$\Rightarrow_{\sqcap} \Delta_3 = \{\text{us1} : \neg \text{foot}, \text{us1} : \neg \text{drink}\} \cup \Delta_2$$

$$\Rightarrow_{\sqcup} \Delta_6 = \{\text{us1} : \text{food}\} \cup \Delta_3$$

$$\Rightarrow_{\perp} \Delta_7 = \{\text{us1} : \perp\} \cup \Delta_6$$

$\Delta_7$  is also contradictory. Since both  $\Delta_5$  and  $\Delta_7$  are contradictory and no further constraint systems are derivable from  $\Delta_0$ , we conclude that the ABox is unsatisfiable.

- p.10

## $\mathcal{ALC}$ and basic modal logic (1)

Comparing the translation  $\pi$  of modal formulae to first-order logic and the translation  $\tau$  of concepts to first-order logic, we see a close correspondence:

modal logic	description logic
propositional variable $p$	concept symbol $A$
$\varphi \wedge \psi$	$C \sqcap D$
$\varphi \vee \psi$	$C \sqcup D$
$\Box \varphi$	$\forall r.C$ (where $r$ is an arbitrary, but fixed role symbol)
$\Diamond \varphi$	$\exists r.C$

- p.11

## $\mathcal{ALC}$ and basic modal logic (2)

Formally, we can define a function  $\eta$  that maps modal formulae to description logic concepts. To this end we uniquely associate with every propositional variable  $p$  a concept symbol  $A_p$  and we fix an arbitrary role symbol  $r$ .

$$\eta(p) = A_p$$

$$\eta(\neg \varphi) = \neg \eta(\varphi)$$

$$\eta(\top) = \top$$

$$\eta(\perp) = \perp$$

$$\eta(\varphi \wedge \psi) = \eta(\varphi) \sqcap \eta(\psi)$$

$$\eta(\varphi \vee \psi) = \eta(\varphi) \sqcup \eta(\psi)$$

$$\eta(\varphi \rightarrow \psi) = \neg \eta(\varphi) \sqcup \eta(\psi)$$

$$\eta(\varphi \leftrightarrow \psi) = \eta(\varphi \rightarrow \psi) \sqcap \eta(\psi \rightarrow \varphi)$$

$$\eta(\Box \varphi) = \forall r.\eta(\varphi)$$

$$\eta(\Diamond \varphi) = \exists r.\eta(\varphi)$$

- p.12

## $\mathcal{ALC}$ and basic modal logic: Example

Consider the modal formula  $(\Box p) \wedge (\Diamond \neg q)$ . We can compute the translation of this formula to an  $\mathcal{ALC}$ -concept as follows.

$$\begin{aligned} \eta((\Box p) \wedge (\Diamond \neg q)) &= \eta(\Box p) \sqcap \eta(\Diamond \neg q) \\ &= \forall r.\eta(p) \sqcap \eta(\Diamond \neg q) \\ &= \forall r.A_p \sqcap \eta(\Diamond \neg q) \\ &= \forall r.A_p \sqcap \exists r.\eta(\neg q) \\ &= \forall r.A_p \sqcap \exists r.\neg \eta(q) \\ &= \forall r.A_p \sqcap \exists r.\neg A_q \end{aligned}$$

- p.13

## $\mathcal{ALC}$ and basic modal logic: Theorem

### Theorem 20.2

Let  $\varphi$  be an arbitrary formula of basic modal logic.

Then  $\varphi$  is satisfiable in basic modal logic iff

$$\eta(\varphi) \text{ is a coherent } \mathcal{ALC}\text{-concept.}$$

Theorem 20.2 tells us that the similarity of basic modal logic and the description logic  $\mathcal{ALC}$  does not only exist on the notational level, but also on the model-theoretic level.

That is, not only do modal formulae and  $\mathcal{ALC}$ -concepts look similar, but also their interpretation is related.

- p.14

## ALC and multi-modal logic (1)

- The correspondence between modal logic and **ALC** is not yet perfect, since the inverse function  $\eta^{-1}$  is not able to translate all **ALC** concepts to modal formulae, since **ALC** concepts can contain several different role symbols.
- There is an extension of basic modal logic, the multi-modal logic  $K_{(m)}$ , which has  $m$  modal operators  $[i]$  and  $\langle i \rangle$ ,  $1 \leq i \leq m$ , instead of just  $\Box$  and  $\Diamond$ .

- p.15

## ALC and multi-modal logic (2)

- Given a **ALC**-signature  $\Sigma = (\mathbf{O}, \mathbf{C}, \mathbf{R})$ , we can now uniquely associate with every index  $i$ ,  $1 \leq i \leq m$ , a role symbol  $r_i \in \mathbf{R}$ , and vice versa.

We can then modify the function  $\eta$  as follows

$$\eta([i]\varphi) = \forall r_i. \eta(\varphi) \quad \eta(\langle i \rangle \varphi) = \exists r_i. \eta(\varphi)$$

replaces the equations for  $\Box\varphi$  and  $\Diamond\varphi$  in the definition of  $\eta$  while the rest of the definition remains unchanged.

- The modified function  $\eta$  not only maps every modal formula of the multi-modal logic  $K_{(m)}$  to an **ALC**-concept, but the inverse function  $\eta^{-1}$  also maps every **ALC**-concept to a formula of  $K_{(m)}$ .

- p.16

## ALC and multi-modal logic: Example

Consider the concept  $(\forall \text{hasEquip.pc}) \sqcap (\exists \text{hasUse.}\neg \text{work})$ .

Let us associate the indices 1 and 2 with `hasEquip` and `hasUse` respectively, and let us associate the propositional variables  $p_{pc}$  and  $p_{work}$  with the concept symbols `pc` and `work`, respectively.

$$\begin{aligned} \text{Then } \eta^{-1}((\forall \text{hasEquip.pc}) \sqcap (\exists \text{hasUse.}\neg \text{work})) \\ &= \eta^{-1}(\forall \text{hasEquip.pc}) \wedge \eta^{-1}(\exists \text{hasUse.}\neg \text{work}) \\ &= ([1]\eta^{-1}(pc)) \wedge \eta^{-1}(\exists \text{hasUse.}\neg \text{work}) \\ &= ([1]p_{pc}) \wedge \eta^{-1}(\exists \text{hasUse.}\neg \text{work}) \\ &= ([1]p_{pc}) \wedge (\langle 2 \rangle \eta^{-1}(\neg \text{work})) \\ &= ([1]p_{pc}) \wedge (\langle 2 \rangle \neg(\eta^{-1}(\text{work}))) \\ &= ([1]p_{pc}) \wedge (\langle 2 \rangle \neg p_{work}) \end{aligned}$$

- p.17

## ALC and multi-modal logic: Theorem

### Theorem 20.3

1. Let  $\varphi$  be an arbitrary formula of multi-modal logic  $K_{(m)}$   
Then  $\varphi$  is satisfiable in  $K_{(m)}$  iff  
 $\eta(\varphi)$  is a coherent **ALC**-concept.
2. Let  $C$  be an arbitrary **ALC**-concept over a finite signature  $\Sigma = (\mathbf{O}, \mathbf{C}, \mathbf{R})$  where  $\mathbf{R}$  contains  $m$  role symbols.  
Then  $C$  is a coherent **ALC**-concept iff  
 $\eta^{-1}(C)$  is satisfiable in  $K_{(m)}$ .

Theorem 20.3 tells us that there is a one-to-one correspondence between formulae of  $K_{(m)}$  and **ALC**-concepts on the notational level as well as on the model-theoretic level.

That is, the two logics are notational variants of each other.

- p.18

## ALC and multi-modal logic: Knowledge bases (1)

- Until now we have only looked at the relationship between modal formulae and **ALC**-concepts.  
Can we extend the relationship to **ALC**-knowledge bases?
- This is possible using a modal logic with universal modality and nominals instead of basic modal logic or multi-modal logic.

Modal logics with nominals are called **hybrid logics**.

Nominals give a 'name' to one particular world in a Kripke model and we can use them to state that a modal formula is true at one particular world.

In contrast, the **universal modality** is an additional modal operator that can be used to state that a modal formula is true at every world of a Kripke model.

- p.19

## ALC and multi-modal logic: Knowledge bases (2)

- Formally, we uniquely associate with every object symbol  $a$  a nominal  $n_a$  (and vice versa), and we denote the universal modality by **A**.
- We can then extend the inverse function  $\eta^{-1}$ , which maps **ALC**-concepts to formulae of  $K_{(m)}$ , to a mapping from concept definitions, concept assertion and role assertions to formulae of a hybrid logic:

$$\eta^{-1}(A \sqsubseteq C) = \mathbf{A}(\eta^{-1}(A) \rightarrow \eta^{-1}(C))$$

$$\eta^{-1}(A \doteq C) = \mathbf{A}(\eta^{-1}(A) \leftrightarrow \eta^{-1}(C))$$

$$\eta^{-1}(a : C) = @_{n_a} \eta^{-1}(C)$$

$$\eta^{-1}((a, b) : r_i) = @_{n_a} \langle i \rangle n_b$$

- p.20

## ALC and multi-modal logic: Conclusion

- There is a close, in some cases one-to-one, correspondence between description logics like **ALC** and modal logics like  $K_{(m)}$  and hybrid logics.
- This correspondence allows us
  - ▶ to transfer theoretical results from description logics to modal logics and vice versa (e.g., decidability and complexity results).
  - ▶ to use procedures developed for solving particular tasks for description logics to solve corresponding tasks for modal logics (e.g., satisfiability of concepts vs. satisfiability of modal formulae).

- p.21

## Knowledge Representation and Reasoning Part 1: Modal and Description Logics

Wiebe van der Hoek

### Lecture 21: Description logic (9)

## Last time ...

- **ALC** and  $K_{(m)}$   
Every **ALC**-concept can be translated into a modal formula of  $K_{(m)}$ , and vice versa, in a satisfiability equivalence preserving way.
- **ALC** and hybrid logic  
Every **ALC**-knowledge base can be translated into a formula of a hybrid logic with nominals and universal modalities in a satisfiability equivalence preserving way.
- **ALC** and first-order logic  
Every **ALC**-concept and **ALC**-knowledge base can be translated into a first-order formula in a satisfiability equivalence preserving way.

- p.1

## Beyond ALC: Motivation

- We know that first-order logic is only semi-decidable, that is, there is no sound and complete deductive system that for arbitrary first-order formulae could determine their satisfiability and could be guaranteed to terminate.
- We have seen that **ALC** is decidable, that is, there is a sound, complete and terminating deductive system that can determine the coherence of arbitrary **ALC**-concepts or the satisfiability of arbitrary **ALC**-knowledge bases.  
Furthermore, the satisfiability problem of **ALC** is only PSPACE-complete.
- So, there must be properties that can be expressed in first-order logic, but not in **ALC**.

- p.2

## Beyond ALC: Number restrictions (1)

- So far we have seen concept assertions like  
$$\text{tim} : \exists \text{hasChild.T}$$
which specifies that tim is someone who has a child.
- How do we specify that tim is someone who has two children?
- The concept assertion

$$\text{tim} : (\exists \text{hasChild.T}) \sqcap (\exists \text{hasChild.T.})$$

does not what we want. For example, it is satisfied by the following terminological interpretation



where tim has just one child.

- p.3

## Beyond ALC: Number restrictions (2)

- What we need are number restrictions.
- We extend the language of **ALC** as follows:  
If  $R$  is a role symbol and  $n$  is a natural number, then  
 $\exists_{\leq n} R$  (there are at most  $n$   $R$ -related elements) and  
 $\exists_{\geq n} R$  (there are at least  $n$   $R$ -related elements)  
are concepts.  
The semantics of these additional concept-forming operators is defined by extending the definition of  $\cdot^{\mathcal{I}}$ :

$$(\exists_{\leq n} R)^{\mathcal{I}} = \{x \mid |\{y \mid (x, y) \in R^{\mathcal{I}}\}| \leq n\}$$

$$(\exists_{\geq n} R)^{\mathcal{I}} = \{x \mid |\{y \mid (x, y) \in R^{\mathcal{I}}\}| \geq n\}$$

The description logic obtained in this way is denoted by **ALCN**.

- p.4

## Beyond ALC: Number restrictions (3)

- Given the new concept-forming operators we can now state using  
$$\text{tim} : (\exists_{\leq 2} \text{hasChild}) \sqcap (\exists_{\geq 2} \text{hasChild})$$
that tim has two children.
- But how do we specify that tim is someone who has two male children?
- We could try to use a hasSon relation instead of the hasChild relation, that is,

$$\text{tim} : (\exists_{\leq 2} \text{hasSon}) \sqcap (\exists_{\geq 2} \text{hasSon})$$

but the relations hasSon and hasChild are completely unrelated (which they aren't in the real world.)

- p.5

## Beyond ALC: Number restrictions (4)

- One solution to this problem are qualified number restrictions.
- We extend the language of **ALC** as follows:  
If  $R$  is a role symbol,  $C$  is a concept, and  $n$  is a natural number, then  
 $\exists_{\leq n} R.C$  (there are at most  $n$   $R$ -related elements and belonging to concept  $C$ )  
 $\exists_{\geq n} R.C$  (there are at least  $n$   $R$ -related elements belonging to concept  $C$ )  
are concepts.

- p.6

## Beyond ALC: Number restrictions (5)

The semantics of these additional concept-forming operators is defined by extending the definition of  $\cdot^{\mathcal{I}}$ :

$$(\exists_{\leq n} R.C)^{\mathcal{I}} = \{x \mid |\{y \mid (x, y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}| \leq n\}$$

$$(\exists_{\geq n} R.C)^{\mathcal{I}} = \{x \mid |\{y \mid (x, y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}| \geq n\}$$

The description logic obtained in this way is denoted by **ALCQ**.

- Note that  
$$(\exists_{\leq n} R.T) \text{ and } (\exists_{\leq n} R),$$
as well as  
$$(\exists_{\geq n} R.T) \text{ and } (\exists_{\geq n} R)$$
are equivalent concepts.  
Thus, **ALCQ** is at least as expressive as **ALCN**.

- p.7

## Beyond ALC: Number restrictions (6)

- Given the new concept-forming operators we can now state using  
$$\text{tim} : (\exists_{\leq 2} \text{hasChild.male}) \sqcap (\exists_{\geq 2} \text{hasChild.male})$$
that tim has two male children.
- It is often convenient to use  
$$(\exists_{\leq n} R) \text{ as a shorthand for } (\exists_{\leq n} R.T)$$
  
$$(\exists_{\geq n} R) \text{ as a shorthand for } (\exists_{\geq n} R.T)$$
  
$$(\exists_{=n} R.C) \text{ as a shorthand for } (\exists_{\leq n} R.C) \sqcap (\exists_{\geq n} R.C)$$
  
$$(\exists_{=n} R) \text{ as a shorthand for } (\exists_{\leq n} R.T) \sqcap (\exists_{\geq n} R.T)$$
in **ALCQ**.

- p.8

## Beyond $\mathcal{ALC}$ : Inverse roles (1)

- We can specify that a parent is a person who has a child using the concept definition

$$\text{parent} \doteq \text{person} \sqcap \exists \text{hasChild}.\top$$

- Suppose we wanted to specify that every child has a parent?
- We could try

$$\text{child} \doteq \exists \text{hasParent}.\top$$

but the relations `hasChild` and `hasParent` are completely unrelated (which they aren't in the real world.)

- p.9

## Beyond $\mathcal{ALC}$ : Inverse roles (2)

- One solution to this problem are inverse roles.
- Intuitively, what we do is the following:  
For every binary relation  $R$  there is a binary relation  $R^{-1}$ , the inverse of  $R$ , such that  $(x, y) \in R$  iff  $(y, x) \in R^{-1}$ .  
We can view  $\cdot^{-1}$  as an operator on binary relations that maps any relation  $R$  to the inverse of  $R$ .  
In  $\mathcal{ALC}$ , role symbols represent binary relation.  
We extend  $\mathcal{ALC}$  by an role-forming operator  $\cdot^{-1}$  that represents the inverse operator on binary relations.

- p.10

## Beyond $\mathcal{ALC}$ : Inverse roles (3)

- Formally, we define an extension  $\mathcal{ALCQI}$  of  $\mathcal{ALCQ}$  as follows:  
The set of role terms (or just roles) is inductively defined as follows:

- every role symbol is a role;
- if  $R$  is a role, then  $R^{-1}$  is a role

The set of concept terms (or just concepts) is inductively defined as follows:

- $\top$  and  $\perp$  are concepts;
- every concept symbol is a concept;
- if  $C$  and  $D$  are concepts and  $R$  is a role, then  $\neg C$ ,  $C \sqcup D$ ,  $C \sqcap D$ ,  $\forall R.C$ ,  $\exists R.C$ ,  $\exists_{\leq n} R.C$ , and  $\exists_{\geq n} R.C$  are concepts.

- p.11

## Beyond $\mathcal{ALC}$ : Inverse roles (4)

The definition of concept assertions and concept definitions is identical to that of  $\mathcal{ALC}$ , but role assertions are now expressions of the form

$$(a, b) : R \quad (\text{read 'the pair } (a, b) \text{ belongs to the role } R')$$

where  $a, b$  are object symbols and  $R$  is a role.

Examples:  $\text{parent} \doteq \text{person} \sqcap \exists \text{hasChild}.\top$   
 $\text{child} \doteq \exists \text{hasChild}^{-1}.\text{parent}$   
 $\text{tim} : \text{person}$   
 $(\text{liz}, \text{tim}) : \text{hasChild}^{-1}$



- p.12

## Beyond $\mathcal{ALC}$ : Inverse roles (4)

The definition of concept assertions and concept definitions is identical to that of  $\mathcal{ALC}$ , but role assertions are now expressions of the form

$$(a, b) : R \quad (\text{read 'the pair } (a, b) \text{ belongs to the role } R')$$

where  $a, b$  are object symbols and  $R$  is a role.

Examples:  $\text{parent} \doteq \text{person} \sqcap \exists \text{hasChild}.\top$   
 $\text{child} \doteq \exists \text{hasChild}^{-1}.\text{parent}$   
 $\text{tim} : \text{person}$   
 $(\text{liz}, \text{tim}) : \text{hasChild}^{-1}$



- p.12

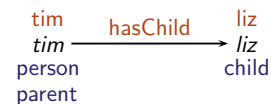
## Beyond $\mathcal{ALC}$ : Inverse roles (4)

The definition of concept assertions and concept definitions is identical to that of  $\mathcal{ALC}$ , but role assertions are now expressions of the form

$$(a, b) : R \quad (\text{read 'the pair } (a, b) \text{ belongs to the role } R')$$

where  $a, b$  are object symbols and  $R$  is a role.

Examples:  $\text{parent} \doteq \text{person} \sqcap \exists \text{hasChild}.\top$   
 $\text{child} \doteq \exists \text{hasChild}^{-1}.\text{parent}$   
 $\text{tim} : \text{person}$   
 $(\text{liz}, \text{tim}) : \text{hasChild}^{-1}$



- p.12

## Exercise 16

Let  $\mathcal{A}_1$  be an  $\mathcal{ALCQI}$  ABox consisting of the two role assertions

$$(\text{tim}, \text{sue}) : \text{hasChild} \quad (\text{tim}, \text{liz}) : \text{hasChild}$$

Which of the following six concept assertions could you add to  $\mathcal{A}_1$  and still obtain a consistent ABox?

- (a)  $\text{tim} : \exists_{=1} \text{hasChild}$       (d)  $\text{sue} : \exists_{=1} \text{hasChild}^{-1}$   
 (b)  $\text{tim} : \exists_{=2} \text{hasChild}$       (e)  $\text{sue} : \exists_{=3} \text{hasChild}^{-1}$   
 (c)  $\text{tim} : \exists_{=3} \text{hasChild}$       (f)  $\text{tim} : \exists_{=1} (\text{hasChild}^{-1})^{-1}$

- p.13

## Exercise 16: Answer

Let  $\mathcal{A}_1$  be an  $\mathcal{ALCQI}$  ABox consisting of the two role assertions

$$(\text{tim}, \text{sue}) : \text{hasChild} \quad (\text{tim}, \text{liz}) : \text{hasChild}$$

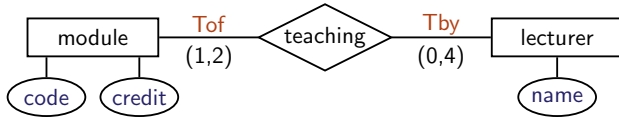
Which of the following six concept assertions could you add to  $\mathcal{A}_1$  and still obtain a consistent ABox?

- (a)  $\text{tim} : \exists_{=1} \text{hasChild}$       (d)  $\text{sue} : \exists_{=1} \text{hasChild}^{-1}$   
 no      yes  
 (b)  $\text{tim} : \exists_{=2} \text{hasChild}$       (e)  $\text{sue} : \exists_{=3} \text{hasChild}^{-1}$   
 yes      yes  
 (c)  $\text{tim} : \exists_{=3} \text{hasChild}$       (f)  $\text{tim} : \exists_{=1} (\text{hasChild}^{-1})^{-1}$   
 yes      no

- p.14

## Application: Representing ER models (1)

We can use *ALCQI* to represent Entity-Relationship Models. Consider the following extract from a ER diagram  $E_1$ :



$$\begin{aligned} \text{teaching} &\stackrel{\dot{=}}{=} (\forall Tof.\text{module}) \sqcap (\exists_{=1} Tof) \sqcap \\ &\quad (\forall Tby.\text{lecturer}) \sqcap (\exists_{=1} Tby) \\ \text{lecturer} &\stackrel{\dot{=}}{=} (\exists \text{name}.\top) \sqcap \\ &\quad (\forall Tby^{-1}.\text{teaching}) \sqcap (\exists_{\geq 0} Tby^{-1}) \sqcap (\exists_{\leq 4} Tby^{-1}) \\ \text{module} &\stackrel{\dot{=}}{=} (\exists \text{code}.\top) \sqcap (\exists \text{credit}.\top) \sqcap \\ &\quad (\forall Tof^{-1}.\text{teaching}) \sqcap (\exists_{\geq 1} Tof^{-1}) \sqcap (\exists_{\leq 2} Tof^{-1}) \end{aligned}$$

- p.15

## Application: Representing ER models (2)

- Note that *ALCQI* is more expressive than Entity-Relationship Models.

In analogy to the previous example, we could specify that labs are either supervised by lecturers or postgraduates:

$$\text{supervising} \stackrel{\dot{=}}{=} (\forall \text{Sof}.\text{lab}) \sqcap (\exists_{=1} \text{Sof}) \sqcap (\forall \text{Sby}.\text{(lecturer} \sqcup \text{postGrad)}) \sqcap (\exists_{=1} \text{Sby})$$

This cannot be expressed in an ER model.

- p.16

## Application: Representing ER models (3)

- Even more important is the absence of negation/complement operators in ER models. By extending the definition of the concept supervision, we can specify that undergraduates are not supervising labs:

$$\begin{aligned} \text{supervising} &\stackrel{\dot{=}}{=} (\forall \text{Sof}.\text{lab}) \sqcap (\exists_{=1} \text{Sof}) \sqcap \\ &\quad (\forall \text{Sby}.\text{((lecturer} \sqcup \text{postGrad}) \sqcap \neg \text{underGrad})) \sqcap \\ &\quad (\exists_{=1} \text{Sby}) \end{aligned}$$

- p.17

## Application: Representing databases

Once we have represented an ER model in *ALCQI* we can also represent the corresponding database. For example:

simon : lecturer  
(simon, 'Simon') : name  
(ta1, simon) : Tby      (ta1, c304) : Tof

ullrich : lecturer  
(ullrich, 'Ullrich') : name  
(ta2, ullrich) : Tby      (ta2, c304) : Tof

c304 : module  
(c304, 304) : code      (c304, 15) : credit

- p.18

## Application: Consistency and query processing

- Consistency of the database with the ER model  
 $\rightsquigarrow$  Consistency of an ABox wrt. to a TBox  
 Answer: Yes, the current ABox is consistent wrt. to the TBox
- Query processing  
 Example: Which lecturers are teaching this semester?  
 $\rightsquigarrow$  Retrieval for the concept  $\text{lecturer} \sqcap \exists Tby^{-1}.\text{teaching}$   
 Answer: {simon, ullrich}

- p.19

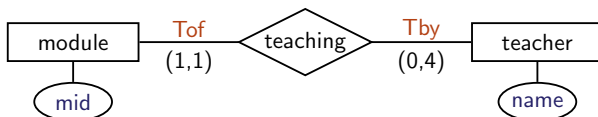
## Application: Query optimisation

- (Logical) Query optimisation  
 Example: Give me all lecturers who are currently teaching and also all lecturers who are currently not teaching.  
 $\rightsquigarrow$  1. Equivalence of concepts:  
 Query:  $(\text{lecturer} \sqcap \exists Tby^{-1}.\text{teaching}) \sqcup (\text{lecturer} \sqcap \neg(\exists Tby^{-1}.\text{teaching}))$   
 is equiv. to  $\text{lecturer} \sqcap ((\exists Tby^{-1}.\text{teaching}) \sqcup \neg(\exists Tby^{-1}.\text{teaching}))$   
 is equiv. to  $\text{lecturer} \sqcap \top$   
 is equiv. to  $\text{lecturer}$   
 2. Retrieval for the concept  $\text{lecturer}$   
 Answer: {simon, ullrich}

- p.20

## Application: Data integration (1)

- Data integration  
 Assume that we want to integrate two databases, one described by the ER diagram  $E_1$ , and a second one described by the ER diagram  $E_2$  below:



We assume that 'teacher' in  $E_2$  corresponds to 'lecturer' in  $E_1$ , and that 'mid' in  $E_2$  corresponds to 'code' in  $E_1$ .

We integrate the ER models by combining the concept definitions corresponding to  $E_1$  and  $E_2$ .

- p.21

## Application: Data integration (2)

For example, we can combine the two definitions of 'module'

$$\begin{aligned} \text{module}_{E_1} &\stackrel{\dot{=}}{=} (\exists \text{code}.\top) \sqcap (\exists \text{credit}.\top) \sqcap \\ &\quad (\forall Tof^{-1}.\text{teaching}) \sqcap (\exists_{\geq 1} Tof^{-1}) \sqcap (\exists_{\leq 2} Tof^{-1}) \\ \text{module}_{E_2} &\stackrel{\dot{=}}{=} (\exists \text{mid}.\top) \sqcap \\ &\quad (\forall Tof^{-1}.\text{teaching}) \sqcap (\exists_{\geq 1} Tof^{-1}) \sqcap (\exists_{\leq 1} Tof^{-1}) \\ \text{as} \\ \text{module} &\stackrel{\dot{=}}{=} (((\exists \text{code}.\top) \sqcap (\exists \text{credit}.\top)) \sqcap (\exists \text{code}.\top)) \sqcap \\ &\quad (\forall Tof^{-1}.\text{teaching}) \sqcap (\exists_{\geq 1} Tof^{-1}) \sqcap (\exists_{\leq 2} Tof^{-1}) \sqcap \\ &\quad (\forall Tof^{-1}.\text{teaching}) \sqcap (\exists_{\geq 1} Tof^{-1}) \sqcap (\exists_{\leq 1} Tof^{-1}) \\ \text{which is equivalent to the simplified definition} \\ \text{module} &\stackrel{\dot{=}}{=} (\exists \text{code}.\top) \sqcap (\exists \text{credit}.\top) \sqcap \\ &\quad (\forall Tof^{-1}.\text{teaching}) \sqcap (\exists_{\geq 1} Tof^{-1}) \sqcap (\exists_{\leq 1} Tof^{-1}) \end{aligned}$$

- p.22

## Application: Data integration (3)

Once we have combined the the two ER models, we can also combine the two databases by combining the corresponding ABoxes.

The combined ABoxes still contain the assertions below

$(ta1, c304) : \text{Tof}$      $(ta2, c304) : \text{Tof}$      $c304 : \text{module}$

We can now check whether the combined ABoxes are consistent wrt. to the TBox, and we find that there is a conflict with

$$\text{module} \sqsubseteq (\exists \text{code}.\top) \sqcap (\exists \text{credit}.\top) \sqcap (\forall \text{Tof}^{-1}.\text{teaching}) \sqcap (\exists_{\geq 1} \text{Tof}^{-1}) \sqcap (\exists_{\leq 1} \text{Tof}^{-1})$$

which states that there can be at most one element related to a module by the  $\text{Tof}^{-1}$  relation.

- p.23

## Beyond $\mathcal{ALC}$ : Enumeration concepts (1)

- Coming back to the task of query processing, we can see that our ability to formulate queries is rather limited.

For example, we have no expressions for

Who is teaching module c304?

or Who is teaching the module with code 304?

- To express such queries we can use enumeration concepts:

We extend the language of  $\mathcal{ALCQI}$  by

- If  $a_1, \dots, a_n$  are object symbols then  $\{a_1, \dots, a_n\}$  (the enumeration concept consisting of  $a_1, \dots, a_n$ ) is a concept.

The language obtained is called  $\mathcal{ALCQIO}$ .

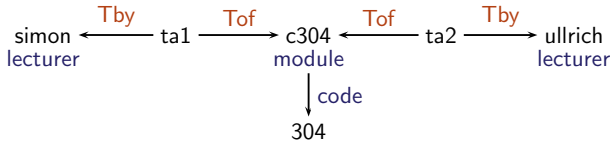
- p.24

## Beyond $\mathcal{ALC}$ : Enumeration concepts (2)

The semantics of enumeration concepts is defined by extending the definition of  $\bar{\cdot}$  as follows:

$$\{a_1, \dots, a_n\}^{\bar{\cdot}} = \{a_1^{\bar{\cdot}}, \dots, a_n^{\bar{\cdot}}\}$$

- Then the two example queries can be expressed by

$$\exists Tby^{-1}.\exists Tof.\{c304\} \quad \text{and} \quad \exists Tby^{-1}.\exists Tof.\exists \text{code}.\{304\}.$$


and the answer, obtained by retrieval, is  $\{\text{simon}, \text{ullrich}\}$  for both.

- p.25

## Beyond $\mathcal{ALC}$ : Range restrictions (1)

- Reconsider the problem of specifying that tim is someone who has two male children.
- Instead of using qualified number restrictions we can use an additional role-forming operator: range restriction.
- In analogy to the extension of  $\mathcal{ALCQ}$  by the inverse operator, we define that

- if  $R$  is a role and  $C$  is a concept, then  $R \downarrow C$  (the restriction of the range of  $R$  to  $C$ ) is a role.

The semantics of this additional operator is again defined by extending the definition of  $\bar{\cdot}$ :

$$(R \downarrow C)^{\bar{\cdot}} = \{(x, y) \mid (x, y) \in R^{\bar{\cdot}} \text{ and } y \in C^{\bar{\cdot}}\}$$

- p.26

## Beyond $\mathcal{ALC}$ : Range restrictions (2)

- Given the new role-forming operator we can now state using  $\text{tim} : (\exists_{\leq 2} \text{hasChild} \downarrow \text{male}) \sqcap (\exists_{\geq 2} \text{hasChild} \downarrow \text{male})$  that tim has two male children.

- p.27

## Beyond $\mathcal{ALC}$ : Role intersection (1)

- Suppose diz is a lecturer. He is teaching some students and he has some students as personal tutees. How can we express that there are two students which are personal tutees of diz and he is also teaching them?

- The concept assertion

$$\text{diz} : (\exists_{=2} \text{isTeaching}.\text{student}) \sqcap (\exists_{=2} \text{hasTutee}.\text{student})$$

does not what we want.

It states that diz is teaching exactly two student and that he has exactly two students as personal tutees, but these pairs of students do not need to be identical.

- p.28

## Beyond $\mathcal{ALC}$ : Role intersection (2)

- A solution of this problem is role intersection.
- We again extend our definition of the set of roles by defining
  - if  $R$  and  $S$  are roles, then  $R \sqcap S$  (the intersection of  $R$  and  $S$ ) is a role.

The semantics of role intersection is defined as follows:

$$(R \sqcap S)^{\bar{\cdot}} = R^{\bar{\cdot}} \sqcap S^{\bar{\cdot}} = \{(x, y) \mid (x, y) \in R^{\bar{\cdot}} \text{ and } (x, y) \in S^{\bar{\cdot}}\}$$

- Now we can state the desired property of diz by

$$\text{diz} : (\exists_{=2} (\text{isTeaching} \sqcap \text{hasTutee}).\text{student})$$

- p.29

## Beyond $\mathcal{ALC}$ : Role complement (1)

- Assume that we want to define the concept of a cheese lover as a person who eats any kind of cheese.

- The concept definition

$$\text{cheeseLover} \doteq \text{person} \sqcap \forall \text{eats}.\text{cheese}$$

does not exactly what we want, since according to this definition a cheese lover eats nothing but cheese, which still leaves the possibility that there are some kinds of cheese that a cheese lover does not eat.

- p.30

## Beyond $\mathcal{ALC}$ : Role complement (2)

- In this case, we need a role complement operator for the correct definition.
- If  $R$  is a role, then  $\neg R$  (the complement of  $R$ ) is a role, and the semantics of role complement is defined by:

$$\begin{aligned}(\neg R)^{\overline{\mathcal{I}}} &= (\Delta \times \Delta) \setminus R^{\overline{\mathcal{I}}} \\ &= \{(x, y) \mid (x, y) \in \Delta \times \Delta \text{ and } (x, y) \notin R^{\overline{\mathcal{I}}}\}\end{aligned}$$

- Then the correct concept definition for a cheese lover is given by  $\text{cheeseLover} \doteq \text{person} \sqcap \forall \neg \text{eats} . \neg \text{cheese}$   
that is, a cheese lover is a person such that all (s)he does not eat is not cheese.

– p.31

## Beyond $\mathcal{ALC}$ : Role union (2)

- Another role-forming operator one may want to add to a description logic is role union.
- If  $R$  and  $S$  are roles, then  $R \sqcup S$  (the union of  $R$  and  $S$ ) is a role, and its semantics is given by

$$\begin{aligned}(R \sqcup S)^{\overline{\mathcal{I}}} &= R^{\overline{\mathcal{I}}} \cup S^{\overline{\mathcal{I}}} \\ &= \{(x, y) \mid (x, y) \in R^{\overline{\mathcal{I}}} \text{ or } (x, y) \in S^{\overline{\mathcal{I}}}\}\end{aligned}$$

- It is straightforward to check that

$$\forall (R \sqcup S).C \text{ is equivalent to } (\forall R.C) \sqcap (\forall S.C)$$

$$\exists (R \sqcup S).C \text{ is equivalent to } (\exists R.C) \sqcup (\exists S.C)$$

- Thus, role union does not add to the expressive power of  $\mathcal{ALC}$  or any of its extensions.

– p.32

## Beyond $\mathcal{ALC}$ : Role definitions

- We have now seen a number of role-forming operators: role intersection, role union, role complement, range restriction, and inverse roles, which can be used to build complex role terms.
- In analogy to concept definitions we can extend description logics by role definitions, that is, expressions of the form
  - ▶  $R \sqsubseteq S$  (read ' $R$  is defined to be a subrole of  $S$ ')
  - ▶  $R \doteq S$  (read ' $R$  is defined to be equal to  $S$ ')where  $R$  is a role symbol and  $S$  a role term.

- We say a terminological interpretation  $\mathcal{I}$  satisfies

- ▶ a role definition  $R \sqsubseteq S$  iff  $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
- ▶ a role definition  $R \doteq S$  iff  $R^{\mathcal{I}} = S^{\mathcal{I}}$

– p.33

## Beyond $\mathcal{ALC}$ : Conclusion (1)

- We have seen a number of description logics that are defined by extending  $\mathcal{ALC}$  with additional operators
- The choice of the appropriate description logic is determined by the application that is considered. Commonly, description logics are specifically tailored for a particular application.
- As we move to more expressive description logics the problem of deciding the coherence of a concept or the consistency of an ABox wrt. a TBox becomes more complex. Eventually, these problems become undecidable.

– p.34

## Beyond $\mathcal{ALC}$ : Conclusion (2)

- Most description logics correspond to fragments of first-order logic and description logics can be seen as variable-free languages for these fragments.
- Again, as we move to more expressive description logics, the absence of variables can become a burden and we may be better off with a first-order language.

– p.35

## Description logics: Summary

- Syntax: Concepts, roles, concept definitions, TBoxes, ABoxes, knowledge bases
- Semantics: Terminological interpretations, interpretation of concept terms, concept definitions, and knowledge bases
- Checking terminological interpretations
- Labelled directed graphs for interpretations and ABoxes
- Inferential services, reduction to satisfiability
- Checking the consistency of an ABox wrt. a TBox: Expansion of a knowledge base, negation normal form, completion rules
- Complexity of  $\mathcal{ALC}$ , Relationship of  $\mathcal{ALC}$  to first-order logic, multi-modal logic and hybrid logics
- Extensions of  $\mathcal{ALC}$

– p.36

## What comes next?

- **MSc level**
  - ▶ Detailed algorithms for model checking and deductive systems and a discussion of implementational issues
  - ▶ Understanding the proofs of all the theorems and the concepts needed for these proofs
  - ▶ Case studies of existing applications
- **PhD level**
  - ▶ Inventing new algorithms for model checking and deductive systems and inventing new optimisation techniques
  - ▶ Proving new theorems
  - ▶ Application of formal methods to new applications

– p.37