

# Applications of Description Logics: Intelligent Conceptual Modeling

## [2, supervised Lab]

Uli Sattler

### General Remarks

In this session,

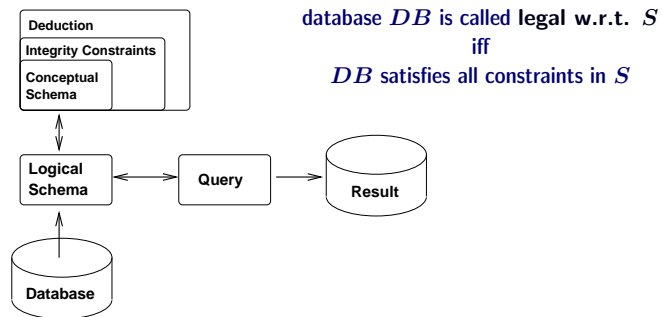
- I assume that you have understood syntax and semantic of the basic description logic (DL)  $\mathcal{ALC}$  and
- show you a nice application of DLs: to support conceptual modeling, e.g., of databases.

To this purpose, I will

- re-fresh your memory of **entity-relationship diagrams** (ER-diagrams): what they are and what they mean,
- explain an extension of  $\mathcal{ALC}$  that is necessary for this application, namely an extension with **inverse roles and number restrictions**, and
- show how an ER-diagram can be translated into a DL knowledge base and what the benefit is

### ER diagrams: a formalism to specify a Conceptual Schema

- A Conceptual Schema**
- is a formal conceptualisation of the world
  - specifies a set of constraints, which declare what should necessarily hold in a **database**
  - Given a conceptual schema  $S$ , a



### ER diagrams

- ER-diagrams**
- formalisms for representing conceptual schemas, e.g., of databases
  - are **graphs** where nodes stand for
    - **entities**: unary predicates (drawn as rectangles) and
    - **relations**:  $\geq 2$ -ary predicates (drawn as trapezes)
  - here, we restrict our attention to **unary and binary** relations: this makes our life easier, but is not necessary: the whole approach you will see in the following works for relations of unrestricted arity
  - in the following, a

database  $DB$  over relations  $E_1, \dots, E_k, R_1, \dots, R_\ell$

is given by

- a domain  $\Delta^{DB}$ ,
- unary relations (sets)  $E_i^{DB} \subseteq \Delta^{DB}$ , and
- binary relations  $R_j^{DB} \subseteq \Delta^{DB} \times \Delta^{DB}$

## ER diagrams

- Clearly, databases are (a special case of) FOL interpretations (see slide 23) and thus we know what  $DB \models \phi$  means
- Next, we define a translation  $\pi$  from an ER-diagram  $S$  into FOL formulae such that

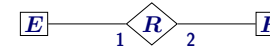
$DB$  is legal w.r.t.  $S$  iff  $DB \models \pi(S)$

and, for any database  $DB$  legal w.r.t.  $S$ , we have

$(a_1, \dots, a_n) \in R^{DB}$  iff  $DB \models R(a_1, \dots, a_n)$

## ER diagrams, their Semantics, and their FOL Translation: Relations

If an ER-diagram  $S$  contains a constraint  $C$  of the form



then this means that, in any database  $DB$  legal w.r.t.  $S$ , we have

$$R^{DB} \subseteq E^{DB} \times F^{DB},$$

which is enforced by the following FOL formula  $\pi(C)$ :

$$\forall x_1, x_2 : R(x_1, x_2) \Rightarrow E(x_1) \wedge F(x_2)$$

i.e., if a database  $DB$  is legal w.r.t.  $C$ , then  $DB \models \pi(C)$ .

## ER diagrams, their Semantics, and their FOL Translation: Attributes

If an ER-diagram  $S$  contains, for  $D$  a “concrete domain” such as String, Integer, etc., the following constraint  $C$ :



then this means that, in any database  $DB$  legal w.r.t.  $S$ , we have

$$E^{DB} \subseteq \{e \in \Delta^{DB} \mid f_A(e) \in D\},$$

for a function  $f_A$ , which is enforced by the following FOL formula  $\pi(C)$ :

$$\forall x : E(x) \Rightarrow (\exists y. f_A(x) = y \wedge D(y))$$

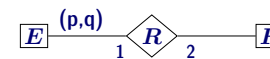
where Strings, Integers, etc., are axiomatised, e.g.,

$$\forall x. (L(x) \Rightarrow \neg(\text{String}(x) \vee \text{Integer}(x))) \wedge (\text{String}(x) \Rightarrow \neg \text{Integer}(x))$$

Please note that  $A$  being an attribute means that each instance of  $E$  has **exactly one** object related to it via  $A$ , which is why we translate it into a function  $f_A$

## ER diagrams, their Semantics, and their FOL Translation: Cardinality Restrictions

If an ER-diagram  $S$  contains the following constraint  $C$ :



then this means that, in any database  $DB$  legal w.r.t.  $S$ , we have

$$E^{DB} \subseteq \{e \in \Delta^{DB} \mid p \leq \#\{b \mid (e, b) \in R^{DB}\} \leq q\},$$

which is enforced by the following FOL formula  $\pi(C)$ :

$$\forall x_1 : E(x_1) \Rightarrow (\exists^{\geq p} x_2. R(x_1, x_2) \wedge \exists^{\leq q} x_2. R(x_1, x_2))$$

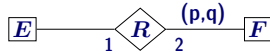
where  $\exists^{\geq p}$  and  $\exists^{\leq q}$  are FOL counting quantifiers, i.e.,

$$\exists^{\geq p} x. \psi(x) \equiv \exists x_1, \dots, x_p. \left( \bigwedge_{1 \leq i < j \leq p} x_i \neq x_j \wedge \bigwedge_{1 \leq i \leq p} \psi(x_i) \right)$$

$$\exists^{\leq q} x. \psi(x) \equiv \forall x_1, \dots, x_{q+1}. \left( \bigwedge_{1 \leq i \leq q+1} \psi(x_i) \Rightarrow \bigvee_{1 \leq i < j \leq q+1} x_i = x_j \right)$$

ER diagrams, their Semantics, and their FOL Translation: Cardinality Restrictions

(Similar:) if an ER-diagram  $S$  contains the following constraint  $C$ :



then this means that, in any database  $DB$  legal w.r.t.  $S$ , we have

$$F^{DB} \subseteq \{e \in \Delta^{DB} \mid p \leq \#\{b \mid (b, e) \in R^{DB}\} \leq q\},$$

which is enforced by the following FOL formula  $\pi(C)$ :

$$\forall x_1 : F(x_1) \Rightarrow (\exists^{\geq p} x_2. R(x_2, x_1) \wedge \exists^{\leq q} x_2. R(x_2, x_1))$$

**Important:** cardinality constraints  $(p, q)$  are allowed for  $p \in \mathbb{N}$  and  $q \in \mathbb{N} \cup \{n\}$   
 If  $q = n$ , this translates to no upper bound

ER diagrams — Cardinality Restrictions: an Example



A valid Database is:

| Professor          | Student          | Supervises         |                  |
|--------------------|------------------|--------------------|------------------|
| <i>professorId</i> | <i>studentId</i> | <i>professorId</i> | <i>studentId</i> |
| Alexa              | John             | Alexa              | John             |
| Bob                | Mary             | Bob                | Laura            |
|                    | Nick             | Alexa              | Mary             |
|                    | Paul             | Bob                | Nick             |
|                    | Laura            | Alexa              | Paul             |

ER diagrams — Cardinality Restrictions: an Example



An invalid Database is:

| Professor          | Student          | Supervises         |                  |
|--------------------|------------------|--------------------|------------------|
| <i>professorId</i> | <i>studentId</i> | <i>professorId</i> | <i>studentId</i> |
| Alexa              | John             | Alexa              | John             |
| Bob                | Mary             | Bob                | Laura            |
|                    | Nick             | Alexa              | Mary             |
|                    | Paul             | Bob                | Nick             |
|                    | Laura            | Alexa              | Paul             |
|                    |                  | Alexa              | Laura            |

ER diagrams — Cardinality Restrictions: an Example

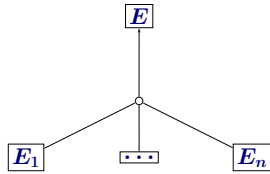


The FOL translation of the whole ER-diagram is:

$$\begin{aligned} &\forall x, y. \text{Supervises}(x, y) \Rightarrow \text{Professor}(x) \wedge \text{Student}(y) \\ &\forall x. \text{Professor}(x) \Rightarrow \exists^{\geq 2} y. \text{Supervises}(x, y) \\ &\forall y. \text{Student}(y) \Rightarrow \exists^{\geq 1} x. \text{Supervises}(x, y) \wedge \exists^{\leq 1} x. \text{Supervises}(x, y) \end{aligned}$$

### ER diagrams: IS-A Relations

If an ER-diagram  $S$  contains the following constraint  $C$ :



then this means that, in any database  $DB$  legal w.r.t.  $S$ , we have

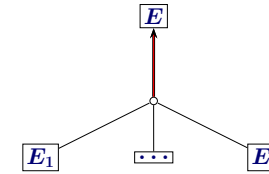
$$E_i^{DB} \subseteq E^{DB}, \text{ for all } i = 1, \dots, n.$$

which is enforced by the following FOL formula  $\pi(C)$ :

$$\bigwedge_{1 \leq i \leq n} \forall x. E_i(x) \Rightarrow E(x)$$

### ER diagrams: Covering IS-A Relations

If an ER-diagram  $S$  contains, additionally, the following constraint  $C$ :



then this means that, in any database  $DB$  legal w.r.t.  $S$ , we have additionally

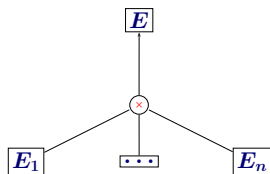
$$E^{DB} \subseteq \bigcup_{1 \leq i \leq n} E_i^{DB}$$

which is enforced by the following FOL formula  $\pi(C)$ :

$$\forall x. E(x) \Rightarrow \bigvee_{1 \leq i \leq n} E_i(x)$$

### ER diagrams: Disjoint IS-A Relations

If an ER-diagram  $S$  contains, additionally, the following constraint  $C$ :



then this means that, in any database  $DB$  legal w.r.t.  $S$ , we have additionally

$$E_i^{DB} \cap E_j^{DB} = \emptyset, \text{ for all } 1 \leq i < j \leq n$$

which is enforced by the following FOL formula  $\pi(C)$ :

$$\bigwedge_{1 \leq i < j \leq n} \forall x. E_i(x) \Rightarrow \neg E_j(x)$$

### ER diagrams: Summary

An ER-diagram  $S$  is translated into a finite set  $\pi(S)$  of FOL formulae comprising

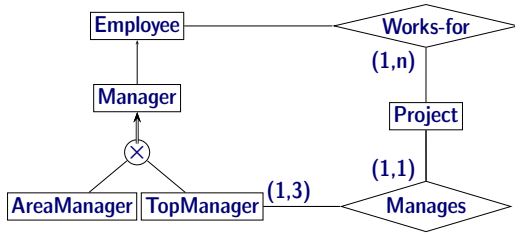
- all translations  $\pi(C)$  of
- all constraints  $C$  in  $S$ .

Then we can show that, for each database  $DB$ , we have

$$DB \text{ is legal w.r.t. } S \text{ iff } DB \models \pi(S)$$

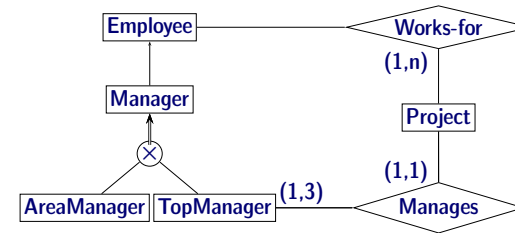
(for the definition of  $DB \models \pi(S)$ , see slide 26)

### ER diagrams – another Example



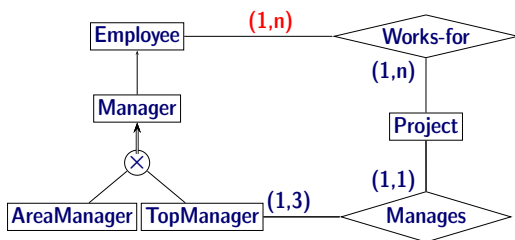
$\forall x, y. \text{Works-for}(x, y) \Rightarrow \text{Employee}(x) \wedge \text{Project}(y)$   
 $\forall x, y. \text{Manages}(x, y) \Rightarrow \text{Top-Manager}(x) \wedge \text{Project}(y)$   
 $\forall y. \text{Project}(y) \Rightarrow \exists x. \text{Works-for}(x, y)$   
 $\forall y. \text{Project}(y) \Rightarrow \exists^{\leq 1} x. \text{Manages}(x, y) \wedge \exists^{\geq 1} x. \text{Manages}(x, y)$   
 $\forall x. \text{Top-Manager}(x) \Rightarrow \exists^{\geq 1} y. \text{Manages}(x, y) \wedge \exists^{\leq 3} y. \text{Manages}(x, y)$

### ER diagrams – another Example (ctd.)



$\forall x. \text{Manager}(x) \Rightarrow \text{Employee}(x)$   
 $\forall x. \text{Manager}(x) \Rightarrow \text{Area-Manager}(x) \vee \text{Top-Manager}(x)$   
 $\forall x. \text{Area-Manager}(x) \Rightarrow \text{Manager}(x)$   
 $\forall x. \text{Top-Manager}(x) \Rightarrow \text{Manager}(x)$   
 $\forall x. \text{Area-Manager}(x) \Rightarrow \neg \text{Top-Manager}(x)$

### ER-diagrams – another Example with Additional (Integrity) Constraints



- Managers do not work for a project (she/he just manages it):  
 $\forall x. \text{Manager}(x) \Rightarrow \forall y. \neg \text{Works-For}(x, y)$
- If, additionally, the minimum cardinality for the participation of employees to the Works-For relationship is increased, then, TopManager becomes unsatisfiable
- If an IS-A link is added stating that every AreaManager is a TopManager, then
  - AreaManager becomes unsatisfiable and
  - every Manager is a TopManager

### ER-diagrams: Reasoning Problems and Inference Services

As we have just seen,

- each ER-diagram  $S$  can be translated into a finite set of FOL formulae  $\pi(S)$  and
- additional integrity constraints can be expressed,
- which might lead to the inconsistency of a conceptual model...

**Inferences:** a finite set  $\Gamma$  of FOL formulae obtained by the translation of an ER-diagram  $S$  plus possibly some FOL formulae  $I$  as additional integrity constraints is called a **FOL conceptual model**.

For  $\Gamma$  a FOL conceptual model, and  $X, Y$  entities or relationships, we say that

- $X$  is **inconsistent** in  $\Gamma$  if  $DB \models \Gamma$  implies  $X^{DB} = \emptyset$   
 iff  $\Gamma \cup \{\exists \vec{x}. X(\vec{x})\}$  is **not satisfiable**
- $X$  is **subsumed** by  $Y$  in  $\Gamma$  if  $DB \models \Gamma$  implies  $X^{DB} \subseteq Y^{DB}$   
 iff  $\Gamma \models \forall \vec{x}. (X(\vec{x}) \Rightarrow Y(\vec{x}))$

## ER-diagrams: Reasoning Problems and Inference Services

Clearly, to verify your conceptual model, it would be nice to **automatically**

- test each entity and relationship for (in)consistency and
- test each pair of entities and each pair of relationships for subsumption and
- notify the user about each
  - inconsistency and
  - subsumption that is not **explicit** in the ER-diagram, where a subsumption relation between  $X$  and  $Y$  is **explicit** if there is a **path** of IS-A links between  $X$  and  $Y$  in the ER-diagram
- unfortunately, as we have seen before, we have no decision procedure for these inferences: they translate to (un)satisfiability and consequence, which are both undecidable for FOL.

What to do?

- see whether we really need **full** FOL for ER-diagrams and additional integrity constraints

## ER diagrams and their Description Logic Translation

As we will see next, we can translate ER-diagrams into **Description Logic TBoxes**

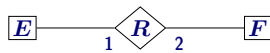
However, the DL  $\mathcal{ALC}$  is not expressive enough for this: in ER-diagrams, we

- constrain the **domain and range** of relations – in  $\mathcal{ALC}$ , we can only constrain the range, e.g., using  $\top \sqsubseteq \forall R.F$ .  
Hence we extend  $\mathcal{ALC}$  with the possibility to “turn around”  $R$  to  $R^-$  and also allow, e.g.,  $\top \sqsubseteq \forall R^-.E$ ,  
i.e., we allow for **inverse roles** in the place of role names
- use cardinality constraints  $(p, q)$  on relations – in  $\mathcal{ALC}$ , we cannot **count**.  
Hence we extend  $\mathcal{ALC}$  with **number restrictions**, i.e., concepts of the form  $(\leq n R)$  and  $(\geq n R)$  for  $n \in \mathbb{N}$  and  $R$  a possibly inverse role

We can show that the subsumption and the satisfiability problem for the extension of  $\mathcal{ALC}$  with inverse roles and number restrictions,  $\mathcal{ALCIN}$ , are still decidable.

## ER diagrams and their Description Logic Translation: Relations

If an ER-diagram  $S$  contains a constraint  $C$  of the form



then this means that, in any database  $DB$  legal w.r.t.  $S$ , we have

$$R^{DB} \subseteq E^{DB} \times F^{DB},$$

which can be enforced by the following DL axiom  $\pi'(C)$ :

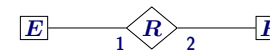
$$\top \sqsubseteq \forall R.F \sqcap \forall R^-.E$$

where  $R^-$  is the **inverse** of  $R$  and all interpretations must satisfy

$$(R^-)^{\mathcal{I}} = \{(d, e) \mid (e, d) \in R^{\mathcal{I}}\}$$

## ER diagrams and their Description Logic Translation: Relations

If an ER-diagram  $S$  contains a constraint  $C$  of the form



...how can we understand

$$\top \sqsubseteq \forall R.F \sqcap \forall R^-.E?$$

First, let's see what it means for an interpretation  $DB$  to satisfy the above axiom:

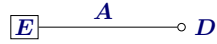
$$\Delta^{DB} \subseteq \{e \mid \forall f. (e, f) \in R^{DB} \Rightarrow f \in F^{DB}\} \cap \{e \mid \forall f. \underbrace{(e, f) \in (R^-)^{DB}}_{(f, e) \in R^{DB}} \Rightarrow f \in E^{DB}\}$$

which is the same as to say

$$\forall e, f \in \Delta^{DB} : (e, f) \in R^{DB} \Rightarrow (e \in E^{DB} \wedge f \in F^{DB})$$

### ER diagrams and their DL Translation: Attributes

If an ER-diagram  $S$  contains, for  $D$  a “concrete domain” such as String, Integer, etc., the following constraint  $C$ :



then this means that, in any database  $DB$  legal w.r.t.  $S$ , we have

$$E^{DB} \subseteq \{e \in \Delta^{DB} \mid f_A(e) \in D\},$$

which can be enforced by the following DL axiom  $\pi'(C)$ :

$$E \dot{\sqsubseteq} \exists A.D \sqcap (\leq 1 A)$$

where  $(\leq 1 A)$  is a **number restriction** and all interpretations  $\mathcal{I}$  must satisfy

$$(\leq 1 A)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \#\{e \mid (d, e) \in A^{\mathcal{I}}\} \leq 1\}$$

Moreover, Strings, Integers, etc., are axiomatised, e.g.,

$$E \dot{\sqsubseteq} L, \quad L \dot{\sqsubseteq} \neg(\text{String} \sqcup \text{Integer}), \quad \text{String} \dot{\sqsubseteq} \neg \text{Integer}$$

### ER diagrams and their DL Translation: Cardinality Restrictions

If an ER-diagram  $S$  contains the following constraint  $C$ :



then this means that, in any database  $DB$  legal w.r.t.  $S$ , we have

$$E^{DB} \subseteq \{e \in \Delta^{DB} \mid p \leq \#\{b \mid (e, b) \in R^{DB}\} \leq q\},$$

which can be enforced by the following DL axiom  $\pi'(C)$ :

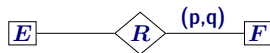
$$E \dot{\sqsubseteq} (\geq p R) \sqcap (\leq q R)$$

where  $(\geq p R)$ ,  $(\leq q R)$  are **number restrictions**, with the following semantics:

$$\begin{aligned} (\geq p R)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \#\{e \mid (d, e) \in R^{\mathcal{I}}\} \geq p\} \\ (\leq q R)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \#\{e \mid (d, e) \in R^{\mathcal{I}}\} \leq q\} \end{aligned}$$

### ER diagrams and their DL Translation: Cardinality Restrictions

(Similar:) if an ER-diagram  $S$  contains the following constraint  $C$ :



then this means that, in any database  $DB$  legal w.r.t.  $S$ , we have

$$\begin{aligned} F^{DB} \subseteq \{e \in \Delta^{DB} \mid p \leq \#\{b \mid (b, e) \in R^{DB}\} \leq q\} = \\ \{e \in \Delta^{DB} \mid p \leq \#\{b \mid (e, b) \in (R^-)^{DB}\} \leq q\}, \end{aligned}$$

which can be enforced by the following DL axiom  $\pi'(C)$ :

$$F \dot{\sqsubseteq} (\geq p R^-) \sqcap (\leq q R^-)$$

### ER diagrams — Cardinality Restrictions: an Example

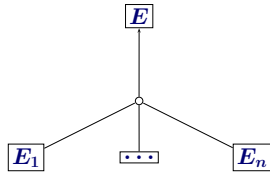


The DL translation of the whole ER-diagram is:

$$\begin{aligned} \top \dot{\sqsubseteq} \forall \text{Supervises.Student} \sqcap \forall \text{Supervises}^- . \text{Professor} \\ \text{Professor} \dot{\sqsubseteq} (\geq 2 \text{Supervises}) \\ \text{Student} \dot{\sqsubseteq} (\geq 1 \text{Supervises}^-) \sqcap (\leq 1 \text{Supervises}^-) \end{aligned}$$

### ER diagrams: IS-A Relations

If an ER-diagram  $S$  contains the following constraint  $C$ :



then this means that, in any database  $DB$  legal w.r.t.  $S$ , we have

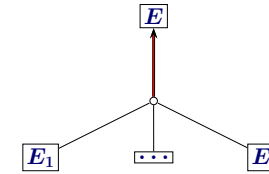
$$E_i^{DB} \subseteq E^{DB}, \text{ for all } i = 1, \dots, n.$$

which is enforced by the following set of DL axioms  $\pi'(C)$ :

$$\bigcup_{1 \leq i \leq n} \{E_i \dot{\subseteq} E\}$$

### ER diagrams: Covering IS-A Relations

If an ER-diagram  $S$  contains, additionally, the following constraint  $C$ :



then this means that, in any database  $DB$  legal w.r.t.  $S$ , we have additionally

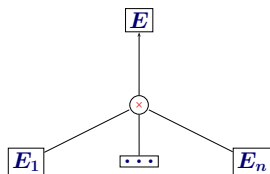
$$E^{DB} \subseteq \bigcup_{1 \leq i \leq n} E_i^{DB}$$

which is enforced by the following DL axiom  $\pi'(C)$ :

$$E \dot{\subseteq} E_1 \sqcup \dots \sqcup E_n$$

### ER diagrams: Disjoint IS-A Relations

If an ER-diagram  $S$  contains, additionally, the following constraint  $C$ :



then this means that, in any database  $DB$  legal w.r.t.  $S$ , we have additionally

$$E_i^{DB} \cap E_j^{DB} = \emptyset, \text{ for all } 1 \leq i < j \leq n$$

which is enforced by the following set of DL axioms  $\pi'(C)$ :

$$\bigcup_{1 \leq i < j \leq n} \{E_i \dot{\subseteq} \neg E_j\}$$

### ER diagrams and DL: Summary

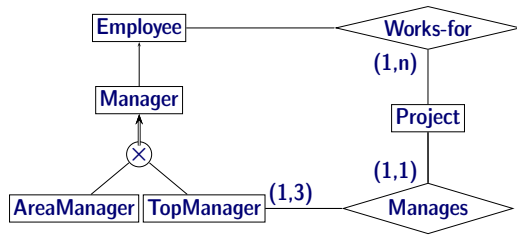
An ER-diagram  $S$  is translated into a finite set  $\pi'(S)$  of  $\mathcal{ALCIN}$  axioms comprising

- all translations  $\pi'(C)$  of
- all constraints  $C$  in  $S$ .

Then we can show that, for each database  $DB$ , we have

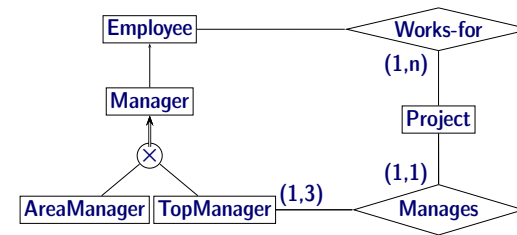
$$DB \text{ is legal w.r.t. } S \text{ iff } DB \models \pi'(S)$$

### ER diagrams and DLs – another Example



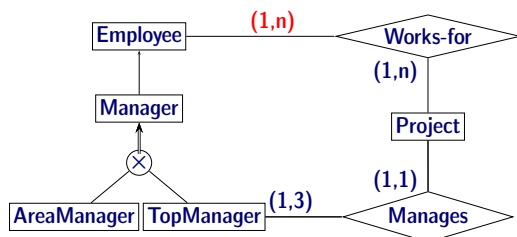
$\mathcal{T} \sqsubseteq \forall \text{Works-for}.\text{Project} \sqcap \forall \text{Works-for}^{\neg}.\text{Employee}$   
 $\mathcal{T} \sqsubseteq \forall \text{Manages}.\text{Project} \sqcap \forall \text{Manages}^{\neg}.\text{Top-Manager}$   
 $\text{Project} \sqsubseteq (\geq 1 \text{ Works-for}^{\neg})$   
 $\text{Project} \sqsubseteq (\leq 1 \text{ Manages}^{\neg}) \sqcap (\geq 1 \text{ Manages}^{\neg})$   
 $\text{Top-Manager} \sqsubseteq (\geq 1 \text{ Manages}) \sqcap (\leq 3 \text{ Manages})$

### ER diagrams – another Example (ctd.)



$\text{Manager} \sqsubseteq \text{Employee}$   
 $\text{Manager} \sqsubseteq \text{Area-Manager} \sqcup \text{Top-Manager}$   
 $\text{Area-Manager} \sqsubseteq \text{Manager}$   
 $\text{Top-Manager} \sqsubseteq \text{Manager}$   
 $\text{Area-Manager} \sqsubseteq \neg \text{Top-Manager}$

### ER-diagrams – another Example with Additional (Integrity) Constraints



- If we also want to express that managers do not work for a project (she/he just manages it), we can add the following to  $\pi'(S)$ :  

$$\text{Manager} \sqsubseteq \forall \text{Works-For}.\perp$$
- If, additionally, the minimum cardinality for the participation of employees to the Works-For relationship is increased, then, TopManager becomes unsatisfiable
- If an IS-A link is added stating that every AreaManager is a TopManager, then
  - AreaManager becomes unsatisfiable and
  - every Manager is a TopManager

### ER-diagrams: Reasoning Problems and Inference Services

As we have just seen,

- each ER-diagram  $S$  can be translated into a finite set of DL axioms  $\pi'(S)$  and
- additional integrity constraints can be expressed,
- which might lead to the inconsistency of a conceptual model...

**Inferences:** a TBox  $\mathcal{T}$  obtained by the translation of an ER-diagram  $S$  plus possibly some DL axioms as additional integrity constraints is called a **DL conceptual model**.

For  $\mathcal{T}$  a DL conceptual model, and  $X, Y$  entities or relationships, we say that

- $X$  is **inconsistent** in  $\mathcal{T}$  if  $DB \models \mathcal{T}$  implies  $X^{DB} = \emptyset$   
 iff  $X$  is **not satisfiable** w.r.t.  $\mathcal{T}$
- $X$  is **subsumed** by  $Y$  in  $\mathcal{T}$  if  $DB \models \mathcal{T}$  implies  $X^{DB} \subseteq Y^{DB}$   
 iff  $X$  is subsumed by  $Y$  w.r.t.  $\mathcal{T}$  in the DL sense

## ER-diagrams: Reasoning Problems and Inference Services

Clearly, to verify your conceptual model, it would be nice to **automatically**

- test each entity and relationship for (in)consistency and
- test each pair of entities and each pair of relationships for subsumption and
- notify the user about each
  - inconsistency and
  - subsumption that is not **explicit** in the ER-diagram, where a subsumption relation between  $X$  and  $Y$  is **explicit** if there is a **path** of IS-A links between  $X$  and  $Y$  in the ER-diagram
- fortunately, there exists a decision procedure for these inferences: they translate to DL (un)satisfiability and subsumption, which are both decidable for the DL *ALCIN*
- see the coursework with icom, which is a tool that implements this idea

## Summary

In this section, we have seen

- that and how ER-diagrams can be translated into FOL and
- how additional integrity constraints can be expressed,
- which might yield inconsistencies or implicit IS-A relationships.

Hence reasoning algorithms would be desirable, and thus we have

- discussed the translation of ER-diagrams and integrity constraints into the DL *ALCIN*,
- for which satisfiability and subsumption are known to be decidable.
- You are going to experience the whole approach in the coursework.

## Non-Standard Reasoning Services in Description Logics [2, supervised Lab]

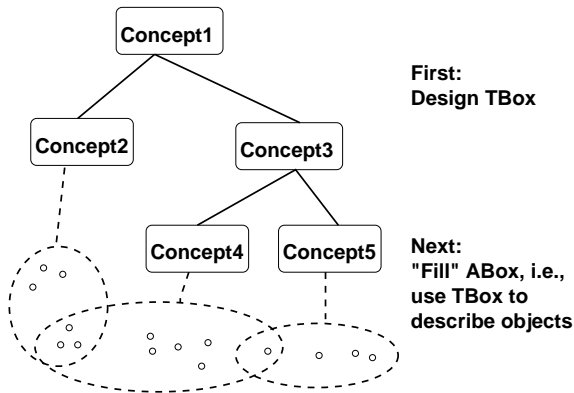
Uli Sattler

## General Remarks

In this session, we will discuss

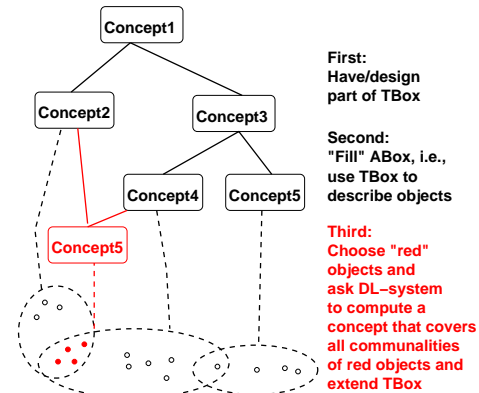
- a selection of reasoning problems that cannot be translated into standard logical reasoning problems such as satisfiability, validity, or consequence,
- but are still logical problems,
- and have nice applications in knowledge representation

Motivation: in general: Top-Down Construction of TBoxes



- Problem:
- to design a TBox, a user has to define concepts
  - user has to know KR formalism/DL and the application field

Motivation: helpful: Bottom-Up Construction of TBoxes



Observation: often, describing objects as prototypical instances of a concept is easier than defining this concept

Problem: how to compute Concept5 for the "red" objects?

Motivation: Example

Assume you have built the following TBox:

- NoSon  $\sqsubseteq \forall \text{child.Female}$ ,
- NoDaughter  $\sqsubseteq \forall \text{child.}\neg \text{Female}$ ,
- SonRichDoctor  $\sqsubseteq \forall \text{child.}(\neg \text{Female} \Rightarrow (\text{Doctor} \sqcap \text{Rich}))$ ,
- DaughterHappyDoctor  $\sqsubseteq \forall \text{child.}(\text{Female} \Rightarrow (\text{Doctor} \sqcap \text{Happy}))$
- ChildrenDoctor  $\sqsubseteq \forall \text{child.Doctor}$

and find the following ABox

- $(a_1, c_1) : \text{child}, c_1 : \text{NoSon} \sqcap \text{DaughterHappyDoctor}$
- $(a_2, c_2) : \text{child}, c_2 : \text{NoDaughter}, c_2 : \text{SonRichDoctor}$

A concept "covering"  $a_1$  and  $a_2$  would be  $\exists \text{child.ChildrenDoctor}$ .

↪ How can we compute such a concept?

Motivation

Suppose you use a DL-based KR system and have, so far, built

- a TBox  $\mathcal{T}$  with the relevant concepts of your application domain, and
- an ABox  $\mathcal{A}$  containing some objects with descriptions and relationships

Now you realize that the objects  $o_1, \dots, o_k$  should all be instances of a certain concept  $C$  that is not yet defined in  $\mathcal{T}$ ...

Since formulating the definition of the concept  $C$  might be tricky, you want some help,

i.e., a system service that automatically generates, from  $\mathcal{A}$  and  $\mathcal{T}$ , a concept  $X$

- such that each  $o_i$  is an instance of it in each model of  $\mathcal{T}$  and  $\mathcal{A}$ , i.e.,  $(\mathcal{T}, \mathcal{A}) \models o_i : X$  and
- $X$  is as specific as possible (otherwise,  $X = \top$  would be an easy answer): whenever  $(\mathcal{T}, \mathcal{A}) \models o_i : Y$ , for all  $1 \leq i \leq k$ , then  $(\mathcal{T}, \mathcal{A}) \models X \sqsubseteq Y$

### Motivation: an Example

- Example:**
- let  $\mathcal{T}$  be empty (you just start to build it) and
  - $\mathcal{A} = \{ o_1 : \exists \text{child}.\top \sqcap \forall \text{child}.\text{Male} \sqcap \text{Doctor} \}$   
 $o_2 : \exists \text{child}.\text{Male} \sqcap \text{Student} \sqcap \forall \text{child}.\text{Doctor} \sqcap \text{Female} \}$
  - what would be a “good” proposal for  $X$  covering  $o_1$  and  $o_2$ ?

- Observation:**
- in  $\mathcal{ALC}$ , we simply take the disjunction, i.e.,  
 $X = (\exists \text{child}.\top \sqcap \dots) \sqcup (\exists \text{child}.\text{Male} \sqcap \text{Student} \sqcap \dots)$ ,  
then clearly
    - $(\mathcal{T}, \mathcal{A}) \models o_1 : X$  and  $(\mathcal{T}, \mathcal{A}) \models o_2 : X$ , and
    - $X$  is the most specific such concept
  - but this is not what we wanted: a concept describing the “commonalities” of  $o_1$  and  $o_2$ !

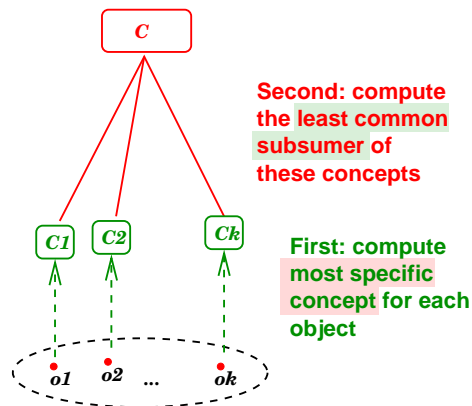
### Motivation: an Example

- Example:**
- let  $\mathcal{T}$  be empty (you just start to build it) and
  - $\mathcal{A} = \{ o_1 : \exists \text{child}.\top \sqcap \forall \text{child}.\text{Male} \sqcap \text{Doctor} \}$   
 $o_2 : \exists \text{child}.\text{Male} \sqcap \text{Student} \sqcap \forall \text{child}.\text{Doctor} \sqcap \text{Female} \}$
  - what would be a “good” proposal for  $X$  covering  $o_1$  and  $o_2$ ?

- Observation:**
- we want: a concept describing the “commonalities” of  $o_1$  and  $o_2$ !
  - better and without disjunction:  
 $X = \exists \text{child}.\text{Male} \sqcap \text{Doctor} \sqcap \forall \text{child}.\text{Doctor}$
  - how do we get this?
  - What if  $\mathcal{A}$  contains role assertions, i.e., some of  $o_i$ 's properties are **not** covered by concept assertions?

### LCS and MSC for Bottom-Up Construction of TBoxes

Usage of the computation of the least common subsumer and the most specific concept to compute a concept  $C$  describing the commonalities of the objects  $o_1, \dots, o_k$



### Important Side Remarks

- In the following, we**
- use  $\mathcal{T}$  for a TBox and
  - $\mathcal{A}$  for an ABox

Since we are concerned with the **computation** of concepts, we have to fix a description logic: besides  $\mathcal{ALC}$ , there are several other DLs, e.g., the extension of  $\mathcal{ALC}$  with inverse roles  $\mathcal{ALCI}$ .

Moreover, also **restrictions** of  $\mathcal{ALC}$  are considered in the literature:

- $\mathcal{EL}$  is the restriction of  $\mathcal{ALC}$  in which
- concepts can only be built using  $\sqcap$ ,  $\exists r.C$ , and  $\top$ .

- Example:**
- $A \sqcap \exists r.(B \sqcap C)$  is an  $\mathcal{EL}$ -concept, but
  - neither  $\forall r.B$  nor  $\neg B$  nor  $A \sqcup B$  are  $\mathcal{EL}$ -concepts

## MSC — Most Specific Concept

In the following,  $\mathcal{L}$  will stand for a description logic, e.g.,  $\mathcal{ALC}$ ,  $\mathcal{EL}$ , etc.

**MSC:** An  $\mathcal{L}$ -concept  $X$  is a **most specific  $\mathcal{L}$ -concept of  $o$  w.r.t.  $\mathcal{T}$  and  $\mathcal{A}$** , written  $X \in \text{msc}_{\mathcal{L}}(o, \mathcal{T}, \mathcal{A})$ , if

1.  $(\mathcal{T}, \mathcal{A}) \models o : X$  and
2. for all  $\mathcal{L}$ -concepts  $Y$ ,  $(\mathcal{T}, \mathcal{A}) \models o : Y$  implies  $(\mathcal{T}, \mathcal{A}) \models X \sqsubseteq Y$

**Examples:**

- for  $\mathcal{T} = \{A \doteq B \sqcap \exists r.C\}$  and  $\mathcal{A} = \{b : B, c : C, (b, c) : r\}$ , we have  $A \in \text{msc}_{\mathcal{EL}}(b, \mathcal{T}, \mathcal{A})$
- for  $\mathcal{T} = \{A \doteq B \sqcap \exists r.C, C \doteq D \sqcap \exists s.(E \sqcap F)\}$  and  $\mathcal{A} = \{b : B, c : D, (b, c) : r, (c, e) : s, e : (E \sqcap F)\}$ , we have  $B \sqcap \exists r.C \in \text{msc}_{\mathcal{EL}}(b, \mathcal{T}, \mathcal{A})$

## MSC — Most Specific Concept

**Observe:**

- $\text{msc}_{\mathcal{L}}(o, \mathcal{T}, \mathcal{A})$  may contain more than one concept, e.g.,  $B \sqcap \exists r.C \in \text{msc}_{\mathcal{EL}}(b, \mathcal{T}, \mathcal{A})$

However, by definition,  $X_1, X_2 \in \text{msc}_{\mathcal{L}}(o, \mathcal{T}, \mathcal{A})$  implies  $(\mathcal{T}, \mathcal{A}) \models X_1 \sqsubseteq X_2$  and  $(\mathcal{T}, \mathcal{A}) \models X_2 \sqsubseteq X_1$

- $\text{msc}_{\mathcal{L}}(o, \mathcal{T}, \mathcal{A})$  may contain no element, e.g.,

$\text{msc}_{\mathcal{EL}}(a, \emptyset, \{(a, a) : r, a : A\}) = \emptyset$  because  $(\emptyset, \{(a, a) : r, a : A\}) \models a : X$  for each

$X \in \{A, A \sqcap \exists r.A, A \sqcap \exists r.(A \sqcap \exists r.A), A \sqcap \exists r.(A \sqcap \exists r.(A \sqcap \exists r.A)), \dots\}$

all these concepts are increasingly specific, and there is no most specific one in  $\mathcal{EL}$

## MSC — Most Specific Concept: how can it be computed?

Clearly, an algorithm for the computation of  $\text{msc}_{\mathcal{L}}(o, \mathcal{T}, \mathcal{A})$  depends on  $\mathcal{L}$ .

As an example, we present an algorithm for the following framework:

1.  $\mathcal{L} = \mathcal{EL}$ , i.e. we only have  $\top$ , conjunctions, and existential restrictions,

2. TBoxes are sets of concepts **definitions** (i.e., no complex concepts on left hand side)

without **definitorial cycles**: a set of definitions referring cyclically to each other, i.e., of the form

$$\begin{aligned} A_0 &\doteq \dots A_1 \dots \\ A_1 &\doteq \dots A_2 \dots \\ &\dots \dots \\ A_n &\doteq \dots A_0 \dots \end{aligned}$$

3. ABoxes whose relational structure (as induced by role assertions) forms **acyclic graph**

## MSC — Most Specific Concept: how can it be computed?

The  $\text{msc}_{\mathcal{EL}}(o, \mathcal{T}, \mathcal{A})$  is then constructed as follows:

1. **construct the graph corresponding to  $\mathcal{A}$ :**

- use 1 node per object  $o_i$ , labelled with  $\mathcal{L}(o_i) = \{C \mid o_i : C \in \mathcal{A}\}$  and
- put an  $r$ -labelled edge from  $o_i$  to  $o_j$  if  $(o_i, o_j) : r \in \mathcal{A}$

2. **un-fold  $\mathcal{T}$  and break-down concepts:**

- whenever a concept symbol  $A \in \mathcal{L}(o_i)$  and  $A \doteq C$  or  $A \sqsubseteq C$  is in  $\mathcal{T}$ , then add  $C$  to  $\mathcal{L}(o_i)$
- whenever a concept  $C_1 \sqcap C_2 \in \mathcal{L}(o_i)$ , add  $C_1$  and  $C_2$  to  $\mathcal{L}(o_i)$
- whenever  $\exists r.C \in \mathcal{L}(o_i)$  and  $o_i$  doesn't have an  $r$ -successor  $o_j$  with  $C \in \mathcal{L}(o_j)$ , generate such an  $r$ -successor with label  $\{C\}$

apply these rules exhaustively—this terminates since  $\mathcal{T}$  is acyclic

3. **read the concept off...**

### MSC — Most Specific Concept: how can it be computed?

#### 3. read the concept off:

- for each path  $p = o \xrightarrow{r_1} o_1 \xrightarrow{r_2} o_2 \dots \xrightarrow{r_\ell} o_\ell$  from  $o$  to a leaf  $o_\ell$ , construct the concept

$$X_p = \exists r_1. (\hat{\mathcal{L}}(o_1) \sqcap \exists r_2. (\hat{\mathcal{L}}(o_2) \sqcap \dots \exists r_\ell. \hat{\mathcal{L}}(o_\ell)) \dots)$$

where  $\hat{\mathcal{L}}(o_i) = \bigsqcap_{C \in \mathcal{L}(o_i)} C$  (where the empty conjunction is  $\top$ )

- if  $p_1, \dots, p_r$  are all paths in the graph starting at  $o$ , return the concept

$$X = \hat{\mathcal{L}}(o) \sqcap X_{p_1} \sqcap \dots \sqcap X_{p_r}$$

- let's see an example on the blackboard for

$$\mathcal{T} = \{B \sqsubseteq \exists t.D, F \sqsubseteq A \sqcap B\}$$

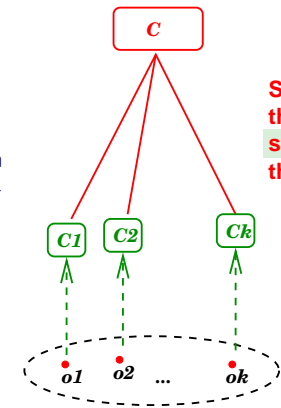
$$\mathcal{A} = \{a : A \sqcap B, b : \exists s.A \sqcap F, c : D \sqcap E,$$

$$(a, b) : r, (a, c) : t, (c, d) : u, (b, d) : s\}$$

### LCS — Least Common Subsumer

Remember: we planned to use both msc and lcs for the bottom-up construction of knowledge bases!

So far, we have discussed the msc, let's have a look at the lcs



Second: compute the least common subsumer of these concepts

First: compute most specific concept for each object

### LCS for the Bottom-Up Construction of TBoxes

Similar to the msc, the lcs (of a set of concepts) is a “common subsumer” and the smallest such subsumer. Hence its definition consists, again, of 2 conditions:

**LCS:** let  $C_1, \dots, C_k$  be concepts.

An  $\mathcal{L}$ -concept  $X$  is a **least common  $\mathcal{L}$ -subsumer** of  $C_1, \dots, C_k$  w.r.t.  $\mathcal{T}$ , written  $X \in \text{lcs}_{\mathcal{L}}(C_1, \dots, C_k, \mathcal{T})$ , if

1.  $\mathcal{T} \models C_i \sqsubseteq X$ , for all  $1 \leq i \leq k$  and
2. for all  $\mathcal{L}$ -concepts  $Y$ , if  $\mathcal{T} \models C_i \sqsubseteq Y$  for all  $1 \leq i \leq k$ , then  $\mathcal{T} \models X \sqsubseteq Y$

**Example:** for  $C_1 = \exists \text{child}.\top \sqcap \forall \text{child}.\text{(Male} \sqcap \text{Doctor)}$

$$C_2 = \exists \text{child}.\text{(Male} \sqcap \text{Student)} \sqcap \forall \text{child}.\text{(Doctor} \sqcap \text{Female)}$$

$$C_1 \sqcup C_2,$$

$$\exists \text{child}.\text{(Male} \sqcap \text{Doctor)} \sqcap \forall \text{child}.\text{Doctor} \in \text{lcs}_{\mathcal{ALC}}(C_1, C_2, \emptyset)$$

### LCS for the Bottom-Up Construction of TBoxes

- Observations:**
- again, the lcs makes only sense in the absence of disjunction since  $C_1 \sqcup \dots \sqcup C_k \in \text{lcs}_{\mathcal{L}}(C_1, \dots, C_k, \mathcal{T})$  if  $\mathcal{L}$  provides  $\sqcup$
  - again,  $\text{lcs}_{\mathcal{L}}(C_1, \dots, C_k, \mathcal{T})$  may contain several concepts, but all are equivalent
  - in contrast to the msc, for  $\text{lcs}_{\mathcal{L}}(C_1, \dots, C_k, \mathcal{T})$  to be empty is less likely: for all DLs  $\mathcal{L}$  considered here, it is never empty!

How to compute the lcs? Again, we restrict our attention to

- the DL  $\mathcal{EL}$  for the lcs concept to be computed and
- TBoxes that
  - contain only concept definitions and
  - that do **not** contain definitorial cycles
- **but** we allow the TBox to involve concepts of the form  $\top$ ,  $C_1 \sqcap C_2$ ,  $\exists R.C$ , and  $\forall R.C$  (for the msc, we did not allow for  $\forall R.C$ )

### Algorithm for the construction of $\mathcal{EL}$ -LCS

The  $\text{lcs}_{\mathcal{EL}}(C_1, C_2, \mathcal{T})$  is then constructed as follows:

1. for each  $i \in \{1, 2\}$ , **construct the tree corresponding to  $C_i$** :
  - start with root node  $o_0^i$  labelled  $\mathcal{L}(o_0^i) = \{C_i\}$
  - whenever a concept symbol  $A \in \mathcal{L}(o_j)$  and  $A \doteq C$  or  $A \sqsubseteq C$  is in  $\mathcal{T}$ , then add  $C$  to  $\mathcal{L}(o_j)$
  - whenever a concept  $D_1 \sqcap D_2 \in \mathcal{L}(o_j)$ , add  $D_1$  and  $D_2$  to  $\mathcal{L}(o_j)$
  - whenever  $\exists r.C \in \mathcal{L}(o_j)$  and  $o_j$  doesn't have an  $r$ -successor  $o_\ell$  with  $C \in \mathcal{L}(o_\ell)$ , generate an  $r$ -successor  $o_\ell$  with  $\mathcal{L}(o_\ell) = \{C\}$
  - whenever  $\forall r.C \in \mathcal{L}(o_j)$  and  $o_j$  has an  $r$ -successor  $o_\ell$  with  $C \notin \mathcal{L}(o_\ell)$ , then add  $C$  to  $\mathcal{L}(o_\ell)$
  - this terminates since  $\mathcal{T}$  is acyclic
2. **read the concept off**:...

### Algorithm for the construction of $\mathcal{EL}$ -LCS

2. **read the concept off** is defined recursively: for  $o, u$  nodes in trees, we define

$$\text{co}(o, u) = \bigcap_{\substack{A \text{ atomic concept in} \\ \mathcal{L}(o) \cap \mathcal{L}(u)}} A \sqcap \bigcap_{\substack{\text{roles } r, \\ r\text{-successors } o' \text{ of } o, \\ r\text{-successors } u' \text{ of } u}} \exists r.\text{co}(o', u')$$

(where the "empty" conjunction is  $\top$ )

Return  $\text{co}(o_0^1, o_0^2)$

- for  $\text{lcs}_{\mathcal{EL}}(C_1, \dots, C_k, \mathcal{T})$ , compute
 
$$\text{lcs}_{\mathcal{EL}}(C_1, \dots, \text{lcs}_{\mathcal{EL}}(C_{k-2}, \text{lcs}_{\mathcal{EL}}(C_{k-1}, C_k, \mathcal{T}), \mathcal{T}) \dots \mathcal{T})$$
- the "read the concept off" construction is a product on trees!
- let's see an example on the blackboard for

$$C_1 = A \sqcap B \sqcap \exists s.C \sqcap \forall s.(D \sqcap E \sqcap \exists t.(F \sqcap G))$$

$$C_2 = A \sqcap \exists s.D \sqcap \forall s.C \sqcap \exists s.(E \sqcap \exists t.F)$$

$$C_3 = B \sqcap \exists s.(C \sqcap \exists t.F) \sqcap \exists s.D$$

### Algorithm for the construction of $\mathcal{EL}$ -LCS

What does this algorithm do? What are its properties?

- When started with  $C_1, C_2$ , and  $\mathcal{T}$  (all conforming to the restrictions mentioned before),
- it always terminates
  - because maximal length of concepts in node labels decreases strictly
  - because  $\mathcal{T}$  contains no definitorial cycles
- it computes some  $X \in \text{lcs}_{\mathcal{EL}}(C_1, C_2, \mathcal{T})$
- whose size can be **exponential** in the size of  $C_1$  and  $C_2$

### Other Non-Standard Inferences: Rewriting

**Rewriting:** of a concept  $C$  w.r.t. a TBox  $\mathcal{T}$ :

- given  $C$  and  $\mathcal{T}$ , compute some  $D$  such that
  - $\mathcal{T} \models C \sqsubseteq D$  and  $\mathcal{T} \models D \sqsubseteq C$   
( $C$  and  $D$  are equivalent w.r.t.  $\mathcal{T}$ ) and
  - $D$  is a concept of minimal length with this property

For example,  $E \sqcap \exists r.(F \sqcap B)$  is a minimal rewriting of

$$\exists r.(B \sqcap (\exists t.\exists s.B)) \sqcap \exists s.\exists t.A$$

w.r.t. the TBox  $\{ E \doteq \exists s.\exists t.A \}$

$$F \doteq \exists t.\exists s.B$$

Extremely useful, e.g., since the  $\text{lcs}$  algorithm can compute quite large concepts!

Such an answer concept has to be analysed by the user to see whether it suits her intuition...so rewriting it to equivalent shorter ones helps this analysis!

## Other Non-Standard Inferences: Approximation

**Approximation:** of a concept  $C$  in a DL  $\mathcal{L}$  into a DL  $\mathcal{L}'$  w.r.t. a TBox  $\mathcal{T}$  where  $\mathcal{L}'$  is **less expressive** than  $\mathcal{L}$ :

- given  $\mathcal{T}$  an  $\mathcal{L}$ -concept  $C$ , compute an  $\mathcal{L}'$ -concept  $D$  such that
  - $\mathcal{T} \models C \sqsubseteq D$  and
  - $D$  is a **most specific**  $\mathcal{L}'$ -concept with this property

For example, the  $\mathcal{ALCC}$ -concept

$$\exists r.(A \sqcap B)$$

is an approximation of the  $\mathcal{ALCCN}$ -concept

$$\exists r.A \sqcap \exists r.B \sqcap (\leq 1 r)$$

Which information is lost in the approximation?

Approximations are useful for users that are no DL experts and only understand “fragments” of the DL used in a certain application, i.e., they provide a **simplified view** of the TBox.

## Non-Standard Inferences: Summary and Outlook

- We have seen several useful inference services that cannot be reduced to classical logical ones such as satisfiability or validity:
  - most specific concept **msc** (in depth)
  - least common subsumer **lcs** (in depth)
  - rewriting of concepts (sketchy)
  - approximation of concepts (sketchy)
- other such inference services have already been investigated
  - matching of concepts
  - unification of concepts
- more such inferences will be needed in the future
  - describe the difference between two concepts
  - tell me all about an object (is this **msc**?)
  - what is the “closest” concept defined in  $\mathcal{T}$  to  $C$ ?

## Temporal Logic [3, supervised Labs]

Uli Sattler

## General Remarks

In this section of the lecture, we will

- discuss different possibilities to represent **temporal knowledge**, i.e., knowledge somehow related to **time**,
- start from a rather simple, well-known formalism, propositional linear temporal logic LTL, and then
- extend it to **structured domains**, i.e., replace the “propositional” with “description logical”



## First Order Logic for Temporal Statements

2. in another approach, **event-token reification** (Davidson, 1967), each
- event-forming predicate (e.g.,  $See(John, Mary)$ ) is extended with
  - an extra argument-place for a variable ranging over time-points, e.g.,  $See(John, Mary, s)$

This allows to draw conclusion such as

$\exists e(See(John, Mary, e) \wedge Place(e, London) \wedge Time(e, Tuesday))$   
implies  $\exists e(See(John, Mary, e) \wedge Time(e, Tuesday))$ .

Again, we have to axiomatise a predicate  $P_{<}$  to compare time-points

## First Order Logic for Temporal Statements

3. the **situation calculus** (McCarthy, Reiter) is a FOL approach to reason about actions, their consequences, and ultimately **plan** agent/robot behaviour. It involves

- actions  $a$  (as terms),
- situations  $s$  (as terms),
- a function  $do(a, s)$  mapping  $a$  and  $s$  to the situation resulting of carrying out action  $a$  in situation  $s$ , and
- other predicates, e.g.,  $On(book, table, s)$ ,  $poss(a, s)$

Effects of actions are expressed as follows:

$$\forall x, y, z, s. (On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \wedge x \neq z) \Rightarrow On(x, z, \underbrace{do(move(x, y, z), s)}_{s'})$$

Actions typically leave many aspects of a snapshot unchanged...describing, for each action,

- what they change seems feasible
- what they don't change seems tedious

## First Order Logic for Temporal Statements

This is known as **frame problem** (how to state what remains unchanged).  
In the situation calculus, a variety of **frame axioms** express what remains unchanged, e.g.,

$$\forall x, y, z, v, u, s. (On(x, y, s) \wedge x \neq u) \Rightarrow On(x, y, do(move(u, v, z), s))$$

4. and many other approaches were developed for different applications

In all three approaches sketched above,

- situations/time-points/intervals are “logical objects” and
- a formalism without a notion of time is used for representing temporal knowledge, which implies that
- extra axioms have to be added to axiomatise the desired temporal properties, e.g., of  $P_{<}$  and
- readability might become poor

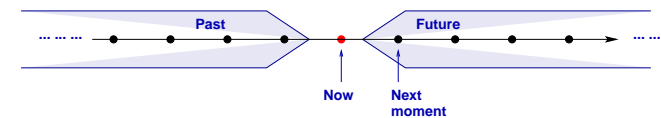
## A Logic for Temporal Knowledge

Next, we discuss an example of a (modal) logical formalism designed for the representation of and reasoning about temporal knowledge.

We will concentrate on

- qualitative
  - point-based
    - discrete
    - linear time,

that is, a time flow that looks as follows:



However, there are similar formalisms for other temporal structures.

## LTL, a Logic for Temporal Knowledge: Syntax

**LTL:** is a modal logic, and its formulae are inductively defined as follows:

given a set  $P$  of propositional variables, the set of LTL formulae is the smallest set such that

- each  $p \in P$  is a LTL formulae
- if  $\phi$  and  $\psi$  are LTL formulae, then

$$\phi \wedge \psi, \quad \phi \vee \psi, \quad \neg\psi, \quad \circ\psi, \quad \text{and } \psi\mathcal{U}\phi$$

Intuitively, we read

- $\circ\psi$  as “in the **next** time-point (moment, day, etc)  $\psi$  holds” and
- $\psi\mathcal{U}\phi$  as “ $\psi$  holds **until**  $\phi$  holds”

**Example:**

- $bf \wedge obt$  can be read as *I have breakfast and next I brush my teeth*
- $bf \wedge wk\mathcal{U}l$  can be read as *I have breakfast and then work until lunch*

## LTL, a Logic for Temporal Knowledge: Semantics

So far for the intuitive reading, let's define the semantics!

As mentioned before, LTL is a **modal logic**, hence we can define its semantics using Kripke structures:

**LTL Semantics** is given by a Kripke Structure  $\mathcal{M} = (\mathbb{N}, <, I)$  where

- $\mathbb{N}$  are the non-negative integers, representing time-points,
- $<$  is the natural ordering on  $\mathbb{N}$ , representing “before”, and
- $I$  maps each propositional variable  $p \in P$  to the set of **time-points**  $I(p)$  in which  $p$  holds.

The interpretation of formulae is (as usually) defined inductively:

$$\begin{aligned} \mathcal{M}, w \models p & \quad \text{iff } w \in I(p) \\ \mathcal{M}, w \models \psi \wedge \phi & \quad \text{iff } \mathcal{M}, w \models \psi \text{ and } \mathcal{M}, w \models \phi \\ \mathcal{M}, w \models \psi \vee \phi & \quad \text{iff } \mathcal{M}, w \models \psi \text{ or } \mathcal{M}, w \models \phi \\ & \quad \text{so far, nothing new} \end{aligned}$$

## LTL, a Logic for Temporal Knowledge: Semantics

So far for the intuitive reading, let's define the semantics!

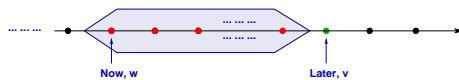
As mentioned before, LTL is a **modal logic**, hence its semantics is based on Kripke structures:

**LTL Semantics** is given by a Kripke Structure  $\mathcal{M} = (\mathcal{N}, <, I)$  where

...

(ctd.) The interpretation of formulae is (as usually) defined inductively:

$$\begin{aligned} \mathcal{M}, w \models \circ\psi & \quad \text{iff } \mathcal{M}, (w + 1) \models \psi \\ \mathcal{M}, w \models \phi\mathcal{U}\psi & \quad \text{iff there is some } v \geq w \text{ such that} \\ & \quad \mathcal{M}, v' \models \phi \text{ for all } v' \text{ with } w \leq v' < v \text{ and} \\ & \quad \mathcal{M}, v \models \psi \end{aligned}$$



## LTL, a Logic for Temporal Knowledge: Abbreviations

Again, we have only specified a few logical constructors, but we can use more as abbreviations:

- $\Rightarrow, \Leftrightarrow, \perp, \top$ , etc. are the standard propositional abbreviations
- we have “next” and “until”, but what about “eventually in the future”?

We can introduce it as abbreviation: in the following, we use  $\diamond\phi$  for “eventually in the future  $\phi$ ”:

$$\diamond\phi = \top\mathcal{U}\phi$$

Let's see why this is ok on the blackboard!

- similarly, we don't have “always in the future”?

Since “always in the future  $\phi$ ” is equivalent to “not eventually in the future  $\neg\phi$ ”, we introduce  $\square\phi$  for “always in the future  $\phi$ ” as follows:

$$\square\phi = \neg\diamond\neg\phi = \neg(\top\mathcal{U}\neg\phi)$$

Let's see why this is ok on the blackboard!

## LTL, a Logic for Temporal Knowledge: Example

Let's model the behaviour of traffic lights  $i$  at one crossing:

- use propositional variables  $g_i$  ( $r_i$ ,  $y_i$ ) for “the traffic light  $i$  shows green (red, yellow)”
- state the behaviour of each single traffic light:

$$\Box((g_i \Rightarrow \neg(r_i \vee y_i)) \wedge (r_i \Rightarrow \neg(g_i \vee y_i)) \wedge (y_i \Rightarrow \neg(g_i \vee r_i))) \wedge$$

$$\Box((g_i \Rightarrow g_i \mathcal{U}(y_i \wedge y_i \mathcal{U} r_i)) \wedge (r_i \Rightarrow r_i \mathcal{U}(y_i \wedge y_i \mathcal{U} g_i)))$$

- Is our system (of  $n$  traffic lights) **lively**? I.e., does the above specification imply

$$\bigwedge_{1 \leq i \leq n} \Box(r_i \Rightarrow \Diamond g_i)$$

- Is our system (of  $n$  traffic lights) **safe**? I.e., does the above specification imply

$$\bigwedge_{1 \leq i < j \leq n} \Box(\neg(g_i \wedge g_j))$$

## LTL, a Logic for Temporal Knowledge: Reasoning Problems I

As we have just seen, we can

- use LTL to **specify** (the behaviour of) a **system**, and then
- translate (un)desired properties of our specification into **implications**
- We know that we can reduce an implication  $\psi \Rightarrow \phi$  (or consequence) to the unsatisfiability of  $\psi \wedge \neg\phi$ , thus
- is (un)satisfiability of LTL decidable? If yes, how complex is it?
  - since LTL is an extension of propositional logic (PL), satisfiability of LTL is at least as hard as satisfiability of PL, i.e., NP-hard
  - it is decidable, and we can design an algorithm that runs in
    - \* **exponential time** (is that much?) and
    - \* **polynomial space**
  - we can show that sat. of LTL is PSpace-complete, i.e., in the next class above NP

## LTL, a Logic for Temporal Knowledge: Reasoning Problems II

Alternatively to

- **specifying** a system (and thus considering a variety of its implementations, i.e., a variety of Kripke structures),
- we can **fix** a system, i.e., consider a **single Kripke structure**  $\mathcal{M}$ , and ask whether

$\mathcal{M}$  satisfies a (desired or undesired) property  $\psi$

- this translates into

given  $\mathcal{M}$  with some “initial state”  $s$  and  $\psi$ , does  $\mathcal{M}, s \models \psi$ ?

- this problem is known as **model checking**
- it is a special case of satisfiability: simply use a specification that is so “strict” that it only has a single model  $\mathcal{M}$
- is LTL model checking decidable? If yes, how complex is it?
  - we can show that LTL model checking is also PSpace-complete, i.e., as complex as satisfiability

## Logics for Temporal Knowledge: Extensions of LTL I

There are several shortcomings of LTL that motivated the investigation of extensions

1. in LTL, we can only talk about the **future**:

to talk about the **past**, we can extend LTL with converse modalities

- $\Box\psi$  for “in the previous moment/day,  $\psi$  was true”
- $\phi\overline{\mathcal{U}}\psi$  for “ $\phi$  did hold since  $\psi$  was true” (where  $\overline{\mathcal{U}}$  is to be read as “since”)
- as before, we can introduce an abbreviation

$\overline{\Diamond}\psi = \top\overline{\mathcal{U}}\psi$  for “somewhere in the past,  $\psi$  was true” and

$\Box\psi = \neg\overline{\Diamond}\neg\psi = \neg(\top\overline{\mathcal{U}}\neg\psi)$  for “always in the past,  $\psi$  was true”

in addition, we have to decide whether our temporal structure

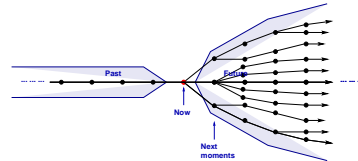
- starts (e.g., with a big bang), i.e., is isomorphic to  $\mathbb{N}$ , or
- has no “start”, i.e., is isomorphic to  $\mathbb{Z}$

## Logics for Temporal Knowledge: Extensions of LTL I

2. in LTL, time is **linear**. However, if we describe

- (the behaviour of) systems, we might want to consider

- several futures/successor states, i.e., our temporal structure looks like a **tree**:



- reflecting non-determinism or interaction with the outside world
- for example, consider two processes with some mutual exclusion part and states
  - non-critical  $n_i$ ,
  - trying to go into the critical part  $t_i$ , and
  - being in the critical part  $c_i$ , where we use semaphore variables  $s_i$  we can only switch into  $c_i$  if  $s_i$  is false.

Since, at each point in time, different actions are possible, modelling this example in “branching” logic is useful

## Logics for Temporal Knowledge: Extensions II

Another shortcoming of LTL is due to the fact that it extends **propositional logic**: each state/moment is a point in which certain propositional variables hold or not hold.

What if our world has a **richer structure**, i.e., if we want to talk about

- objects (Peter, Paul, and Mary),
- predicates (Happy, Human), and
- relations between object (likes, marriedTo, neighbourOf)?

Solutions:

1. if we only have few objects and relations, “press” it all in propositional setting, e.g., use propositional variables  $\text{HappyPeter}$ ,  $\text{PeterLikesMary}$ , etc.
2. use **temporal FOL** – but FOL is already undecidable, hence any extension will be!
3. use an appropriate **decidable fragment** of FOL and temporalise it...

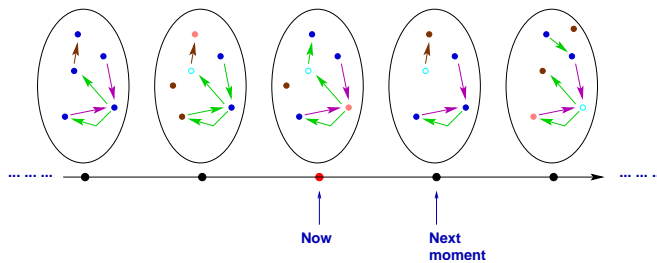
## Temporal Description Logics

We know extensions of propositional logic that are decidable fragments of FOL:

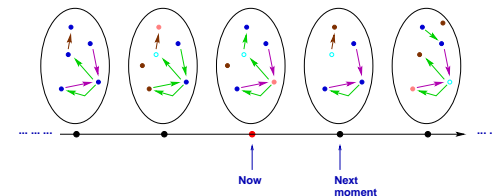
### Description Logics

So let's see how they can be temporalised:

**General Idea:** at each time point, consider a whole DL interpretation



## Temporal Description Logics



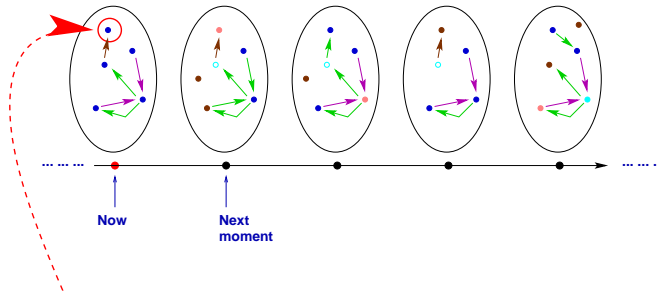
- next, temporalise concepts:

$\text{Stud} \sqcap \diamond \text{Rich} \sqcap \square \text{Happy} \sqcap \text{Stud} \mathcal{U} \text{GoodJob}$

describes students who are eventually rich, are always in the future happy, and who will be students until they find a good job

an object does not only “live” in one interpretation, but in many different interpretations at various time points

## Temporal Description Logics



this objects is an instance of

$\text{Blue} \sqcap \Diamond \text{Brown} \sqcap \Box (\neg \exists \text{greenRel.Brown}) \sqcap \Diamond (\exists \text{greenRel.Blue})$

where the  $\Box (\neg \exists \text{greenRel.Brown})$  is “difficult” because we only see part of time scale!

## Temporal Description Logics

• next, we stronger temporalise concepts:

- $\text{Human} \sqcap \Diamond (\exists \text{marriedTo.} (\text{Bold} \sqcap \text{Human}))$  describes humans who will eventually be married to a bold human, whereas
- $\text{Human} \sqcap \exists \text{marriedTo.} (\Diamond \text{Bold} \sqcap \text{Human})$  describes humans who are married to a human who will eventually be bold!

• next, extend TBoxes to allow for temporalised concepts and temporalise axioms:

- $\Box (\text{Living} \dot{\sqsubseteq} \text{Living} \mathcal{U} (\text{Dies} \sqcap \Box \neg \text{Living}))$  expresses that it will always be the case that any living object will remain alive until it dies, and then remain dead
- $\Diamond (\text{Living} \dot{\sqsubseteq} \text{Happy})$  expresses that there will eventually be a “world” where all living objects are happy

• so far for the intuition, let’s see syntax and semantics of this logic!

## Temporal Description Logics: Syntax of $\text{LTL}_{\mathcal{ALC}}$

**$\text{LTL}_{\mathcal{ALC}}$ -concepts:** for  $N_K$  and  $N_R$  sets of atomic concepts and roles, the set of  **$\text{LTL}_{\mathcal{ALC}}$ -concepts** is the smallest set such that

- every  $A \in N_K$  is a  $\text{LTL}_{\mathcal{ALC}}$ -concept and,
- if  $C$  and  $D$  are  $\text{LTL}_{\mathcal{ALC}}$ -concepts and  $r$  is a role, then also  $\neg C, C \sqcap D, C \sqcup D, \exists r.C, \forall r.C, \circ C, \text{CUD}$  are  $\text{LTL}_{\mathcal{ALC}}$ -concepts.

So far, we have simply extended  $\mathcal{ALC}$  with the temporal operators on concepts – what about axioms?

For axioms, we will allow TBox and ABox axioms, their temporalisation, and Boolean combinations thereof...

## Temporal Description Logics: Syntax of $\text{LTL}_{\mathcal{ALC}}$

**$\text{LTL}_{\mathcal{ALC}}$ -axioms:** the set of  $\text{LTL}_{\mathcal{ALC}}$ -axioms is the smallest set such that

- if  $C, D$  are  $\text{LTL}_{\mathcal{ALC}}$ -concepts,  $a, b$  are object names, and  $r$  is a role, then

$$a : C, (a, b) : r, C \dot{\sqsubseteq} D, C \dot{\supseteq} D$$

are  $\text{LTL}_{\mathcal{ALC}}$ -axioms and,

- if  $\psi, \phi$  are  $\text{LTL}_{\mathcal{ALC}}$ -axioms, then

$$\neg \phi, \phi \wedge \psi, \phi \vee \psi, \circ \psi, \phi \mathcal{U} \psi$$

are also  $\text{LTL}_{\mathcal{ALC}}$ -axioms

- again, we can use all the usual abbreviations  $\Rightarrow, \dots$  and  $\Diamond \psi, \Box \psi$
- perhaps we should go back to the motivating examples and check whether they fit the syntax definition?
- what about the semantics?

## Temporal Description Logics: Semantics of $LTL_{\mathcal{ALC}}$

As shown in the pictures before, semantics is given by

- a sequence of  $\mathcal{ALC}$  interpretations  $\mathcal{I}_i$  or
- the two-dimensional combination of LTL and  $\mathcal{ALC}$  interpretations
- To define this, we must first make some assumptions:
  - can objects **appear**? can objects **disappear**?

↪ here, objects can neither **disappear** nor **appear**,  
i.e., all interpretations  $\mathcal{I}_i$  share the same interpretation domain  $\Delta^{\mathcal{I}}$

**$LTL_{\mathcal{ALC}}$ -semantics:** is given by an infinite sequence  $\mathcal{M} = \mathcal{I}_i, i \geq 0$  of  $\mathcal{ALC}$ -interpretations  $\mathcal{I}_i = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}_i})$  sharing the same domain  $\Delta^{\mathcal{I}}$ .

As usual, each  $\cdot^{\mathcal{I}_i}$  associates

- a set  $A^{\mathcal{I}_i} \subseteq \Delta^{\mathcal{I}}$  with each atomic concept  $A$  and
- a binary relation  $r^{\mathcal{I}_i} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$  with each role name  $r$
- an element  $a^{\mathcal{I}_i} \in \Delta^{\mathcal{I}}$  with each object name

## Temporal Description Logics: Semantics of $LTL_{\mathcal{ALC}}$

**$LTL_{\mathcal{ALC}}$ -concepts** are interpreted as follows:

$$\begin{aligned} \top^{\mathcal{I}_i} &= \Delta^{\mathcal{I}}, & (\neg C)^{\mathcal{I}_i} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}_i} \\ (C \sqcap D)^{\mathcal{I}_i} &= C^{\mathcal{I}_i} \cap D^{\mathcal{I}_i}, & (C \sqcup D)^{\mathcal{I}_i} &= C^{\mathcal{I}_i} \cup D^{\mathcal{I}_i} \\ (\exists r.C)^{\mathcal{I}_i} &= \{d \mid \text{there is } e \text{ s.t. } (d, e) \in r^{\mathcal{I}_i} \text{ and } e \in C^{\mathcal{I}_i}\} \\ (\forall r.C)^{\mathcal{I}_i} &= \{d \mid \text{for all } e, \text{ if } (d, e) \in r^{\mathcal{I}_i}, \text{ then } e \in C^{\mathcal{I}_i}\} \end{aligned}$$

nothing unusual so far – but that  $\mathcal{I}$  carries an index  $i$ !

$$\begin{aligned} (\circ C)^{\mathcal{I}_i} &= C^{\mathcal{I}_{i+1}} \\ (CUD)^{\mathcal{I}_i} &= \{d \mid \text{there is } j \geq i \text{ s.t. } d \in D^{\mathcal{I}_j} \text{ and} \\ &\quad \text{for all } \ell \text{ with } i \leq \ell < j : d \in C^{\mathcal{I}_\ell}\} \end{aligned}$$

Before going to the semantics of axioms, let's consider instances of concepts

Human  $\sqcap$  Happy  $\sqcap$   $\circ$ Bold  
 Human  $\sqcap$  ( $\neg$ Happy) $\mathcal{U}$ ( $\exists$ marriedTo.Bold)  
 Human  $\sqcap$  ( $\neg$ Happy) $\mathcal{U}$ ( $\exists$ marriedTo. $\diamond$ Bold)  
 Human  $\sqcap$   $\exists$ marriedTo.( $\diamond$ Bold)

## Temporal Description Logics: Semantics of $LTL_{\mathcal{ALC}}$

**$LTL_{\mathcal{ALC}}$ -axioms** are interpreted as follows: for  $n \in \mathcal{N}$ , we define

$$\begin{aligned} \mathcal{M}, n \models \psi \wedge \phi & \text{ iff } \mathcal{M}, n \models \psi \text{ and } \mathcal{M}, n \models \phi \\ \mathcal{M}, n \models \psi \vee \phi & \text{ iff } \mathcal{M}, n \models \psi \text{ or } \mathcal{M}, n \models \phi \\ \mathcal{M}, n \models \neg \phi & \text{ iff not } \mathcal{M}, n \models \phi \quad \text{nothing unusual so far!} \\ \mathcal{M}, n \models a : C & \text{ iff } a^{\mathcal{I}_n} \in C^{\mathcal{I}_n} \\ \mathcal{M}, n \models (a, b) : r & \text{ iff } (a^{\mathcal{I}_n}, b^{\mathcal{I}_n}) \in r^{\mathcal{I}_n} \\ \mathcal{M}, n \models C \sqsubseteq D & \text{ iff } C^{\mathcal{I}_n} \subseteq D^{\mathcal{I}_n} \\ \mathcal{M}, n \models C \doteq D & \text{ iff } C^{\mathcal{I}_n} = D^{\mathcal{I}_n} \quad \text{still nothing unusual!} \\ \mathcal{M}, n \models \circ \psi & \text{ iff } \mathcal{M}, n+1 \models \psi \\ \mathcal{M}, n \models \phi \mathcal{U} \psi & \text{ iff there is some } m \geq n \text{ s.t. } \mathcal{M}, m \models \psi \text{ and} \\ & \text{for all } i \text{ with } n \leq i < m : \mathcal{M}, i \models \phi \\ & \text{only combinations of old, known stuff!} \end{aligned}$$

## Temporal Description Logics: Outlook

We have just defined the very powerful combination of LTL and  $\mathcal{ALC}$  because

- we wanted to represent **structured** worlds at each time-point (i.e., LTL did not suffice) and
- we know that satisfiability of FOL formulae is undecidable, (i.e., its temporalisation is as well undecidable)
- so what about the satisfiability of  $LTL_{\mathcal{ALC}}$ -axioms? Is it decidable?
  - yes, it is decidable, but of a rather high complexity – far beyond that of  $\mathcal{ALC}$

## Temporal Description Logics: Outlook

We have just defined the very powerful combination of LTL and  $\mathcal{ALC}$  thus allowing the temporal statements over structured domains.

Clearly, according to the application we are interested in, we can

- replace the underlying DL with something weaker (e.g.,  $\mathcal{EL}$ ) or stronger (e.g.,  $\mathcal{ALCCIN}$ )
- replace the underlying temporal logic LTL with another one, e.g., one for branching time or one with temporal operators for the past

A nice application of this framework is to reason about **temporal ER-diagrams**:

- extend ER-diagrams with the notion of time, and
- translate it into the temporalisation of  $\mathcal{ALCCIN}$
- just as we have done it for ER-diagrams with additional integrity constraints

## Defaults [3, supervised Labs]

Uli Sattler

## Motivation

In the first section, we have seen different readings of is-a relations:

- if  $A$  and  $B$  are concepts/classes, does is-a translate to
  - each instance of  $A$  is also an instance of  $B$ ? E.g., each square is a rectangle
  - by default/normally  $A$ s are  $B$ s? E.g. normally birds are flying animals?
  - $A$ s inherit all properties of  $B$ —if not stated otherwise? E.g., white elephants are elephants, but they are not grey, but white
  - ...?

So far, all formalisms presented employ the first reading – what about the second one?

- suppose you are describing animal families:
  - birds fly – but for penguins, those with broken wings, those in an oil disaster, etc
  - dogs have a tail – but those whose tail is docked
  - mammals have their heart on the right side – but those that have it on the left
  - etc.

## Motivation

- We can introduce concepts “NormalBird”, “NormalMammal”, etc., and model abnormalities explicitly, e.g.,

$$\begin{aligned} \text{NormalBird} &\stackrel{\dot{=}}{=} \text{Bird} \sqcap \text{Normal} \\ \text{NormalBird} &\stackrel{\dot{=}}{=} \exists \text{ableTo.Fly} \end{aligned}$$

**Problem:** if we learn that  $a : \text{Bird}$ ,

- we cannot conclude/assume/believe that  $a$  is a **normal** bird, i.e.,
- normality cannot be **assumed by default**, but
- has to be **stated explicitly**.
- Hence we cannot assume that  $a$  is a normal bird (and can thus fly) unless we learn about some abnormality

- Or we can extend our knowledge representation formalism with **defaults** i.e., expressive means to make statements such as

*if I know that  $a$  is a bird, and it is safe to assume that  $a$  is normal, then I conclude that  $a$  can fly*

## Propositional Default Logic: Syntax

**Default:** for  $\alpha, \beta, \gamma$  propositional formulae, we call  $\frac{\alpha : \beta}{\gamma}$  a **default** with

- **pre-requisite**  $\alpha$ ,
- **justification**  $\beta$ , and
- **consequent**  $\gamma$ .

A **default theory**  $(\mathcal{W}, \mathcal{D})$  consists of

- a finite set  $\mathcal{W}$  of propositional formulae (the background knowledge) and
- a finite set  $\mathcal{D}$  of defaults

**Reading:** a default  $\frac{\alpha : \beta}{\gamma}$  is read as *if I know  $\alpha$ , and it is safe to assume  $\beta$ , then I can conclude  $\gamma$*

**Example:** the (prop. version of the) bird example can be formalised as  $\frac{\text{Bird} : \text{Normal}}{\text{Fly}}$

**Question:** what does “it is safe to assume  $\beta$ ”/“I can conclude  $\gamma$ ” mean? Semantics?

## Propositional Default Logic: Semantics – Preliminaries

The exact meaning of defaults is given via so-called **extensions**, where an extension is

- a **set of formulae** that is
- **deductively closed**, i.e., if  $E$  is deductively closed and  $E \models \psi$ , then  $\psi \in E$

**Theory:** for  $\Gamma$  a set of formulae, its **theory**  $Th(\Gamma)$  the smallest set that

- **contains**  $\Gamma$  and that
- is **deductively closed**

**Remarks:** • given  $\Gamma$ , we can “construct”  $Th(\Gamma)$  by exhaustively adding all consequences of  $\Gamma$

• e.g.,  $Th(\{p \wedge q, p \Rightarrow (s \wedge (t \vee u))\}) = \{p \wedge q, p \Rightarrow (s \wedge (t \vee u)), p, q, (s \wedge (t \vee u)), s, (t \vee u), \dots\}$

• since we are in propositional logic, we can decide  $E \models \psi$ ?

• since there are only finitely many propositional formulae over a finite signature (up to equivalence), we can thus effectively **compute**  $Th(\Gamma)$  if  $\Gamma$  is finite

## Propositional Default Logic: Semantics

**Extensions:** let  $\Gamma$  be a set of formulae and  $(\mathcal{W}, \mathcal{D})$  be a default theory. We set

- $E_0 = \mathcal{W}$  and
- $E_{i+1} = E_i \cup \{\gamma \mid \frac{\alpha : \beta}{\gamma} \in \mathcal{D}, E_i \models \alpha, \text{ and } \neg\beta \notin \Gamma\}$

Then  $\Gamma$  is an **extension** of  $(\mathcal{W}, \mathcal{D})$  if  $\Gamma = \bigcup_{i \geq 0} Th(E_i)$ , and

$\psi$  is **consistent with**  $(\mathcal{W}, \mathcal{D})$  if there is an extension  $\Gamma$  of  $(\mathcal{W}, \mathcal{D})$  with  $\psi \in \Gamma$

**Remarks:** • **careful:** in the definition of  $E_i$ ,  $\Gamma$  is already used, intuitively to translate “it is safe to assume  $\beta$ ”

- since we are only adding consequents of defaults in  $\mathcal{D}$ , every extension  $\Gamma$  is of the form  $Th(\mathcal{W} \cup Con(\hat{\mathcal{D}}))$  for some  $\hat{\mathcal{D}} \subseteq \mathcal{D}$  and where
  - $Pre(\mathcal{D})$  denotes the set of **pre-requisites** of defaults in  $\mathcal{D}$ ,
  - $Jus(\mathcal{D})$  denotes the set of **justifications** of defaults in  $\mathcal{D}$ , and
  - $Con(\mathcal{D})$  denotes the set of **consequents** of defaults in  $\mathcal{D}$ .

## Propositional Default Logic: Semantics – Examples

- for the default theory

–  $\mathcal{W} = \{\text{Bird}\}$

–  $\mathcal{D} = \{\frac{\text{Bird} : \text{Normal}}{\text{Flies}}\}$ ,

– there is only one extension:  $\{\text{Bird}, \text{Flies}\}$ , and thus, e.g.,

– Flies is consistent with  $(\mathcal{W}, \mathcal{D})$

- for the default theory

–  $\mathcal{W} = \{\text{Bird}, \neg\text{Normal}\}$

–  $\mathcal{D} = \{\frac{\text{Bird} : \text{Normal}}{\text{Flies}}\}$ ,

– there is only one extension:  $\{\text{Bird}, \neg\text{Normal}\}$ , and thus

– Flies is **not** consistent with  $(\mathcal{W}, \mathcal{D})$

### Propositional Default Logic: Semantics – Examples

- for the default theory
  - $\mathcal{W} = \{\text{Bat} \vee \text{Bird}\}$
  - $\mathcal{D} = \left\{ \frac{\text{Bird} : \text{Flies}}{\text{Flies}}, \frac{\text{Bat} : \text{Flies}}{\text{Flies}} \right\}$ ,
  - there is only one extension:  $\{\text{Bat} \vee \text{Bird}\}$ , and thus, e.g.,
  - Flies is **not** consistent with  $(\mathcal{W}, \mathcal{D})$
- for the default theory
  - $\mathcal{W} = \{\text{Penguin}, \text{Penguin} \Rightarrow \text{Bird}\}$
  - $\mathcal{D} = \left\{ \frac{\text{Bird} : \text{Flies}}{\text{Flies}}, \frac{\text{Penguin} : \neg \text{Flies}}{\neg \text{Flies}} \right\}$ ,
  - there are **two extensions**:
  - $\{\text{Penguin}, \text{Bird}, \text{Flies}\}$ , and
  - $\{\text{Penguin}, \text{Bird}, \neg \text{Flies}\}$ ,
  - hence both Flies and  $\neg \text{Flies}$  are consistent with  $(\mathcal{W}, \mathcal{D})$

### Propositional Default Logic: Semantics – Examples

- for the default theory
  - $\mathcal{W} = \{\text{Penguin}, \text{Penguin} \Rightarrow \text{Bird}\}$
  - $\mathcal{D} = \left\{ \frac{\text{Bird} : \text{Winged}}{\text{Winged}}, \frac{\text{Penguin} : \neg \text{Flies}}{\neg \text{Flies}}, \frac{\text{Winged} : \text{Flies}}{\text{Flies}} \right\}$ ,
  - there are also **two extensions**, one that contains
  - Penguin, Bird, Winged and Flies, and one that contains
  - Penguin, Bird, Winged and  $\neg \text{Flies}$
- the default theory
  - $\mathcal{W} = \{P\}$
  - $\mathcal{D} = \left\{ \frac{P : \neg Q}{Q} \right\}$
  - has **no extension**: if  $E$  were an extension,
    - \*  $E$  had to be of the form  $E = \bigcup_{i \geq 0} Th(E_i)$ , and
    - \* if  $Q \in E$ , then we have “put” it into some  $E_i$  because  $Q \notin E \rightsquigarrow$  contradiction
    - \* if  $Q \notin E$ , then we had to put  $Q$  into  $E_1$ , and thus  $Q \in E \rightsquigarrow$  contradiction

### Propositional Default Logic: Semantics – Observations

- ⇒ a default theory can have no or more than one extension!
- ⇒ as we have seen, both  $\psi$  and  $\neg \psi$  can be consistent with a single  $(\mathcal{W}, \mathcal{D})$
- ⇒ default logic is **non-monotonic** because there are  $(\mathcal{W}, \mathcal{D})$  such that
  - $\psi$  is consistent with  $(\mathcal{W}, \mathcal{D})$ , but
  - $\psi$  is **not** consistent with  $(\mathcal{W} \cup \{\phi\}, \mathcal{D})$
- ⇒ how can we **compute extensions**?

### Propositional Default Logic: Computation of Extensions

Input:  $(\mathcal{W}, \mathcal{D})$ , both finite  
 First test: if  $\mathcal{W}$  is not satisfiable, Return  $Th(\mathcal{W})$   
 Init:  $E_0 := \mathcal{W}$ ,  $Used := \emptyset$ ,  $i := 0$   
 Repeat choose some  $\hat{\mathcal{D}} \subseteq \left\{ \frac{\alpha : \beta}{\gamma} \in \mathcal{D} \mid E_i \models \alpha \right\}$   
 set  $E_{i+1} := E_i \cup Con(\hat{\mathcal{D}})$ ,  $Used := Used \cup \hat{\mathcal{D}}$ ,  $i := i + 1$   
 Until Return  $E := Th(E_i)$  if  
 1. for all  $\frac{\alpha : \beta}{\gamma} \in Used : \neg \beta \notin E$   
 2. for all  $\frac{\alpha : \beta}{\gamma} \in \mathcal{D} \setminus Used : \neg \beta \in E$  or  $\alpha \notin E$   
 Stop, return NIL if there is a  $\frac{\alpha : \beta}{\gamma} \in Used : E_i \models \neg \beta$

- this algorithm always terminates (we are in propositional logic)
- and, for each extension  $E$  of  $(\mathcal{W}, \mathcal{D})$ , it can choose a suite of sets  $\hat{\mathcal{D}}$  such that it returns  $E$
- hence it indeeds computes (in a non-deterministic way) all extensions of  $(\mathcal{W}, \mathcal{D})$

### Propositional Default Logic: Computation of Extensions

Can this algorithm be enhanced? Yes, we can choose  $\hat{\mathcal{D}}$  more efficiently, i.e., before computing  $E_{i+1}$ , we can estimate whether our choice of  $\hat{\mathcal{D}}$  is ok:

**Input:**  $(\mathcal{W}, \mathcal{D})$ , a default theory  
**First test:** if  $\mathcal{W}$  is not satisfiable, **Return**  $Th(\mathcal{W})$   
**Init:**  $E_0 := \mathcal{W}$ ,  $Used := \emptyset$ ,  $J_0 := \emptyset$ ,  $i := 0$   
**Repeat** **set**  $\mathcal{D}_i := \{\frac{\alpha : \beta}{\gamma} \in \mathcal{D} \mid E_i \models \alpha\}$   
 choose some  $\hat{\mathcal{D}} \subseteq \mathcal{D}_i$  **that satisfies**  
 for all  $\beta \in Jus(\hat{\mathcal{D}})$ :  $E_i \cup Con(\hat{\mathcal{D}}) \cup J_i \cup \neg Jus(\mathcal{D}_i \setminus \hat{\mathcal{D}}) \not\models \neg\beta$   
**set**  $E_{i+1} := E_i \cup Con(\hat{\mathcal{D}})$   
**set**  $J_{i+1} := J_i \cup \neg Jus(\mathcal{D}_i \setminus \hat{\mathcal{D}})$   
**set**  $Used := Used \cup \hat{\mathcal{D}}$ ,  $i := i + 1$   
**Return**  $E := Th(E_i)$  if 1. for all  $\frac{\alpha : \beta}{\gamma} \in Used$ :  $\neg\beta \notin E$   
 2. for all  $\frac{\alpha : \beta}{\gamma} \in \mathcal{D} \setminus Used$ :  $\neg\beta \in E$  or  $\alpha \notin E$   
**Stop**, return NIL if there is a  $\frac{\alpha : \beta}{\gamma} \in Used$ :  $E_i \models \neg\beta$

### Propositional Default Logic: Computation of Extensions

**Explanation:**

- $E_i$  is always a subset of  $\mathcal{W} \cup Con(\mathcal{D})$
- $J_i$  is always a subset of  $\neg Jus(\mathcal{D})$ , where  $\neg Jus(\mathcal{D}) = \{\neg\beta \mid \beta \in Jus(\mathcal{D})\}$
- the idea of the additional set  $J_i$  is as follows:
  - if  $\frac{\alpha : \beta}{\gamma} \in \mathcal{D}_i \setminus \hat{\mathcal{D}}$ , then  $\frac{\alpha : \beta}{\gamma}$  could have been “applied”, but was not.
  - This is only acceptable if its justification is not consistent with the final extension, i.e.,  $\neg\beta \in E$ , and thus
  - we add  $\neg\beta$  to  $J_i$  and take  $J_i$  into account in the additional condition of  $\hat{\mathcal{D}}$
  - since we will add  $\neg Jus(\mathcal{D}_i \setminus \hat{\mathcal{D}})$  to  $J_i$ , we can also take  $\neg Jus(\mathcal{D}_i \setminus \hat{\mathcal{D}})$  into account in the additional condition of  $\hat{\mathcal{D}}$
- this algorithm always terminates (we are in propositional logic)
- and, for each extension  $E$  of  $(\mathcal{W}, \mathcal{D})$ , it can choose  $\hat{\mathcal{D}}$ s such that it returns  $E$
- hence it indeeds computes (in a non-deterministic way) all extensions of  $(\mathcal{W}, \mathcal{D})$

### Propositional Default Logic: Summary

So far, we have

- seen a logic with defaults, and
- defined extensions and seen algorithms that compute extensions...so what's next?
- let's consider again the example
  - $\mathcal{W} = \{\text{Penguin}, \text{Penguin} \Rightarrow \text{Bird}\}$
  - $\mathcal{D} = \{\frac{\text{Bird} : \text{Flies}}{\text{Flies}}, \frac{\text{Penguin} : \neg\text{Flies}}{\neg\text{Flies}}\}$ ,
  - we can verify that there are **two extensions**, one that contains
  - Penguin, Bird and Flies, and one that contains
  - Penguin, Bird and  $\neg\text{Flies}$
- **Problem:** given that Penguin is **more specific** than Bird, we should
  - activate the penguin-default **before** the bird-default or
  - **prefer** the penguin-default **over** the bird-default,
  - thus accepting only the Penguin, Bird and  $\neg\text{Flies}$  extension

### Propositional Default Logic: Preferences/Prioritisation

In the literature, a variety of different approaches to solve this problem exist,

- they all involve some ordering of defaults, i.e.,
  - giving them priorities, and
  - adapting the definition of extensions to take into account priorities
- they have different (dis)advantages
- here, we discuss an approach that fits nicely with the remainder of the course:
  - we extend defaults from propositional ones to description logic defaults, and
  - use **subsumption** between concepts as a natural indicator for prioritisation/being more specific than
- more precisely, we
  - extend syntax and semantics of defaults, e.g.,  $\frac{a : \text{Bird} \mid a : \text{Flies}}{a : \text{Flies}}$  and
  - extend the algorithm for the computation of extensions

## Description Logic Defaults – an Example

Remarks: in the following, we consider default theories  $(\mathcal{W}, \mathcal{D})$ :

- where the world description  $\mathcal{W} = (\mathcal{T}, \mathcal{A})$  consists of
  - an acyclic set of  $\mathcal{ALC}$  concept definitions  $\mathcal{T}$  (see Section 7) and
  - an  $\mathcal{ALC}$  ABox  $\mathcal{A}$
- for defaults, we can think of starting with defaults of the form  $\frac{C : D}{a : E}$  for concepts  $C, D$ , and  $E$  and *instantiate* them with all object names from the ABox, i.e., if  $a, b$  are objects occurring in  $\mathcal{A}$  and

$$\frac{a : C \mid a : D}{a : E} \in \mathcal{D}, \text{ then we also have } \frac{b : C \mid b : D}{b : E} \in \mathcal{D}$$

- Example:
- $\mathcal{T} = \{\text{Penguin} \sqsubseteq \text{Bird}\}$
  - $\mathcal{A} = \{a : \text{Penguin}\}$
  - $\mathcal{D} = \left\{ \frac{a : \text{Bird} \mid a : \text{Flies}}{a : \text{Flies}}, \frac{a : \text{Penguin} \mid a : \neg \text{Flies}}{a : \neg \text{Flies}} \right\}$ ,

## Description Logic Defaults – Extensions

We start with defining extensions *without prioritisation*:

Extensions: let  $\mathcal{T}$  be a TBox,  $\mathcal{A}'$  an ABox consistent with  $\mathcal{T}$ , and  $(\mathcal{T}, \mathcal{A}, \mathcal{D})$  a description logical default theory. We set

$$\bullet E_0 = \{a : C \mid \mathcal{T}, \mathcal{A} \models a : C\} \text{ and}$$

$$\bullet E_{i+1} = E_i \cup \left\{ a : E \mid \frac{a : C \mid a : D}{a : E} \in \mathcal{D}, \mathcal{T}, E_i \models a : C, \text{ and } \mathcal{T}, \mathcal{A}' \not\models a : \neg D \right\}$$

Then  $\mathcal{A}'$  is an extension of  $(\mathcal{T}, \mathcal{A}, \mathcal{D})$  if

$$\mathcal{A}' = \{a : C \mid \mathcal{T}, \bigcup_{i \geq 0} \{E_i\} \models a : C\}$$

## Description Logic Defaults – Extensions

Next, we extend this definition to take into account *prioritisation*, for which we define **active** defaults:

Active: a default  $\frac{a : C \mid a : D}{a : E}$  is **active** in  $\mathcal{T}, \mathcal{A}$  if  $\mathcal{T}, \mathcal{A} \models a : C$  and  $\mathcal{T}, \mathcal{A} \not\models a : \neg D$

Please note that “active” only refers to the current ABox and not to an extension, and thus being active can be decided (how?)!

## Description Logic Defaults – P-Extensions

P-Extensions: let  $\mathcal{T}$  be a TBox,  $\mathcal{A}'$  an ABox consistent with  $\mathcal{T}$ , and  $(\mathcal{T}, \mathcal{A}, \mathcal{D})$  a description logical default theory. We set

$$\bullet E_0 = \{a : C \mid \mathcal{T}, \mathcal{A} \models a : C\} \text{ and}$$

$$\bullet E_{i+1} = E_i \cup \left\{ a : E \mid \frac{a : C \mid a : D}{a : E} \in \mathcal{D} \text{ is active in } \mathcal{T}, \mathcal{A}' \right.$$

**and for all**  $\frac{a : C' \mid a : D'}{a : E'} \in \mathcal{D}$ ,

if  $\mathcal{T} \models C' \sqsubseteq C$ , and  $\mathcal{T} \not\models C \sqsubseteq C'$ ,

then  $\frac{a : C' \mid a : D'}{a : E'}$  is not active in  $\mathcal{T}, E_i$

Then  $\mathcal{A}'$  is a **p-extension** of  $(\mathcal{T}, \mathcal{A}, \mathcal{D})$  if

$$\mathcal{A}' = \{a : C \mid \mathcal{T}, \bigcup_{i \geq 0} \{E_i\} \models a : C\}$$

## Description Logic Defaults – P-Extensions

Finally, we say that an assertion  $a : C$  is **consistent** with a prioritised  $\mathcal{ALC}$  default theory  $(\mathcal{T}, \mathcal{A}, \mathcal{D})$  if there exists a p-extension  $E$  of  $(\mathcal{T}, \mathcal{A}, \mathcal{D})$  with  $\mathcal{T}, E \models a : C$ .

- Remarks:**
- the “p” in “p-extension” stands for “prioritised”.
  - as for propositional default theories,  $\mathcal{ALC}$  default theories can have several or no extensions
  - the Penguin example has exactly one extension:
    - we start with  $E_0 = \mathcal{A}$ , and then
    - both defaults are **active** in  $\mathcal{T}, E_0$ , but Penguin is more specific, thus the Bird-default is not “applied”, i.e.,  $E_1 = \mathcal{A} \cup \{a : \neg\text{Flies}\}$ , and then
    - stop since (trivially)  $\mathcal{T}, E_i \models a : \neg\text{Flies}$  with the extension
- $$\mathcal{A}' = \{a : C \mid \mathcal{T}, \{a : \text{Penguin}, a : \neg\text{Flies}\} \models a : C\}$$
- $$\{a : \text{Penguin}, a : \neg\text{Flies}, a : (\text{Bird} \sqcap \neg\text{Flies}), \dots\}$$

## Description Logic Defaults – Sub-Summary

In this section, we have seen

- propositional defaults: their syntax and semantics, given via extensions
- a “naive” and an “enhanced” algorithm for the computation of extensions
- that there is a problem with “one default being more specific than another one”, which we solved in
- description logic default theories:
  - they are a bit more expressive, i.e., we have ABoxes, relations between individuals, etc., and
  - we can use subsumption between the pre-requisites of defaults to **prioritise** defaults: intuitively, if we could “apply” two defaults with pre-requisites  $a : C$  and  $a : C'$  where  $C$  is subsumed by  $C'$ , we only apply the one with  $a : C$ , and only later possibly the other one!
- finally, we adapt the (naive) algorithm for the computation of extensions of propositional default theories to description logic default theories:

## Description Logic Defaults – How to Compute Extensions

- Input:**  $(\mathcal{T}, \mathcal{A}, \mathcal{D})$ , a (finite)  $\mathcal{ALC}$  default theory
- First test:** if  $\mathcal{A}$  is not consistent with  $\mathcal{T}$ , **Return “inconsistent”**
- Init:**  $E_0 := \mathcal{A}$ ,  $Used := \emptyset$ ,  $i := 0$
- Repeat**  $\mathcal{D}_i := \left\{ \frac{a : C \mid a : D}{a : E} \in \mathcal{D} \mid \frac{a : C \mid a : D}{a : E} \text{ is active in } \mathcal{T}, E_i \right.$   
 and for all  $\frac{a : C' \mid a : D'}{a : E'} \in \mathcal{D}$  active in  $\mathcal{T}, E_i$ ,  
 if  $\mathcal{T} \models C' \sqsubseteq C$ , then  $\mathcal{T} \models C \sqsubseteq C'$   
 choose some  $\hat{\mathcal{D}} \subseteq \mathcal{D}_i$   
 set  $E_{i+1} := E_i \cup \{a : E \mid \frac{a : C \mid a : D}{a : E} \in \hat{\mathcal{D}}\}$   
 set  $Used := Used \cup \hat{\mathcal{D}}$ ,  $i := i + 1$   
**Return**  $E := \{a : C \mid \mathcal{T}, E_{i+1} \models a : C\}$  if
1. for all  $\frac{a : C \mid a : D}{a : E} \in Used$ :  $\mathcal{T}, E \not\models a : \neg D$
  2. for all  $\frac{a : C \mid a : D}{a : E} \in \mathcal{D} \setminus Used$ :  $\mathcal{T}, E \models a : \neg D$  or  $\mathcal{T}, E \neg \models a : C$
- Stop**, return NIL if there is a  $\frac{a : C \mid a : D}{a : E} \in Used$ :  $\mathcal{T}, E_{i+1} \models a : \neg D$

## Description Logic Defaults – Example Extensions

Apply the algorithm to the following example  $\mathcal{ALC}$  default theory, where  $KP$  stands for KingPenguin,  $P$  for Penguin, and  $B$  for Bird

- $\mathcal{T} = \{ KP \dot{\sqsubseteq} P \sqcap \exists \text{has.GoldSwish}, P \dot{\sqsubseteq} B, B \dot{\sqsubseteq} \text{Animal}, \text{Runs} \dot{\sqsubseteq} \text{Locom} \sqcap \neg \text{Hops} \sqcap \neg \text{Flies} \}$
- $\mathcal{A} = \{ a : P, b : P, (b, c) : \text{has}, c : \text{GoldSwish} \}$
- $\mathcal{D} = \left\{ \frac{x : KP \mid x : \text{Runs}}{x : \text{Runs}}, \frac{x : P \mid x : \text{Hops}}{x : \text{Hops}}, \frac{x : P \mid x : \neg \text{Flies}}{x : \neg \text{Flies}}, \frac{x : B \mid x : \text{Flies}}{x : \text{Flies}} \mid x \in \{a, b, c\} \right\}$
- to compute  $\mathcal{D}_0$ , we first have to find the defaults that are active in  $E_0 (= \mathcal{A})$ , and then find the “most specific ones”
- next, we have to choose some  $\hat{\mathcal{D}} \subseteq \mathcal{D}_0$ , which yields  $E_1$  and a new  $Used$ , etc.
- we check whether we are done – if not, we compute  $\mathcal{D}_1$  and choose another  $\hat{\mathcal{D}}$ , etc...

- In this section, we have seen formalisms for **non-monotonic** inferences:
  - propositional default logic and
  - description logic defaults
- discussed syntax and semantics and
- seen algorithms for the computation of extensions:
  - a naive one for propositional default logic,
  - an enhanced one for propositional default logic, and
  - one for prioritised description logic defaults
- There are a variety of other default logics and a
- large variety of other **non-monotonic** formalisms, e.g.
  - circumscription,
  - formalisms with closed world assumption/negation as failure,
  - auto-epistemic logics
  - see <http://plato.stanford.edu/entries/logic-nonmonotonic/>