2
A Basic Description Logic

In this chapter, we introduce and explain the basic notions of Description Logic, including syntax, semantics and reasoning services, and we explain how the latter are used in applications.

2.1 The concept language of the DL $\mathcal{ALC}$

In this section, we will describe the central notions of Description Logic first on an intuitive level and then on a more precise level. As a running example, we use the domain of university courses and teaching, and we will use a conceptualisation given informally, in graphical form, in Figure 2.1. Please note that this is *one* way of viewing university teaching – which might be very different from the reader's way of viewing it. Also, as it is an informal representation, different readers may interpret arrows in different ways; that is, our representation does not come with a well-defined semantics that would inform us in an unambiguous way how to interpret the different arrows.[1] In the next sections, we will describe our way of viewing university teaching in a DL knowledge base, thereby establishing some constraints on the meaning of terms like "Professor" and "teaches" used in Figure 2.1 and throughout this section.

In Description Logic, we assume that we want to describe some abstraction of some domain of interest, and that this abstraction is populated by *elements*.[2] We use three main building blocks to describe these elements:

- *Concepts* represent sets of elements and can be viewed as unary pred-

[1] Our graphical representation looks somewhat similar to an extended ER diagram, for which such a well-defined semantics has been specified [Che76, CLN94].

[2] We have chosen the term "elements" rather than "individuals" or "objects" to prevent the reader from making false assumptions.

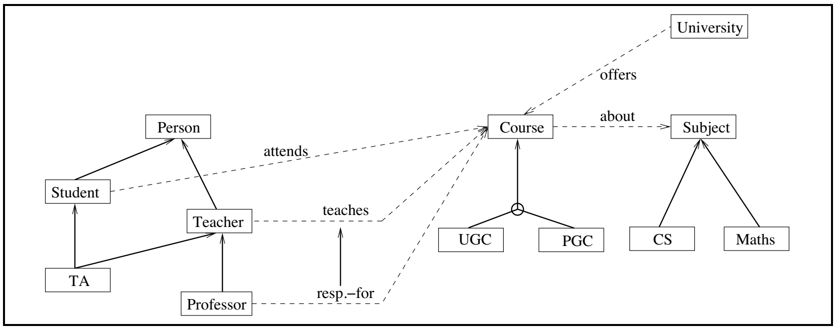

Fig. 2.1. An informal, graphical view of our running example.

icates. Concepts are built from *concept names* and *role names* (see below) using the constructors provided by the DL used. The set a concept represents is called its *extension*. For example, Person and Course are concept names, and m is an element in the extension of Person and $\mathsf{c6}$ is in the extension of Course. To make our life a bit easier, we often use "is a" as an abbreviation for "is in the extension of" as, for example, in "m is a Person".

- *Role names* stand for binary relations on elements and can be viewed as binary predicates. If a role r relates one element with another element, then we call the latter one an *r-filler* of the former one. For example, if m $\mathit{teaches}$ $c6$, then we call $c6$ a $\mathit{teaches}$-filler of m.

At the heart of a specific DL, we find a *concept language*; that is, a formal language that allows us to build *concept descriptions* (and *role descriptions*) from concept names, role names, and possibly other primitives. For example, $\mathsf{Person} \sqcap \exists \mathit{teaches}.\mathsf{Course}$ is such a concept description built from the concept names Person and Course and the role name $\mathit{teaches}$. Next, we formalise the exact meaning of these notions.

Definition 2.1. Let $\mathbf{C}$ be a set of *concept names* and $\mathbf{R}$ be a set of *role names* disjoint from $\mathbf{C}$. The set of *$\mathcal{ALC}$ concept descriptions* over $\mathbf{C}$ and $\mathbf{R}$ is inductively defined as follows:

- Every concept name is an $\mathcal{ALC}$ concept description.
- $\top$ and $\bot$ are $\mathcal{ALC}$ concept descriptions.
- If C and D are $\mathcal{ALC}$ concept descriptions and r is a role name, then the following are also $\mathcal{ALC}$ concept descriptions:

  $C \sqcap D$ (conjunction),

$C \sqcup D$ (disjunction),
$\neg C$ (negation),
$\exists r.C$, (existential restriction), and
$\forall r.C$ (value restriction).

As usual, we use parentheses to clarify the structure of concepts.

Definition 2.1 fixes the *syntax* of $\mathcal{ALC}$ concept descriptions; that is, it allows us to distinguish between expressions that are well formed and those that are not. For example, $\exists r.C$ and $A \sqcap \exists r.\forall s.(E \sqcap \neg F)$ are $\mathcal{ALC}$ concept descriptions, whereas $\exists C$ and $\forall s.s$ are not; in the former case since $\exists C$ is missing a role name, and in the latter case since s cannot be both a concept and a role name.

Next, we will introduce some DL parlance and abbreviations. First, we often use "$\mathcal{ALC}$ concept" as an abbreviation of "$\mathcal{ALC}$ concept description" and, if it is clear from the context that we talk about $\mathcal{ALC}$ concepts, we may even drop the $\mathcal{ALC}$ and use "concepts" for "$\mathcal{ALC}$ concepts". Moreover, when clear from the context, $\mathbf{C}$ and $\mathbf{R}$ are not mentioned explicitly.

Remark. Please note that in the DL setting a concept is, basically, a string, and is not to be confused with the notion of a "concept" in the sense of an abstract or general idea from philosophy. When we use a DL in an application, we may use a DL concept to describe a relevant application "concept", but the latter is far more subtle and intricate than the former.

Second, we sometimes distinguish between *atomic* and *compound* (also called complex) concepts. An atomic concept consists of a single lexical token, i.e., in $\mathcal{ALC}$, a concept name, $\top$, or $\bot$. A compound concept is constructed using at least one of the available operators, i.e., in $\mathcal{ALC}$, $\sqcap$, $\sqcup$, $\neg$, $\exists$ and $\forall$. In the following, we will use upper case letters A, B for concept names, upper case letters C, D for possibly compound concepts, and lower case letters r, s for role names.

Before we define the semantics, i.e., the meaning of concepts and roles, we will present an intuitive reading for compound concepts.

- A *negation* is written $\neg\mathsf{Student}$ and can be read as "not $\mathsf{Student}$". It describes everything that is not in the extension of $\mathsf{Student}$.
- A *conjunction* is written $\mathsf{Student} \sqcap \mathsf{Teacher}$ and can be read as "$\mathsf{Student}$ and $\mathsf{Teacher}$". It describes those elements that are in the extension of both $\mathsf{Student}$ and $\mathsf{Teacher}$.

- A *disjunction* is written $\mathsf{Student} \sqcup \mathsf{Teacher}$ and can be read as "Student or Teacher". It describes those elements that are in the extension of either Student or Teacher, or both.
- A *value restriction* is written $\forall \textit{teaches}.\mathsf{Course}$ and can be read as "all *teaches*-fillers are Courses". It describes all those elements that have only elements in Course related to them via *teaches*. It is written with an upside-down A because of the "**a**ll" in its reading and its close relationship with universal quantification in first-order logic.
- An *existential restriction* is written $\exists \textit{teaches}.\mathsf{Course}$ and can be read as "there exists a *teaches*-filler which is a Course". It describes all elements that have at least one *teaches*-filler that is in Course. It is written with a backwards E because of the "there **e**xists" in its reading and its close relationship with existential quantification in first-order logic.

Now, to fix the meaning of concepts and roles, we make use of an *interpretation*, that is, a structure that:

- consists of a non-empty set called its *interpretation domain*. We call the elements of this interpretation domain simply "elements", but they are sometimes called individuals or objects elsewhere;
- fixes, for each concept name, its extension – that is, it tells us, for each concept name, which of the elements is (or isn't) in the extension of this concept;
- fixes, for each role name, its extension – that is, it tells us, for each role name, which pairs of elements are related to each other by this role.

Interpretations, as well as the extension of concept descriptions, are defined next.

Definition 2.2. An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a non-empty set $\Delta^\mathcal{I}$, called the *interpretation domain*, and a mapping $\cdot^\mathcal{I}$ that maps

- every concept name $A \in \mathbf{C}$ to a set $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$, and
- every role name $r \in \mathbf{R}$ to a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$.

The mapping $\cdot^{\mathcal{I}}$ is extended to $\top$, $\bot$ and compound concepts as follows:

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

We call

- $C^{\mathcal{I}}$ the *extension of* C in $\mathcal{I}$,
- $b \in \Delta^{\mathcal{I}}$ an *r-filler of a in $\mathcal{I}$* if $(a,b) \in r^{\mathcal{I}}$.

Please note that an interpretation is not restricted other than as explicitly specified above: its domain must be non-empty, but can be of any cardinality, and in particular it can be infinite; the extension of a concept can have any number of elements between "none" and "all"; and a role can relate any number of pairs of elements, from "none" to "all".

Also, please note that $A^{\mathcal{I}}$ stands for the result of applying the mapping $\cdot^{\mathcal{I}}$ to the concept name A; this is an unusual way of writing mappings, yet it is quite helpful and ink-saving. In the past, DL researchers have used different notations such as $\mathcal{I}(A)$ or $[[A]]_{\mathcal{I}}$, but the one used here is the one that stuck.

As an example, let us consider the following interpretation $\mathcal{I}$:

$$\begin{aligned}
\Delta^{\mathcal{I}} &= \{m, c6, c7, et\},\\
\mathsf{Teacher}^{\mathcal{I}} &= \{m\},\\
\mathsf{Course}^{\mathcal{I}} &= \{c6, c7, et\},\\
\mathsf{Person}^{\mathcal{I}} &= \{m, et\},\\
\mathsf{PGC}^{\mathcal{I}} &= \{c7\},\\
\mathit{teaches}^{\mathcal{I}} &= \{(m, c6), (m, c7), (et, et)\}.
\end{aligned}$$

We can easily modify $\mathcal{I}$ to obtain other interpretations $\mathcal{I}_1$, $\mathcal{I}_2$ etc., by adding or removing elements and changing the interpretation of concept and role names. An interpretation is often conveniently drawn as a directed, labelled graph with a node for each element of the interpretation domain and labelled as follows: a node is labelled with all concept names the corresponding element of the interpretation domain belongs to, and we find an edge from one node to another labelled with r if the element

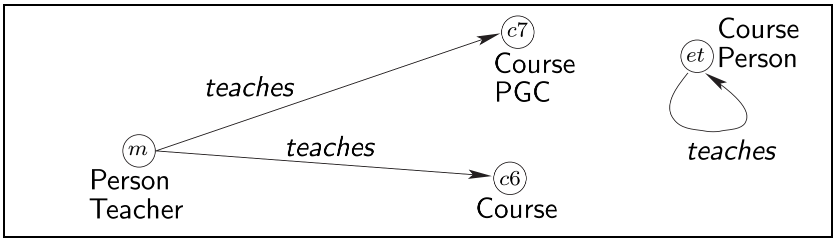

Fig. 2.2. A graphical representation of the example interpretation $\mathcal{I}$.

corresponding to the latter node is an r-filler of the element corresponding to the former node. As an example, Figure 2.2 shows a graphical representation of $\mathcal{I}$.

Let us take a closer look at $\mathcal{I}$. By definition, all elements are in the extension of $\top$, and no element is in the extension of $\bot$. The elements m and et are, for example, in the extension of Person, and et is a *teaches*-filler of itself. If we extend $\mathcal{I}$ to compound concepts as specified in Definition 2.2, then we can see that, for example, m is in the extension of Person$\sqcap$Teacher, and $c6$ is in the extension of Course$\sqcap\neg$Person, because $c6$ is a Course and *not* a Person. Similarly, for existential restrictions, m and et are in the extension of $\exists$*teaches*.Course, but only m is in the extension of $\exists$*teaches*.$\neg$Person. For value restrictions, all elements are in the extension of $\forall$*teaches*.Course: for m and et, this is clear, and for $c6$ and $c7$, this is because they do not have *any* *teaches*-fillers, and hence *all* their *teaches*-fillers vacuously satisfy any condition we may impose. In general, if an element has no r-filler, then it is in the extension of $\forall r.C$ for any concept C. In contrast, m is *not* in the extension of $\forall$*teaches*.(Course $\sqcap$ PGC) because m has $c6$ as a *teaches*-filler that is *not* in the extension of Course$\sqcap$PGC since it is not a PGC. At this stage, we repeat our invitation to the reader to consider some more interpretations and concepts and determine which element is in the extension of which concept.

We can also investigate extensions of more compound concept descriptions such as Person$\sqcap\exists$*teaches*.(Course$\sqcap\neg$PGC): for example, m is in the extension of this concept since it is in the extension of both conjuncts: by definition of $\mathcal{I}$, it is in the extension of Person, and it also is in the extension of the second conjunct, because it has a *teaches*-successor, $c6$, that is in the extension of (Course $\sqcap$ $\neg$PGC).

We have been rather generous in our syntax definition since we provide a non-minimal set of concept constructors, i.e., one with "syntactic sugar". The following lemma makes this observation precise.

Lemma 2.3. *Let $\mathcal{I}$ be an interpretation, C, D concepts, and r a role. Then*

$$\begin{array}{lrcl}
\text{(i)} & \top^{\mathcal{I}} & = & (C \sqcup \neg C)^{\mathcal{I}}, \\
\text{(ii)} & \bot^{\mathcal{I}} & = & (C \sqcap \neg C)^{\mathcal{I}}, \\
\text{(iii)} & (\neg\neg C)^{\mathcal{I}} & = & C^{\mathcal{I}}, \\
\text{(iv)} & \neg(C \sqcap D)^{\mathcal{I}} & = & (\neg C \sqcup \neg D)^{\mathcal{I}}, \\
\text{(v)} & \neg(C \sqcup D)^{\mathcal{I}} & = & (\neg C \sqcap \neg D)^{\mathcal{I}}, \\
\text{(vi)} & (\neg(\exists r.C))^{\mathcal{I}} & = & (\forall r.\neg C)^{\mathcal{I}}, \\
\text{(vii)} & (\neg(\forall r.C))^{\mathcal{I}} & = & (\exists r.\neg C)^{\mathcal{I}}.
\end{array}$$

Proof. These equations follow rather immediately from Definition 2.2:

By definition, $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$, and $(C \sqcup \neg C)^{\mathcal{I}} = C^{\mathcal{I}} \cup (\neg C)^{\mathcal{I}} = C^{\mathcal{I}} \cup (\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}) = \Delta^{\mathcal{I}}$. Hence Equation (i) holds.

Equation (ii) can be proven analogously and is left to the reader.

For Equation (iii), $(\neg\neg C)^{\mathcal{I}} = (\Delta^{\mathcal{I}} \setminus (\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}))$, which is of course the same as $C^{\mathcal{I}}$.

For Equation (iv), which is also known as one of de Morgan's laws, we have $\neg(C \sqcap D)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus (C \sqcap D)^{\mathcal{I}}$. Now $d \in \Delta^{\mathcal{I}} \setminus (C \sqcap D)^{\mathcal{I}}$ if and only if $d \notin C^{\mathcal{I}}$ or $d \notin D^{\mathcal{I}}$ (or both), which is the case if and only if $d \in (\neg C)^{\mathcal{I}} \cup (\neg D^{\mathcal{I}}) = (\neg C \sqcup \neg D)^{\mathcal{I}}$.

Equation (v), another of de Morgan's laws, can be proven analogously.

For Equation (vi), by definition of the semantics,

$$\begin{array}{l}
(\neg(\exists r.C))^{\mathcal{I}} \\
= \Delta^{\mathcal{I}} \setminus \{d \in \Delta^{\mathcal{I}} \mid \text{there is an } e \in \Delta^{\mathcal{I}} \text{ with } (d,e) \in r^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}}\} \\
= \{d \in \Delta^{\mathcal{I}} \mid \text{there is no } e \in \Delta^{\mathcal{I}} \text{ with } (d,e) \in r^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}}\} \\
= \{d \in \Delta^{\mathcal{I}} \mid \text{for all } e \in \Delta^{\mathcal{I}} \text{ if } (d,e) \in r^{\mathcal{I}} \text{ then } e \notin C^{\mathcal{I}}\} \\
= (\forall r.\neg C)^{\mathcal{I}}.
\end{array}$$

Equation (vii) can be proven analogously and is left to the reader. □

As a consequence of Lemma 2.3, and as we will see later, we can rewrite, e.g., $(C \sqcup D)$ to $\neg(\neg C \sqcap \neg D)$, and thus avoid explicit disjunctions.

2.2 $\mathcal{ALC}$ knowledge bases

If we were to use a DL-based system in an application, we would build concept descriptions that describe relevant notions from this application domain. For example, for a molecular biology application, we would build concepts describing proteins, genes and so on. We would then use these concepts in a *knowledge base*, and we can do this in (at least) four different ways:

(i) As in an encyclopedia, we define the meaning of some concept names in terms of concept descriptions. For example, we can define the meaning of $\mathsf{UG\text{-}Student}$ and $\mathsf{CS\text{-}Teacher}$ using the following equations:[3]

$$\begin{aligned}\mathsf{UG\text{-}Student} &\equiv \mathsf{Student} \sqcap \forall \mathit{attends}.\mathsf{UGC},\\ \mathsf{CS\text{-}Teacher} &\equiv \mathsf{Teacher} \sqcap \exists \mathit{teaches}.(\mathsf{Course} \sqcap \exists \mathit{about}.\mathsf{CS}).\end{aligned}$$

Intuitively, the first equation says that $\mathsf{UG\text{-}Students}$ are those students that attend only UGCs, and the second one says that $\mathsf{CS\text{-}Teacher}$s are those $\mathsf{Teacher}$s that teach some Course about CS.

(ii) We express background knowledge. For example, we can state that an undergraduate course (UGC) cannot be a postgraduate course (PGC), and that a $\mathsf{University}$ necessarily *offers* both UGCs and PGCs, using the following equations:

$$\begin{aligned}\mathsf{UGC} &\sqsubseteq \neg\mathsf{PGC},\\ \mathsf{University} &\sqsubseteq \exists \mathit{offers}.\mathsf{UGC} \sqcap \exists \mathit{offers}.\mathsf{PGC}.\end{aligned}$$

(iii) We assert that individual names stand for instances of (possibly compound) concept descriptions. For example, we can say that Mary stands for an instance of $\mathsf{Teacher} \sqcap \exists \mathit{teaches}.\mathsf{PGC}$ and $\mathsf{CS600}$ stands for an instance of Course.

(iv) We relate individual names by roles. For example, we can say that Mary *teaches* $\mathsf{CS600}$.

Traditionally, we distinguish two parts of a DL knowledge base. The *terminological* part, called the *TBox*, contains statements of the form described in items (i) and (ii), and the *assertional* part, called the *ABox*, contains statements of form described in items (iii) and (iv). If we compare this to databases, then we can view a TBox as a schema because it expresses general constraints on what (our abstraction of) the world looks like. And we can view the ABox as the data since it talks about concrete elements, their properties and their relationships.

2.2.1 $\mathcal{ALC}$ TBoxes

We start by defining the syntax and semantics of TBoxes.

[3] The exact meaning of these equations will be defined later.

Definition 2.4. For C and D possibly compound $\mathcal{ALC}$ concepts, an expression of the form $C \sqsubseteq D$ is called an $\mathcal{ALC}$ *general concept inclusion* and abbreviated *GCI*. We use $C \equiv D$ as an abbreviation for $C \sqsubseteq D,\ D \sqsubseteq C$.

A finite set of GCIs is called an $\mathcal{ALC}$ *TBox*.

An interpretation $\mathcal{I}$ *satisfies* a GCI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An interpretation that satisfies each GCI in a TBox $\mathcal{T}$ is called a *model* of $\mathcal{T}$.

As usual, if it is clear that we are talking about $\mathcal{ALC}$ concepts and TBoxes, we omit "$\mathcal{ALC}$" and use simply TBox or GCI. We will sometimes refer to abbreviations of the form $C \equiv D$ as an *equivalence axiom*, and use *axiom* to refer to either an equivalence axiom or a GCI.

The interpretation $\mathcal{I}$ given in Figure 2.2 satisfies each of the GCIs in

$$\begin{aligned}
\mathcal{T}_1 = \{\mathsf{Teacher} &\sqsubseteq \mathsf{Person},\\
\mathsf{PGC} &\sqsubseteq \neg\mathsf{Person},\\
\mathsf{Teacher} &\sqsubseteq \exists \mathit{teaches}.\mathsf{Course},\\
\exists \mathit{teaches}.\mathsf{Course} &\sqsubseteq \mathsf{Person}\}
\end{aligned}$$

and thus $\mathcal{I}$ is a model of $\mathcal{T}_1$. To verify this, for each GCI $C \sqsubseteq D$, we determine $C^{\mathcal{I}}$ and $D^{\mathcal{I}}$ and then check whether $C^{\mathcal{I}}$ is indeed a subset of $D^{\mathcal{I}}$. For the first GCI, we observe that $\mathsf{Teacher}^{\mathcal{I}} = \{m\} \subseteq \{m, et\} = \mathsf{Person}^{\mathcal{I}}$. Similarly, for the second one, we have $\mathsf{PGC}^{\mathcal{I}} = \{c7\} \subseteq \{c6, c7\} = (\neg\mathsf{Person})^{\mathcal{I}}$. For the third one, m is the only element in the extension of Teacher and also in the extension of $\exists \mathit{teaches}.\mathsf{Course}$, hence it is also satisfied by $\mathcal{I}$. Finally $(\exists \mathit{teaches}.\mathsf{Course})^{\mathcal{I}} = \{m, et\}$ and both m and et are in the extension of Person.

In contrast, $\mathcal{I}$ does not satisfy the GCIs

$$\mathsf{Course} \sqsubseteq \neg\mathsf{Person}, \tag{2.1}$$
$$\exists \mathit{teaches}.\mathsf{Course} \sqsubseteq \mathsf{Teacher}, \tag{2.2}$$

because et is both a Person and a Course, and because et *teaches* some Course, but is not a Teacher.

In general, a TBox $\mathcal{T}$ allows us to distinguish between those interpretations that are and those that are not models of $\mathcal{T}$. In practice, this means that we can use a TBox to restrict our attention to those interpretations that fit our intuitions about the domain. For example, Formula (2.1) should be in our TBox if we think that a Course cannot be a Person, and Formula (2.2) should be in our TBox if we think that only Teachers can teach Courses. In general, the more GCIs our TBox contains, the fewer models it has. This is expressed in the following lemma.

$$\begin{array}{rcll}
\mathcal{T}_{ex} = \{\mathsf{Course} & \sqsubseteq & \neg\mathsf{Person}, & (\mathcal{T}_{ex}.1)\\
\mathsf{UGC} & \sqsubseteq & \mathsf{Course}, & (\mathcal{T}_{ex}.2)\\
\mathsf{PGC} & \sqsubseteq & \mathsf{Course}, & (\mathcal{T}_{ex}.3)\\
\mathsf{Teacher} & \equiv & \mathsf{Person} \sqcap \exists \mathit{teaches}.\mathsf{Course}, & (\mathcal{T}_{ex}.4)\\
\exists \mathit{teaches}.\top & \sqsubseteq & \mathsf{Person}, & (\mathcal{T}_{ex}.5)\\
\mathsf{Student} & \equiv & \mathsf{Person} \sqcap \exists \mathit{attends}.\mathsf{Course}, & (\mathcal{T}_{ex}.6)\\
\exists \mathit{attends}.\top & \sqsubseteq & \mathsf{Person}\ \} & (\mathcal{T}_{ex}.7)
\end{array}$$

Fig. 2.3. The example TBox $\mathcal{T}_{ex}$.

Lemma 2.5. *If $\mathcal{T} \subseteq \mathcal{T}'$ for two TBoxes $\mathcal{T}$, $\mathcal{T}'$, then each model of $\mathcal{T}'$ is also a model of $\mathcal{T}$.*

Proof. The proof is rather straightforward: let $\mathcal{T} \subseteq \mathcal{T}'$ be two TBoxes and $\mathcal{I}$ a model of $\mathcal{T}'$. By definition, $\mathcal{I}$ satisfies all GCIs in $\mathcal{T}'$ and thus, since $\mathcal{T} \subseteq \mathcal{T}'$, also all GCIs in $\mathcal{T}$. Hence $\mathcal{I}$ is, as required, also a model of $\mathcal{T}$. $\square$

Next, in Figure 2.3, we define a TBox $\mathcal{T}_{ex}$ that partially captures our intuition of teaching as presented in Figure 2.1. Axiom $\mathcal{T}_{ex}.4$ is an equivalence that defines a Teacher as a Person who *teaches* a Course. That is, every Teacher is a Person who *teaches* a Course and, vice versa, if a Person *teaches* a Course, then they are a Teacher. Axiom $\mathcal{T}_{ex}.5$ ensures that only Persons can teach a course. As mentioned above, the interpretation depicted in Figure 2.2 is not a model of $\mathcal{T}_{ex}$ since it violates axiom $\mathcal{T}_{ex}.1$.

2.2.2 $\mathcal{ALC}$ ABoxes

Next, we define ABoxes and knowledge bases.

Definition 2.6. Let $\mathbf{I}$ be a set of *individual names* disjoint from $\mathbf{R}$ and $\mathbf{C}$. For $a, b \in \mathbf{I}$ individual names, C a possibly compound $\mathcal{ALC}$ concept, and $r \in \mathbf{R}$ a role name, an expression of the form

- $a : C$ is called an $\mathcal{ALC}$ *concept assertion*, and
- $(a, b) : r$ is called an $\mathcal{ALC}$ *role assertion*.

A finite set of $\mathcal{ALC}$ concept and role assertions is called an $\mathcal{ALC}$ *ABox*.

An interpretation function $\cdot^{\mathcal{I}}$ is additionally required to map every individual name $a \in \mathbf{I}$ to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. An interpretation $\mathcal{I}$ *satisfies*

- a concept assertion $a : C$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and

- a role assertion $(a, b) : r$ if $(a^\mathcal{I}, b^\mathcal{I}) \in r^\mathcal{I}$.

An interpretation that satisfies each concept assertion and each role assertion in an ABox $\mathcal{A}$ is called a *model* of $\mathcal{A}$.

Again, if it is clear that we are talking about $\mathcal{ALC}$ concepts and ABoxes, we omit "$\mathcal{ALC}$" and use simply ABox, concept assertion etc. Moreover, for the sake of brevity, we will occasionally use "individual" as an abbreviation for "individual name".

In Figure 2.4, we present an example ABox $\mathcal{A}_{ex}$ with concept and role assertions. The following interpretation $\mathcal{I}$ is a model of this ABox:

$$\begin{aligned}
\Delta^\mathcal{I} &= \{h, m, c6, p4\},\\
\mathsf{Mary}^\mathcal{I} &= m,\\
\mathsf{Betty}^\mathcal{I} &= \mathsf{Hugo}^\mathcal{I} = h,\\
\mathsf{CS600}^\mathcal{I} &= c6,\\
\mathsf{Ph456}^\mathcal{I} &= p4,\\
\mathsf{Person}^\mathcal{I} &= \{h, m, c6, p4\},\\
\mathsf{Teacher}^\mathcal{I} &= \{h, m\},\\
\mathsf{Course}^\mathcal{I} &= \{c6, p4\},\\
\mathsf{PGC}^\mathcal{I} &= \{p4\},\\
\mathsf{UGC}^\mathcal{I} &= \{c6\},\\
\mathsf{Student}^\mathcal{I} &= \emptyset,\\
\mathit{teaches}^\mathcal{I} &= \{(m, c6), (h, p4)\},\\
\mathit{attends}^\mathcal{I} &= \{(h, p4), (m, p4)\}.
\end{aligned}$$

Please observe that the individual names Hugo and Betty are interpreted as the same element: h. This is allowed by our definition of the semantics. Some logics, including many early description logics, make the so-called *Unique Name Assumption* (UNA) which requires that $a^\mathcal{I} \neq b^\mathcal{I}$ in the case $a \neq b$, and would thus rule out such an interpretation. Throughout this book, we do not make the UNA unless it is stated to the contrary.

We can further observe that, in $\mathcal{I}$, the extension of Teacher has more elements than strictly required by $\mathcal{A}_{ex}$: nothing in $\mathcal{A}_{ex}$ requires m, $c6$ or $p4$ to be in the extension of Teacher. Moreover, $\mathcal{I}$ interprets the concept UGC, although this concept isn't mentioned in $\mathcal{A}_{ex}$. Again, all this is allowed by our definition of the semantics. Also, please note that $\mathcal{I}$ is not a model of the TBox $\mathcal{T}_{ex}$ given in Figure 2.3; for example, $h \in (\mathsf{Person} \sqcap \exists \mathit{attends}.\mathsf{Course})^\mathcal{I}$, but $h \notin \mathsf{Student}$, and thus $\mathcal{I}$ does not satisfy the axiom $\mathcal{T}_{ex}.6$.

Next, we combine TBoxes and ABoxes in knowledge bases: this allows us to specify terms and their meaning in a TBox, and then make use of these in our ABox.

Definition 2.7. An $\mathcal{ALC}$ *knowledge base* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of an $\mathcal{ALC}$ TBox $\mathcal{T}$ and an $\mathcal{ALC}$ ABox $\mathcal{A}$. An interpretation that is both a model of $\mathcal{A}$ and of $\mathcal{T}$ is called a *model* of $\mathcal{K}$.

Hence for an interpretation to be a model of $\mathcal{K}$, it has to satisfy all assertions in $\mathcal{K}$'s ABox and all GCIs in $\mathcal{K}$'s TBox. As an example, consider $\mathcal{K}_{ex} = (\mathcal{T}_{ex}, \mathcal{A}_{ex})$, with $\mathcal{T}_{ex}$ and $\mathcal{A}_{ex}$ as presented in Figures 2.3 and 2.4. As mentioned earlier, the interpretation $\mathcal{I}$ given above is *not* a model of $\mathcal{K}_{ex}$, because $\mathcal{I}$ does not satisfy two of the axioms in $\mathcal{T}_{ex}$ and hence is not a model of $\mathcal{T}_{ex}$:

(i) m and h are Persons attending a Course, $p4$, but they are not in the extension of Student in $\mathcal{I}$, thereby violating axiom $\mathcal{T}_{ex}.6$.
(ii) We have two elements, $c6$ and $p4$, that are both in the extension of Person and of Course, thereby violating axiom $\mathcal{T}_{ex}.1$.

We can, however, easily construct a model $\mathcal{I}'$ of $\mathcal{K}_{ex}$ as follows:

$$\begin{array}{rcl}
\Delta^{\mathcal{I}'} & = & \{h, m, b, c6, p4, c5\}, \\
\mathsf{Mary}^{\mathcal{I}'} & = & m, \\
\mathsf{Betty}^{\mathcal{I}'} & = & b, \\
\mathsf{Hugo}^{\mathcal{I}'} & = & h, \\
\mathsf{CS600}^{\mathcal{I}'} & = & c6, \\
\mathsf{Ph456}^{\mathcal{I}'} & = & p4, \\
\mathsf{Person}^{\mathcal{I}'} & = & \{h, m, b\}, \\
\mathsf{Teacher}^{\mathcal{I}'} & = & \{h, m, b\}, \\
\mathsf{Course}^{\mathcal{I}'} & = & \{c6, p4, c5\}, \\
\mathsf{PGC}^{\mathcal{I}'} & = & \{p4\}, \\
\mathsf{UGC}^{\mathcal{I}'} & = & \{c6\}, \\
\mathsf{Student}^{\mathcal{I}'} & = & \{h, m, b\}, \\
\textsf{\textit{teaches}}^{\mathcal{I}'} & = & \{(m, c6), (h, p4), (b, c5)\}, \\
\textsf{\textit{attends}}^{\mathcal{I}'} & = & \{(h, p4), (m, p4), (b, p4)\}.
\end{array}$$

An important difference relative to databases and other similar formalisms can be illustrated using this example. In $\mathcal{A}_{ex}$, we have stated that Betty is a Teacher, and we know from axiom $\mathcal{T}_{ex}.4$ in Figure 2.3 that Betty must therefore teach at least one course, but we have not said which course she teaches; i.e., there is no role assertion of the form $(\mathsf{Betty}, ?) : \textsf{\textit{teaches}}$ in $\mathcal{A}_{ex}$. In a database setting, an *integrity constraint*

$$
\begin{array}{rll}
\mathcal{A}_{ex} = \{\mathsf{Mary} : & \mathsf{Person}, & (\mathcal{A}_{ex}.1)\\
\mathsf{CS600} : & \mathsf{Course}, & (\mathcal{A}_{ex}.2)\\
\mathsf{Ph456} : & \mathsf{Course} \sqcap \mathsf{PGC}, & (\mathcal{A}_{ex}.3)\\
\mathsf{Hugo} : & \mathsf{Person}, & (\mathcal{A}_{ex}.4)\\
\mathsf{Betty} : & \mathsf{Person} \sqcap \mathsf{Teacher}, & (\mathcal{A}_{ex}.5)\\
(\mathsf{Mary}, \mathsf{CS600}) : & \mathit{teaches}, & (\mathcal{A}_{ex}.6)\\
(\mathsf{Hugo}, \mathsf{Ph456}) : & \mathit{teaches}, & (\mathcal{A}_{ex}.7)\\
(\mathsf{Betty}, \mathsf{Ph456}) : & \mathit{attends}, & (\mathcal{A}_{ex}.8)\\
(\mathsf{Mary}, \mathsf{Ph456}) : & \mathit{attends}\ \} & (\mathcal{A}_{ex}.9)
\end{array}
$$

Fig. 2.4. The example ABox $\mathcal{A}_{ex}$.

can be used to make an apparently similar statement (i.e., that teachers must teach at least one course), but such a constraint would make it mandatory to explicitly specify at least one course that Betty teaches, and failure to do so would be treated as a violation of the integrity constraint (an error). In contrast, in our DL setting it is perfectly fine for a knowledge base to contain such *incomplete information* – we know that Betty stands for an element that is *teaches*-related to some Course, but we do not know to which element; i.e., $\mathcal{K}_{ex}$ has other models in which Betty teaches different courses.

Similarly, in $\mathcal{I}'$, we have that Hugo *attends* Ph456 thanks to $(h, p4) \in \mathit{attends}^{\mathcal{I}'}$, yet this is not enforced by $\mathcal{K}_{ex}$. Due to this interpretation of *attends*, however, it is crucial that $h \in \mathsf{Student}^{\mathcal{I}'}$ (which it is) since, otherwise, $\mathcal{I}'$ would not satisfy axiom $\mathcal{T}_{ex}.6$ in Figure 2.3. So, in $\mathcal{I}'$, Hugo is in the extension of Student, although this is not enforced by $\mathcal{K}_{ex}$; i.e., $\mathcal{K}_{ex}$ has other models in which Hugo is not in the extension of Student. In contrast, in a database setting interpretations can only model those facts that explicitly occur in the database.

Furthermore, assume that we add the following axiom to $\mathcal{T}_{ex}$:

$$\mathsf{PG\text{-}Student} \equiv \mathsf{Student} \sqcap \forall \mathit{attends}.\mathsf{PGC}.$$

Since $\mathcal{A}_{ex}$ explicitly asserts that Betty *attends* Ph456, which is a PGC, and this is the only course that she attends, we might assume that, in each model of our extended $\mathcal{K}_{ex}$, Betty is interpreted as an element in the extension of PG-Student. However, this is not the case: nothing in $\mathcal{K}_{ex}$ rules out the possibility that Betty might attend other courses, and we could construct a model $\mathcal{I}''$ of $\mathcal{K}_{ex}$ that extends $\mathcal{I}'$ by setting $\mathit{teaches}^{\mathcal{I}''} = \mathit{teaches}^{\mathcal{I}'} \cup \{(b, c6)\}$. In $\mathcal{I}''$ $c6$ is not in the extension of PGC, and so Betty is not in the extension of PG-Student. Thus Betty is not interpreted as a PG-Student in every model of $\mathcal{K}_{ex}$; i.e., $\mathcal{K}_{ex}$ does *not* entail Betty : PG-student. In the general AI literature, this important principle is referred to as the *open world assumption*, and we will come back to it later.

2.2.3 Restricted TBoxes and concept definitions

In Section 2.1, we introduced $C \equiv D$ as an abbreviation for $C \sqsubseteq D,\ D \sqsubseteq C$, and used it in our TBox $\mathcal{T}$ to define the meaning of Teacher and Student:

$$\begin{array}{lcl} \mathsf{Teacher} & \equiv & \mathsf{Person} \sqcap \exists \mathit{teaches}.\mathsf{Course}, \\ \mathsf{Student} & \equiv & \mathsf{Person} \sqcap \exists \mathit{attends}.\mathsf{Course}. \end{array}$$

For A a concept name, we call an axiom of the form $A \equiv C$ a *concept definition* of A, and an axiom of the form $A \sqsubseteq C$ a *primitive* concept definition of A. Before we discuss these in detail, let us first convince ourselves that we can restrict our attention to (non-primitive) concept definitions, as formalised in the following lemma.

Lemma 2.8. *Let $A \sqsubseteq C$ be a primitive concept definition in which A_C does not occur. Every model of $A \sqsubseteq C$ can be extended to a model of $A \equiv A_C \sqcap C$ and, vice versa, any model of $A \equiv A_C \sqcap C$ is a model of $A \sqsubseteq C$.*

As a consequence of Lemma 2.8, we can faithfully transform primitive concept definitions into non-primitive ones, and therefore restrict our attention to the latter.

Proof of Lemma 2.8. Let $\mathcal{I}$ be a model of $A \sqsubseteq C$, i.e., $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$. Since A_C does not occur in C, we are free to extend $\mathcal{I}$ by setting $A_C^{\mathcal{I}} = A^{\mathcal{I}}$, thereby obtaining an extended interpretation $\mathcal{I}$ with $A^{\mathcal{I}} = A_C^{\mathcal{I}} \cap C^{\mathcal{I}}$, i.e., a model of $A \equiv A_C \sqcap C$.

Vice versa, consider a model $\mathcal{I}$ of $A \equiv A_C \sqcap C$. Since $(A_C \sqcap C)^{\mathcal{I}} \subseteq C^{\mathcal{I}}$, we have that $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$, and thus $\mathcal{I}$ is also a model of $A \sqsubseteq C$. □

Now consider the concept definition

$$\mathsf{Happy} \equiv \mathsf{Person} \sqcap \forall \mathsf{likes}.\mathsf{Happy}. \tag{2.3}$$

First, observe that this concept definition is *cyclic*: the definition of Happy involves the concept Happy on its right-hand side. Next, we consider an interpretation $\mathcal{I}$ with $\{(p,m),(m,p)\} = \mathsf{likes}^{\mathcal{I}}$ and $\{p,m\} = \mathsf{Person}^{\mathcal{I}}$, and ask ourselves whether p is Happy in $\mathcal{I}$. Since Happy is a defined concept, we might expect that we can determine this by simply considering the interpretation of other concepts and roles. This is, however, not the case: we can choose either $\mathsf{Happy}^{\mathcal{I}} = \{p, m\}$ or $\mathsf{Happy}^{\mathcal{I}} = \emptyset$, and both choices would make $\mathcal{I}$ a model of the concept definition (2.3).

To give TBoxes more *definitorial power*, we can restrict them so as to avoid cyclic references as in the example above.

Definition 2.9. An $\mathcal{ALC}$ *concept definition* is an expression of the form $A \equiv C$ for A a concept name and C a possibly compound $\mathcal{ALC}$ concept.

Let $\mathcal{T}$ be a finite set of concept definitions. We say that A *directly uses* B if there is a concept definition $A \equiv C \in \mathcal{T}$ such that B occurs in C. We say that A *uses* B if A directly uses B, or if there is a concept name B' such that A *uses* B' and B' directly uses B; i.e., *uses* is the transitive closure of *directly uses*.

We call a finite set $\mathcal{T}$ of concept definitions an *acyclic TBox* if

- there is no concept name in $\mathcal{T}$ that uses itself, and
- no concept name occurs more than once on the left-hand side of a concept definition in $\mathcal{T}$.

If $\mathcal{T}$ is an acyclic TBox with $A \equiv C \in \mathcal{T}$, we say that A is *exactly defined in* $\mathcal{T}$, and call C the *definition of* A in $\mathcal{T}$.

In an acyclic TBox we cannot, by definition, have a situation such as follows:

$$\begin{array}{rcl} A_1 & \equiv & \ldots A_2 \ldots \\ A_2 & \equiv & \ldots A_3 \ldots \\ \vdots & & \vdots \\ A_n & \equiv & \ldots A_1 \ldots \end{array}$$

Since acyclic TBoxes are a syntactic restriction of TBoxes, we do not need to define their semantics since it follows directly from the semantics for (general) TBoxes.

To see how an acyclic TBox $\mathcal{T}$ does *not* restrict the interpretation of the concepts that are not defined in $\mathcal{T}$, we make the following observation.

Lemma 2.10. *Let $\mathcal{T}$ be an acyclic TBox, and $\mathcal{I}$ be an interpretation. Then there exists a model $\mathcal{J}$ of $\mathcal{T}$ that coincides with $\mathcal{I}$ on the interpretation of all role and concept names that are* not *defined in $\mathcal{T}$.*

In other words, any interpretation of terms that are not defined in $\mathcal{T}$ can be extended to a model of $\mathcal{T}$ by interpreting defined concept names in a suitable way.

Proof of Lemma 2.10. Let $\mathcal{T}$ be an acyclic TBox, $\mathcal{I}$ an interpretation and $\{A_1, \ldots, A_k\}$ the set of concept names that are not defined in $\mathcal{T}$. By definition, $\mathcal{T}$ is of the form $A_1 \equiv C_1, \ldots, A_k \equiv C_k$. Without loss of generality and because $\mathcal{T}$ is acyclic, we can assume that the indices $\cdot_i$ are such that, if A_i directly uses A_j, then $j < i$. We define the following series of interpretations $\mathcal{I}_i$ as modifications of $\mathcal{I}$:

- for each i, we set

  $\Delta^{\mathcal{I}_i} = \Delta^{\mathcal{I}}$,
  $r^{\mathcal{I}_i} = r^{\mathcal{I}}$ for all role names in $\mathcal{T}$, and
  $A^{\mathcal{I}_i} = A^{\mathcal{I}}$ for all concept names not defined in $\mathcal{T}$, and

- we fix the interpretation of defined concepts as follows:

  $A_1^{\mathcal{I}_1} = C_1^{\mathcal{I}}$, $A_j^{\mathcal{I}_1} = \emptyset$ for all $j > 1$,
  $A_1^{\mathcal{I}_2} = A_1^{\mathcal{I}_1}$, $A_2^{\mathcal{I}_2} = C_2^{\mathcal{I}_1}$, $A_j^{\mathcal{I}_2} = \emptyset$ for all $j > 2$,
  ...
  $A_1^{\mathcal{I}_k} = A_1^{\mathcal{I}_{k-1}}$, $A_2^{\mathcal{I}_k} = A_2^{\mathcal{I}_{k-1}}$, ..., $A_k^{\mathcal{I}_k} = C_k^{\mathcal{I}_{k-1}}$.

By our assumption on the naming of concept names, A_1 uses no defined concept name, and each concept name A_i uses only concept names A_j with $j < i$. Hence the interpretation $\mathcal{I}_k$ is well defined. By definition, $\mathcal{I}_k$ coincides with $\mathcal{I}$ on the interpretation of all role names and concept names that are not defined in $\mathcal{T}$. Moreover, $\mathcal{I}_k$ is a model of $\mathcal{T}$ since it satisfies each axiom in $\mathcal{T}$. $\square$

Next, we will discuss how to *expand* or *unfold* acyclic TBoxes by treating concept definitions like macros. In a nutshell, assume we are given a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ where $\mathcal{T}$ is acyclic, and that we obtain $\mathcal{A}'$ from $\mathcal{A}$ by recursively replacing all occurrences of concept names in $\mathcal{A}$ with their definitions from $\mathcal{T}$, then we can show that $(\mathcal{T}, \mathcal{A})$ and $\mathcal{A}'$ carry the same meaning in the sense that they have (essentially) the same models.

Definition 2.11. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ALC}$ knowledge base, where $\mathcal{T}$ is acyclic and of the form $\mathcal{T} = \{A_i \equiv C_i \mid 1 \leq i \leq m\}$. Let $\mathcal{A}_0 = \mathcal{A}$ and let $\mathcal{A}_{j+1}$ be the result of carrying out the following replacement:

(i) find some $a : D \in \mathcal{A}_j$ in which some A_i occurs in D, for some $1 \leq i \leq m$;
(ii) replace all occurrence of A_i in D with C_i.

If no more replacements can be applied to $\mathcal{A}_k$, we call $\mathcal{A}_k$ the *result of unfolding $\mathcal{T}$ into $\mathcal{A}$.*

Please note that, if $\mathcal{A}_k$ is the result of unfolding $\mathcal{T}$ into $\mathcal{A}$ and $A \equiv C \in \mathcal{T}$, then A does not occur in the right-hand side of any assertions in $\mathcal{A}_k$ (otherwise we could apply the replacement from Definition 2.11 to produce $\mathcal{A}_{k+1}$). Next, we show that the meaning of $(\mathcal{T}, \mathcal{A})$ is the same as the meaning of $\mathcal{A}_k$.

Lemma 2.12. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ALC}$ knowledge base with $\mathcal{T}$ being acyclic. Then the result of unfolding $\mathcal{T}$ into $\mathcal{A}$ exists and, for $\mathcal{A}'$ the result of unfolding $\mathcal{T}$ into $\mathcal{A}$, we have that*

(i) *each model of $\mathcal{K}$ is a model of $\mathcal{A}'$, and*
(ii) *each model $\mathcal{I}$ of $\mathcal{A}'$ can be modified to one of $\mathcal{K}$ that coincides with $\mathcal{I}$ on the interpretation of roles and concepts that are not defined in $\mathcal{T}$.*

Proof. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, $\mathcal{A}_0 = \mathcal{A}$, and $\mathcal{A}_j$ be as described in Definition 2.11. To prove that unfolding indeed terminates, consider the graph $G(\mathcal{A}_j)$ where

- for each concept name in $\mathcal{T}$ and each individual name in $\mathcal{A}_j$, there is a node in $G(\mathcal{A}_j)$,
- there is an edge from A to B if A directly uses B in $\mathcal{T}$, and
- there is an edge from a to A if there is a concept assertion $a : C \in \mathcal{A}_j$ such that A occurs in C.

Since $\mathcal{T}$ is acyclic, the graph $G(\mathcal{A}_0)$ is acyclic, and the replacement rule does not introduce cycles into $G(\mathcal{A}_j)$. Moreover, by Definition 2.11, the edges between concept names do not change from $\mathcal{A}_j$ to $\mathcal{A}_{j+1}$, and the set of nodes remains stable as well. Most importantly, the replacement rule in Definition 2.11 strictly shortens the length of at least one path from an individual name in $\mathcal{A}$ to a leaf node B, and does not lengthen any path. As a consequence, the replacement rule will eventually no longer be applicable, unfolding will therefore terminate, and the result of unfolding $\mathcal{T}$ into $\mathcal{A}$ exists.

For (i), we show by induction on j that $\mathcal{I}$ being a model of $(\mathcal{T}, \mathcal{A}_j)$ implies that $\mathcal{I}$ is a model of $(\mathcal{T}, \mathcal{A}_{j+1})$. Let $\mathcal{I}$ be a model of $(\mathcal{T}, \mathcal{A}_j)$, and let $\mathcal{A}_{j+1}$ be the result of replacing all occurrences of A_i with C_i in an assertion $a : D$ in $\mathcal{A}_j$. Then $\mathcal{I}$ being a model of $\mathcal{T}$ and $\mathcal{A}_i \equiv C_i \in \mathcal{T}$ implies that $A_i^{\mathcal{I}} = C_i^{\mathcal{I}}$, and thus $(D')^{\mathcal{I}} = D^{\mathcal{I}}$ for D' the result of this replacement. Hence $\mathcal{I}$ satisfies $a : D'$ and thus is a model of $(\mathcal{T}, \mathcal{A}_{j+1})$.

For (ii), let $\mathcal{I}$ be a model of $\mathcal{A}'$. As in the proof of Lemma 2.10, we assume that the concept name indices are such that, if A_i directly uses

A_j, then $j < i$. If $\mathcal{I}$ is not a model of $\mathcal{T}$, then modify $\mathcal{I}$ in the following way, starting from $i = 0$ and considering A_i in ascending order:

$$\text{if } \mathcal{I} \text{ does not satisfy } A_i \equiv C_i \in \mathcal{T}, \text{ then set } A_i^{\mathcal{I}} \text{ to } C_i^{\mathcal{I}}.$$

Call the result of this modification $\mathcal{J}$. First, $\mathcal{J}$ is well defined: the order in which we modify $\mathcal{I}$ ensures that the interpretation of a defined concept name A_i only depends on concept names already considered, and the fact that each concept name occurs at most once on the left-hand side of an axiom in $\mathcal{T}$ ensures that $\mathcal{J}$ is well defined. Secondly, $\mathcal{J}$ coincides with $\mathcal{I}$ on the interpretation of roles and concepts not defined in $\mathcal{T}$. Third, by construction, $\mathcal{J}$ is a model of $\mathcal{T}$. Finally, $\mathcal{J}$ is a model of $\mathcal{A}$: $\mathcal{J}$ satisfies

- each role assertion in $\mathcal{A}$ because $\mathcal{I}$ and $\mathcal{J}$ coincide on the interpretation of individual and role names, and $\mathcal{I}$ is a model of $\mathcal{A}$;
- each concept assertion in $\mathcal{A}$: let $a : C \in \mathcal{A}$. Then there is some $a : C' \in \mathcal{A}'$ where C' is the result of replacing concept names defined in $\mathcal{T}$ with their definition. Since there are no defined concept names occurring in C', the construction of $\mathcal{J}$ and $\mathcal{J}$ being a model of $\mathcal{T}$ implies that $(C')^{\mathcal{J}} = C^{\mathcal{I}}$, and thus $\mathcal{J}$ satisfies $a : C$. □

Hence we have shown that acyclic TBox definitions are like macros that can be expanded directly into an ABox. It should be noted, however, that unfolding of acyclic definitions may cause an exponential blow-up of the size of the ABox, as demonstrated by the following example.

Example 2.13. Consider the ABox $\mathcal{A} = \{A : a\}$ together with the acyclic TBox $\mathcal{T}$ consisting of the following definitions:

$$\begin{array}{rcl} A_0 & \equiv & \forall r.A_1 \sqcap \forall s.A_1, \\ A_1 & \equiv & \forall r.A_2 \sqcap \forall s.A_2, \\ & \vdots & \\ A_{n-1} & \equiv & \forall r.A_n \sqcap \forall s.A_n. \end{array}$$

The knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ has a size that is linear in n, but the ABox obtained by unfolding $\mathcal{T}$ into $\mathcal{A}$ contains the concept name A_n 2^n times.

We will see in Section 4.2.2 an improved *lazy* way to unfold an acyclic TBox, and discuss how this avoids the exponential blow-up that the *eager* unfolding introduced above and used in the example may cause.

2.3 Basic reasoning problems and services

So far, we have defined the components of a DL knowledge base and what it means for an interpretation to be a model of such a knowledge base. Next, we define the reasoning problems commonly considered in DLs, and discuss their relationships. We start by defining the basic reasoning problems in DLs upon which the basic system services of a DL *reasoner* are built, and then provide a number of examples.

Definition 2.14. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ALC}$ knowledge base, C, D possibly compound $\mathcal{ALC}$ concepts, and b an individual name. We say that

(i) C is *satisfiable* with respect to $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ and some $d \in \Delta^{\mathcal{I}}$ with $d \in C^{\mathcal{I}}$;
(ii) C is *subsumed by* D with respect to $\mathcal{T}$, written $\mathcal{T} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$;
(iii) C and D are *equivalent* with respect to $\mathcal{T}$, written $\mathcal{T} \models C \equiv D$, if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$;
(iv) $\mathcal{K}$ is *consistent* if there exists a model of $\mathcal{K}$;
(v) b is an *instance of* C with respect to $\mathcal{K}$, written $\mathcal{K} \models b : C$, if $b^{\mathcal{I}} \in C^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{K}$.

We use the standard entailment symbol $\models$ because the semantics of DL entailment coincides with the semantics of entailment in first-order logic (see Section 2.6.1). To underline the fact that, once $\mathcal{T}$ is fixed, subsumption and equivalence with respect to $\mathcal{T}$ is a binary relation between (possibly compound) concepts, we often use $C \sqsubseteq_{\mathcal{T}} D$ for $\mathcal{T} \models C \sqsubseteq D$ and $C \equiv_{\mathcal{T}} D$ for $\mathcal{T} \models C \equiv D$.

Please note that satisfiability and subsumption are defined with respect to a TBox, whereas consistency and instance are defined with respect to a TBox and an ABox. We can always assume that the TBox (or the TBox and ABox) are empty: in this case, "all models of $\mathcal{T}$ (or $\mathcal{K}$)" becomes simply "all interpretations". We will sometimes talk about the consistency of a TBox $\mathcal{T}$ or an ABox $\mathcal{A}$, which is equivalent to the consistency of $\mathcal{K} = (\mathcal{T}, \emptyset)$ and $\mathcal{K} = (\emptyset, \mathcal{A})$ respectively.

Please make sure you understand the difference between an element being *in the extension of* a concept C in an interpretation $\mathcal{I}$, and an individual name being *an instance of* a concept C: an individual name b can be interpreted in many different ways, and $b^{\mathcal{I}_1}$ can have quite different properties from $b^{\mathcal{I}_2}$. A knowledge base $\mathcal{K}$ can, however, enforce that $b^{\mathcal{I}}$ is in the extension of C in *every* model $\mathcal{I}$ of $\mathcal{K}$, which is why we define the notion of an instance for individual names. For example, consider our TBox $\mathcal{T}_{ex}$ and ABox $\mathcal{A}_{ex}$ from Figures 2.3 and 2.4, and our example model $\mathcal{I}'$ of $\mathcal{K}_{ex} = (\mathcal{T}_{ex}, \mathcal{A}_{ex})$. In $\mathcal{I}'$, h is in the extension of $\mathsf{Teacher}$, but there can be other interpretations $\mathcal{I}'$ where $h \notin \mathsf{Teacher}^{\mathcal{I}'}$ and even where $h \notin \Delta^{\mathcal{I}}$. However, in every model $\mathcal{I}$ of $\mathcal{K}$, the element of $\Delta^{\mathcal{I}}$ that interprets Hugo (i.e., $\mathsf{Hugo}^{\mathcal{I}}$) *must* be in the extension of $\mathsf{Teacher}$; i.e., $\mathcal{K} \models \mathsf{Hugo} : \mathsf{Teacher}$.

So far, most of the concepts we have seen were satisfiable, but we have also seen concepts such as $A \sqcap \neg A$ that are unsatisfiable even with respect to the empty TBox, i.e., we cannot find any interpretation $\mathcal{I}$ in which $(A \sqcap \neg A)^{\mathcal{I}} \neq \emptyset$, because this would mean that we have some element in both the extension of A and of $\neg A$. Thus $\bot$ and $A \sqcap \neg A$ are equivalent (with respect to the empty TBox). In fact there are (infinitely) many such concepts; for example, $\exists r.A \sqcap \forall r.\neg A$ is also unsatisfiable, because any element in the extension of this concept would need to have an r-filler that is in the extension of both A and $\neg A$. More interesting are concepts that are satisfiable with respect to some but not all TBoxes. For example, consider again the TBox $\mathcal{T}_{ex}$; $\mathsf{Course} \sqcap \exists \mathit{teaches}.\mathsf{Course}$ is *not* satisfiable with respect to $\mathcal{T}_{ex}$ because axioms $\mathcal{T}_{ex}.1$ and $\mathcal{T}_{ex}.5$ prevent an element in the extension of Course from having a $\mathit{teaches}$-filler.

Similarly, (infinitely) many subsumption relations are entailed even by the empty TBox; for example, it is easy to see that $\emptyset \models A \sqcap B \sqsubseteq A$, $\emptyset \models A \sqsubseteq A \sqcup B$, and $\emptyset \models \exists r.A \sqcap B \sqsubseteq \exists r.A$. Slightly more tricky is $\emptyset \models \exists r.A \sqcap \forall r.B \sqsubseteq \exists r.B$: every element x in the extension of $\exists r.A \sqcap \forall r.B$ has an r-filler in A, and the second conjunct implies that this r-filler also needs to be in the extension of B; hence x is also in the extension of $\exists r.B$. If we again consider $\mathcal{T}_{ex}$, we have that $\mathcal{T}_{ex} \models \mathsf{PGC} \sqsubseteq \neg\mathsf{Person}$, and that $\mathcal{T}_{ex} \models \exists \mathit{teaches}.\mathsf{Course} \sqsubseteq \neg\mathsf{Course}$. To see the latter, try to find a model $\mathcal{I}$ of $\mathcal{T}_{ex}$ with $x \in (\exists \mathit{teaches}.\mathsf{Course})^{\mathcal{I}}$: since x has a $\mathit{teaches}$-filler, x must be in $\mathsf{Person}^{\mathcal{I}}$ and, if x were in $\mathsf{Course}^{\mathcal{I}}$, then x would need to be in $(\neg\mathsf{Person})^{\mathcal{I}}$ – thereby contradicting $x \in \mathsf{Person}^{\mathcal{I}}$. We will formalise this in Theorem 2.17 (ii).

As we have already seen, the knowledge base $\mathcal{K}_{ex} = (\mathcal{T}_{ex}, \mathcal{A}_{ex})$ is consistent since we have built a model $\mathcal{I}'$ of it. In contrast, the knowledge base $(\mathcal{T}_{ex}, \mathcal{A}_2)$, with $\mathcal{A}_2$ defined as follows, is not consistent:

$$\mathcal{A}_2 = \{\mathsf{ET} : \mathsf{Course},\ (\mathsf{ET}, \mathsf{Foo}) : \mathit{teaches}\}.$$

If we try to build a model $\mathcal{I}$ of $(\mathcal{T}_{ex}, \mathcal{A}_2)$, we will fail because $\mathsf{ET}^{\mathcal{I}}$ would need to be in $\mathsf{Course}^{\mathcal{I}}$, therefore *not* in $\mathsf{Person}^{\mathcal{I}}$ due to the axiom $\mathcal{T}_{ex}.1$, yet *in* $\mathsf{Person}^{\mathcal{I}}$ because $\mathsf{ET}^{\mathcal{I}}$ has a *teaches*-filler. Removing either of the two assertions from $\mathcal{A}_2$ results in an ABox that is consistent with $\mathcal{T}_{ex}$.

Next, we would like to point out that it is possible for a knowledge base $(\mathcal{T}, \mathcal{A})$ to be consistent and for a concept C to be unsatisfiable with respect to $\mathcal{T}$: clearly, $\bot$ is unsatisfiable with respect to every TBox since, by Definition 2.2, $\bot^{\mathcal{I}} = \emptyset$ for every interpretation $\mathcal{I}$. Even if C is defined in $\mathcal{T}$ (see Definition 2.9), it is possible that C is unsatisfiable with respect to $\mathcal{T}$ while $\mathcal{T}$ is consistent; consider, for example, the TBox $\mathcal{T} = \{A \equiv B \sqcap \neg B\}$ which has infinitely many models, but in all of them the extension of A is empty.

Finally, Mary and Hugo are instances of Teacher with respect to $\mathcal{K}_{ex} = (\mathcal{T}_{ex}, \mathcal{A}_{ex})$, because $\mathcal{A}_{ex}$ contains assertions that they are both Persons and teach some Courses, and because axiom $\mathcal{T}_{ex}$.4 implies that a Person who *teaches* a Course is a Teacher. Hence, in every model $\mathcal{I}$ of $\mathcal{K}_{ex}$, $\mathsf{Mary}^{\mathcal{I}} \in \mathsf{Teacher}^{\mathcal{I}}$ and $\mathsf{Hugo}^{\mathcal{I}} \in \mathsf{Teacher}^{\mathcal{I}}$.

To deepen the readers' understanding of the reasoning problems, we discuss some important properties of the subsumption relationship.

Lemma 2.15. *Let C, D and E be concepts, b an individual name, and $(\mathcal{T}, \mathcal{A})$, $(\mathcal{T}', \mathcal{A}')$ knowledge bases with $\mathcal{T} \subseteq \mathcal{T}'$ and $\mathcal{A} \subseteq \mathcal{A}'$.*

(i) *$C \sqsubseteq_{\mathcal{T}} C$.*
(ii) *If $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} E$, then $C \sqsubseteq_{\mathcal{T}} E$.*
(iii) *If b is an instance of C with respect to $(\mathcal{T}, \mathcal{A})$ and $C \sqsubseteq_{\mathcal{T}} D$, then b is an instance of D with respect to $(\mathcal{T}, \mathcal{A})$.*
(iv) *If $\mathcal{T} \models C \sqsubseteq D$ then $\mathcal{T}' \models C \sqsubseteq D$.*
(v) *If $\mathcal{T} \models C \equiv D$ then $\mathcal{T}' \models C \equiv D$.*
(vi) *If $(\mathcal{T}, \mathcal{A}) \models b : E$ then $(\mathcal{T}', \mathcal{A}') \models b : E$.*

Part (ii) of Lemma 2.15 says that the subsumption relationship is transitive, and parts (iv)–(vi) say that $\mathcal{ALC}$ is *monotonic*: the more statements a knowledge base contains, the more entailments it has.

Proof. Let C, D, E, b and $(\mathcal{T}, \mathcal{A})$ be as described in Lemma 2.15.

(i) For any interpretation $\mathcal{I}$ and any concept C, we obviously have $C^{\mathcal{I}} = C^{\mathcal{I}}$, and thus $C^{\mathcal{I}} \subseteq C^{\mathcal{I}}$. Hence we have $C \sqsubseteq_{\mathcal{T}} C$.
(ii) Let $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} E$ and consider a model $\mathcal{I}$ of $\mathcal{T}$: we have that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and $D^{\mathcal{I}} \subseteq E^{\mathcal{I}}$. Hence we have, by transitivity of $\subseteq$, $C^{\mathcal{I}} \subseteq E^{\mathcal{I}}$. Since $\mathcal{I}$ was an arbitrary model of $\mathcal{T}$, this implies $C \sqsubseteq_{\mathcal{T}} E$.

(iii) Let b be an instance of C with respect to $(\mathcal{T}, \mathcal{A})$ and $C \sqsubseteq_{\mathcal{T}} D$. Hence, for each model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$, we have that $b^{\mathcal{I}} \in C^{\mathcal{I}}$ and $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Thus, for each model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$, we have that $b^{\mathcal{I}} \in D^{\mathcal{I}}$, and thus b is an instance of D with respect to $(\mathcal{T}, \mathcal{A})$.

(iv) This is an immediate consequence of the fact that $\mathcal{T} \subseteq \mathcal{T}'$ and Lemma 2.5.

(v) and (vi) can be proven analogously to Lemma 2.5 and are left to the reader. □

Now we reconsider our observation about the generosity of the set of operators to build $\mathcal{ALC}$ concept descriptions and take Lemma 2.3 a bit further.

Lemma 2.16. *Let C and D be concepts, r a role, $\mathcal{T}_0 = \emptyset$ the empty TBox, and $\mathcal{T}$ an arbitrary TBox.*

(i) $\mathcal{T}_0 \models \top \equiv (\neg C \sqcup C)$.
(ii) $\mathcal{T}_0 \models \bot \equiv (\neg C \sqcap C)$.
(iii) $\mathcal{T}_0 \models C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$.
(iv) $\mathcal{T}_0 \models \forall r.C \equiv \neg(\exists r.\neg C)$.
(v) $\mathcal{T} \models C \sqsubseteq D$ *if and only if* $\mathcal{T} \models \top \sqsubseteq (\neg C \sqcup D)$.

As a consequence of Lemma 2.16, we can indeed rewrite every concept description into an equivalent one that does not use $\top$, $\bot$, disjunction or universal restrictions. Also, we could formulate an alternative form of this lemma that would allow us to drop conjunction rather than disjunction, and existential rather than universal restrictions. As a further consequence of Lemma 2.15, these equivalences are entailed by all TBoxes – and thus we call them *tautologies*.

Proof of Lemma 2.16. Equivalences (i) and (ii) are an immediate consequence of Lemma 2.3 (i) and (ii) which state that $\top^{\mathcal{I}} = (C \sqcup \neg C)^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = (C \sqcap \neg C)^{\mathcal{I}}$ hold in any interpretation. Hence $\emptyset \models \top \equiv (\neg C \sqcup C)$ and $\emptyset \models \bot \equiv (\neg C \sqcap C)$.

For (iii), Lemma 2.3 (iii) and (v) imply that, for any interpretation $\mathcal{I}$, $(C \sqcup D)^{\mathcal{I}} = (\neg\neg(C \sqcup D))^{\mathcal{I}} = (\neg(\neg C \sqcap \neg D))^{\mathcal{I}}$, and thus $\emptyset \models C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$.

For (iv), Lemma 2.3 (iii) and (vii) imply that, for any interpretation $\mathcal{I}$, $(\forall r.C)^{\mathcal{I}} = (\neg\neg(\forall r.C))^{\mathcal{I}} = (\neg(\exists r.\neg C))^{\mathcal{I}}$, and thus $\emptyset \models \forall r.C \equiv \neg(\exists r.\neg C)$.

For (v), assume that $\mathcal{T} \models C \sqsubseteq D$, and consider a model $\mathcal{I}$ of $\mathcal{T}$ and some $a \in \Delta^{\mathcal{I}}$. Since $\mathcal{I}$ is a model of $\mathcal{T}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. If $a \in C^{\mathcal{I}}$, then

$\mathcal{T} \models C \sqsubseteq D$ implies that $a \in D^\mathcal{I}$, and thus $a \in (\neg C \sqcup D)^\mathcal{I}$. Otherwise, $a \in (\neg C)^\mathcal{I}$ and thus also in $(\neg C \sqcup D)^\mathcal{I}$. Hence $\mathcal{T} \models \top \sqsubseteq (\neg C \sqcup D)$. The other direction is analogous. $\square$

Next, we formalise some of the implicit relationships between DL reasoning problems that we have used intuitively in our considerations above.

Theorem 2.17. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ALC}$ knowledge base, C, D possibly compound $\mathcal{ALC}$ concepts and b an individual name.*

(i) $C \equiv_\mathcal{T} D$ *if and only if* $C \sqsubseteq_\mathcal{T} D$ *and* $D \sqsubseteq_\mathcal{T} C$.
(ii) $C \sqsubseteq_\mathcal{T} D$ *if and only if* $C \sqcap \neg D$ *is not satisfiable with respect to* $\mathcal{T}$.
(iii) C *is satisfiable with respect to* $\mathcal{T}$ *if and only if* $C \not\sqsubseteq_\mathcal{T} \bot$.
(iv) C *is satisfiable with respect to* $\mathcal{T}$ *if and only if* $(\mathcal{T}, \{b : C\})$ *is consistent.*
(v) $(\mathcal{T}, \mathcal{A}) \models b : C$ *if and only if* $(\mathcal{T}, \mathcal{A} \cup \{b : \neg C\})$ *is* not *consistent.*
(vi) *if* $\mathcal{T}$ *is acyclic, and* $\mathcal{A}'$ *is the result of unfolding* $\mathcal{T}$ *into* $\mathcal{A}$*, then* $\mathcal{K}$ *is consistent if and only if* $(\emptyset, \mathcal{A}')$ *is consistent.*

As a consequence of this theorem, we can focus our attention on knowledge base consistency, since all the reasoning problems introduced in Definition 2.14 can be *reduced* to knowledge base consistency; i.e., we can use an algorithm for knowledge base consistency to decide all of these other reasoning problems. Note, however, that there are other reasoning problems not mentioned yet for which such a reduction is not possible, and even if it is possible it may in some cases incur an exponential blow-up in the size of the problem. In particular, conjunctive query answering (see Chapter 7) is 2ExpTime-complete for $\mathcal{ALCI}$, whereas $\mathcal{ALCI}$ knowledge base consistency is “only” ExpTime-complete [Lut08], and for $\mathcal{SROIQ}$, the decidability of conjunctive query answering is still open, whereas knowledge base consistency is known to be decidable and N2ExpTime-complete [GLHS08, Kaz08].

Next, we will prove Theorem 2.17.

Proof. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ALC}$ knowledge base, C, D possibly compound $\mathcal{ALC}$ concepts and b an individual name.

(i) Let $C \equiv_\mathcal{T} D$. By definition, this means that $C^\mathcal{I} = D^\mathcal{I}$, for each model $\mathcal{I}$ of $\mathcal{T}$. This implies that, for each model $\mathcal{I}$ of $\mathcal{T}$, we have $C^\mathcal{I} \subseteq D^\mathcal{I}$ and $D^\mathcal{I} \subseteq C^\mathcal{I}$. Hence we have, by definition, that $C \sqsubseteq_\mathcal{T} D$ and $D \sqsubseteq_\mathcal{T} C$.

Now let $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$. We can use an analogous way of reasoning to conclude that $C \equiv_{\mathcal{T}} D$.[4]

(ii) Let $C \sqsubseteq_{\mathcal{T}} D$. By definition, this means that, in every model $\mathcal{I}$ of $\mathcal{T}$, we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Hence there cannot be a model $\mathcal{I}$ of $\mathcal{T}$ in which there is some $x \in C^{\mathcal{I}}$ with $x \notin D^{\mathcal{I}}$. This means that there cannot be a model $\mathcal{I}$ of $\mathcal{T}$ in which there is some $x \in C^{\mathcal{I}}$ with $x \in (\neg D)^{\mathcal{I}}$, and thus $C \sqcap \neg D$ is not satisfiable with respect to $\mathcal{T}$.

For the other direction, let $C \sqcap \neg D$ be unsatisfiable with respect to $\mathcal{T}$. Hence in every model $\mathcal{I}$ of $\mathcal{T}$, we have that $(C \sqcap \neg D)^{\mathcal{I}} = \emptyset$, and thus $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds in every model $\mathcal{I}$ of $\mathcal{T}$.

(iii) First, remember that, by Definition 2.2, $\perp^{\mathcal{I}} = \emptyset$ in every interpretation $\mathcal{I}$. Now let C be satisfiable with respect to $\mathcal{T}$. Hence there is some model $\mathcal{I}$ of $\mathcal{T}$ with $C^{\mathcal{I}} \neq \emptyset$, and thus $C^{\mathcal{I}} \not\subseteq \perp^{\mathcal{I}}$.

Similarly, if $C \sqsubseteq_{\mathcal{T}} \perp$, then $C^{\mathcal{I}} = \emptyset$ in every model $\mathcal{I}$ of $\mathcal{T}$, and thus C is not satisfiable with respect to $\mathcal{T}$.

(iv) Let C be satisfiable with respect to $\mathcal{T}$. Hence there exists some $\mathcal{I}$ with $C^{\mathcal{I}} \neq \emptyset$. Take some $x \in C^{\mathcal{I}}$ and extend $\mathcal{I}$ by setting $b^{\mathcal{I}} = x$. This clearly preserves $\mathcal{I}$ being a model of $\mathcal{T}$, and also makes $\mathcal{I}$ a model of the ABox $\{b : C\}$. Hence $(\mathcal{T}, \{b : C\})$ is consistent.

If $(\mathcal{T}, \{b : C\})$ is consistent, then it has some model, say $\mathcal{I}$. By definition, $b^{\mathcal{I}} \in C^{\mathcal{I}}$, and thus $C^{\mathcal{I}} \neq \emptyset$.

(v) Let b be an instance of C with respect to $\mathcal{K}$. By definition, we have $b^{\mathcal{I}} \in C^{\mathcal{I}}$, for every model $\mathcal{I}$ of $\mathcal{K}$. Together with the fact that $C^{\mathcal{I}}$ and $(\neg C)^{\mathcal{I}}$ are disjoint, this implies that there is no model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$ in which $b^{\mathcal{I}} \in (\neg C)^{\mathcal{I}}$, and thus $(\mathcal{T}, \mathcal{A} \cup \{b : \neg C\})$ is not consistent. Please note that the above line of reasoning is independent of $\mathcal{K}$'s consistency.

Let $(\mathcal{T}, \mathcal{A} \cup \{b : \neg C\})$ be inconsistent. If $(\mathcal{T}, \mathcal{A})$ is also inconsistent, we are done since any model of $(\mathcal{T}, \mathcal{A})$ satisfies everything because there are no such models. Otherwise, there are models of $(\mathcal{T}, \mathcal{A})$, but there cannot be a model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$ with $b^{\mathcal{I}} \in (\neg C)^{\mathcal{I}}$ because this would contradict our assumption. Hence in every model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$, we have $b^{\mathcal{I}} \notin (\neg C)^{\mathcal{I}}$, which is, by Definition 2.2, the same as $b^{\mathcal{I}} \in C^{\mathcal{I}}$. Hence b is an instance of C with respect to $\mathcal{K}$.

(vi) This is an immediate consequence of Lemma 2.12. □

In general, when designing or changing a knowledge base, it is helpful to see the *effects* of the current TBox and ABox statements. We will use

[4] And we cordially invite the reader to verify this.

the reasoning problems from Definition 2.14 to formalise some of these effects and formulate them in terms of *reasoning services*. The following is a list of the most basic DL reasoning services.

(i) Given a TBox $\mathcal{T}$ and a concept C, check whether C is *satisfiable* with respect to $\mathcal{T}$.
(ii) Given a TBox $\mathcal{T}$ and two concepts C and D, check whether C is *subsumed by* D with respect to $\mathcal{T}$.
(iii) Given a TBox $\mathcal{T}$ and two concepts C and D, check whether C and D are *equivalent* with respect to $\mathcal{T}$.
(iv) Given a knowledge base $(\mathcal{T}, \mathcal{A})$, check whether $(\mathcal{T}, \mathcal{A})$ is *consistent*.
(v) Given a knowledge base $(\mathcal{T}, \mathcal{A})$, an individual name a, and a concept C, check whether a is an *instance of* C with respect to $(\mathcal{T}, \mathcal{A})$.

Please note that these basic reasoning services correspond one-to-one to the basic reasoning problems from Definition 2.14. As a consequence, we know exactly *what* each of these reasoning services should do, even though we might not know *how* such a service could be implemented – this will be discussed in Chapter 4. To put it differently, the behaviour of a service has been described independently of a specific algorithm or its implementation, and thus we can expect that, for example, every satisfiability checker for $\mathcal{ALC}$ gives the same answer when asked whether a certain concept is satisfiable with respect to a certain TBox – regardless of how this satisfiability checker works.

Clearly, we might be able to compute these services by hand, yet this is unfeasible for larger knowledge bases, and it has turned out to be quite useful to have implementations of these services. In the past, numerous DLs have been investigated with respect to their *decidability* and *complexity*, i.e., whether or which of the reasoning problems are decidable and, if they are, how complex they are in terms of computation time and space. As we saw in Theorem 2.17, we can reduce all these basic reasoning problems to knowledge base consistency, and thus use an algorithm that decides consistency, for example, as a sub-routine in an algorithm that checks subsumption.

Using these most basic reasoning services, we can specify slightly more sophisticated reasoning services as follows.

- *Classification* of a TBox: given a TBox $\mathcal{T}$, compute the *subsumption hierarchy* of all concept names occurring in $\mathcal{T}$ with respect to $\mathcal{T}$. That is, for each pair A, B of concept names occurring in $\mathcal{T}$, check whether A is subsumed by B with respect to $\mathcal{T}$ and whether B is subsumed by A with respect to $\mathcal{T}$.

- Checking the *satisfiability* of concepts in $\mathcal{T}$: given a TBox $\mathcal{T}$, for each concept name A in $\mathcal{T}$, test whether A is satisfiable with respect to $\mathcal{T}$. If it is not, then this is usually an indication of a modelling error.

- *Instance retrieval*: given a concept C and a knowledge base $\mathcal{K}$, return all those individual names b such that b is an instance of C with respect to $\mathcal{K}$. That is, for each individual name b occurring in $\mathcal{K}$, check whether it is an instance of C with respect to $\mathcal{K}$, and return the set of those individual names for which this test is positive.

- *Realisation* of an individual name: given an individual name b and a knowledge base $\mathcal{K}$, test, for each concept name A occurring in $\mathcal{T}$, whether b is an instance of A with respect to $\mathcal{K}$, and return the set of those concept names for which this test is positive.

The result of classification is usually presented in form of a *subsumption hierarchy*, that is, a graph whose nodes are labelled with concept names from $\mathcal{T}$ and where we find an edge from a node labelled A to a node labelled B if A is subsumed by B with respect to $\mathcal{T}$. We may want to choose a slightly more succinct representation: from Lemma 2.15, we know that the subsumption relationship $\sqsubseteq_{\mathcal{T}}$ is a *pre-order*, i.e., a reflexive and transitive relation. It is common practice to consider the induced *strict partial order* $\sqsubset_{\mathcal{T}}$, i.e., an irreflexive and transitive (and therefore anti-symmetric) relation, by identifying all concepts participating in a cycle $C \sqsubseteq_{\mathcal{T}} \ldots \sqsubseteq_{\mathcal{T}} C$ – or collapsing them all into a single node in our graphical representation. In addition, we might want to show only direct edges; that is, we might not want to draw an edge from a node labelled C to a node labelled E in case there is a node labelled D such that $C \sqsubseteq_{\mathcal{T}} D \sqsubseteq_{\mathcal{T}} E$: this is commonly known as the *Hasse diagram* of a partial order.

In Figure 2.5, we present the subsumption hierarchy for the TBox $\mathcal{T}_{ex}$ from Figure 2.3. Please make sure you understand the difference between this graphical representation of a subsumption hierarchy and the graphical representation of an interpretation such as the one presented in Figure 2.2: both are graphs, but with very different meanings.

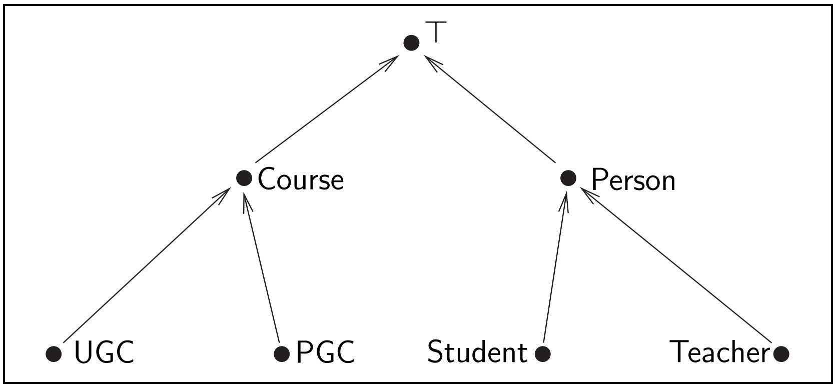

Fig. 2.5. A graphical representation of the subsumption hierarchy for the TBox $\mathcal{T}_{ex}$ from Figure 2.3.

2.4 Using reasoning services

Here, we sketch how DL reasoning services can be used during the construction of a DL knowledge base. Assume we want to design a DL knowledge base about universities, courses, students etc. First, we would need to fix some set of interesting *terms* and decide which of them are concept names and which are role names. Then we could explicate some background knowledge, for example that Courses and Persons are disjoint and that only a Person ever *teaches* somebody or *attends* something; see axioms $\mathcal{T}_{ex}.1$, $\mathcal{T}_{ex}.5$ and $\mathcal{T}_{ex}.7$ in Figure 2.3. Next, we could define some relevant concepts, for example UGC and PGC as kinds of Course, $\mathsf{Teacher}$ as a Person who *teaches* a Course, and $\mathsf{Student}$ as a Person who *attends* a Course; see axioms $\mathcal{T}_{ex}.2$, $\mathcal{T}_{ex}.3$, $\mathcal{T}_{ex}.4$ and $\mathcal{T}_{ex}.6$ in Figure 2.3. Then it might be useful to see the subsumption hierarchy of our TBox $\mathcal{T}_{ex}$. In our example, we can easily compute this hierarchy by hand; see Figure 2.5.

Now assume that we extend $\mathcal{T}_{ex}$ by adding the following concept definition:

$$\mathsf{Professor} \equiv \exists \mathit{teaches}.\mathsf{PGC}.$$

For $\mathcal{T}'_{ex}$ this extended TBox, it is a bit more tricky to see that, in addition to the subsumptions above, we also have $\mathcal{T}'_{ex} \models \mathsf{Professor} \sqsubseteq \mathsf{Person}$. However, this still fits our intuition, and we can continue extending our knowledge base. Assume we extend $\mathcal{T}'_{ex}$ with the following GCI that expresses that a $\mathsf{LazyStudent}$ does not *attend* any Courses:

$$\mathsf{LazyStudent} \sqsubseteq \forall \mathit{attends}.\neg\mathsf{Course}.$$

Let $\mathcal{T}''_{ex}$ be the result of this extension. It is not too hard to see that a $\mathsf{LazyStudent}$ is *not* a $\mathsf{Student}$ (because every $\mathsf{Student}$ $\mathit{attends}$ at least one Course), i.e., $\mathcal{T}''_{ex} \not\models \mathsf{LazyStudent} \sqsubseteq \mathsf{Student}$. This is no longer consistent with our intuition or concept naming scheme. We might try to fix this perceived problem by modifying the newly added GCI, for example, by turning it into the following concept definition:

$$\mathsf{LazyStudent} \equiv \mathsf{Student} \sqcap \forall \mathit{attends}.\neg\mathsf{Course}.$$

This modification now makes make $\mathsf{LazyStudent}$ unsatisfiable with respect to the resulting TBox since axiom $\mathcal{T}_{ex}.6$ states that a $\mathsf{Student}$ necessarily $\mathit{attends}$ some Course. We might consider introducing a new role, $\mathit{activelyAttends}$, and defining lazy students as those who do not actively attend a course; however, the DL $\mathcal{ALC}$ is too weak to capture the interaction between active attendance and attendance, so we will abandon our efforts to model lazy students, and go back to $\mathcal{T}'_{ex}$.

Now assume we add some knowledge about concrete individuals; for example, we add our ABox $\mathcal{A}_{ex}$ from Figure 2.4 to give $\mathcal{K} = (\mathcal{T}'_{ex}, \mathcal{A}_{ex})$. Then it would be quite helpful to learn that Mary and Hugo are instances of $\mathsf{Teacher}$ and that Hugo is an instance of $\mathsf{Professor}$ with respect to $\mathcal{K}$– even though this knowledge is not explicitly stated in our knowledge base, it follows from it, and thus should be made available to the user. For example, if one asks to retrieve all $\mathsf{Teacher}$s in $\mathcal{K}$, then Betty, Mary and Hugo should be returned.

The design of ontology editors that help users to build, maintain and use a DL knowledge base is a very active research area, partly due to the fact that the web ontology language OWL is based on DLs, and DL reasoning services can thus be used to support *ontology engineering* in OWL; we will discuss this in more detail in Chapter 8.

2.5 Extensions of the basic DL $\mathcal{ALC}$

We next motivate and introduce the syntax and semantics for a number of important extensions of the basic DL $\mathcal{ALC}$, namely inverse roles, number restrictions, nominals, role inclusions and transitive roles.

2.5.1 Inverse roles

Consider our running example and assume that we want to add to our TBox $\mathcal{T}_{ex}$ from Figure 2.3 the following GCIs to express that $\mathsf{Professors}$

are $\mathsf{Teacher}$s, and that Courses are not taught by $\mathsf{Professor}$s:

$$\begin{aligned}\mathsf{Professor} &\sqsubseteq \mathsf{Teacher},\\ \mathsf{Course} &\sqsubseteq \forall \textit{taught-by}.\neg\mathsf{Professor}.\end{aligned}$$

Let us call the resulting TBox $\mathcal{T}'_{ex}$. Intuitively, $\mathsf{Professor}$ should be unsatisfiable with respect to $\mathcal{T}'_{ex}$: due to the first GCI above, an element p in the extension of $\mathsf{Professor}$ would also need to be in the extension of $\mathsf{Teacher}$, and hence axiom $\mathcal{T}_{ex}.4$ implies that p has a *teaches*-filler, say c, that is a Course. Now, if p *teaches* c, then c should be $\mathsf{taught\text{-}by}$ p, and thus the second statement above implies that p is a $\neg\mathsf{Professor}$, contradicting our assumption. Now this argumentation contains a serious flaw: *teaches* and *taught-by* are interpreted as some arbitrary binary relations, and thus it is *not* the case that, if p *teaches* c, then c is $\mathsf{taught\text{-}by}$ p. Indeed, $\mathsf{Professor}$ is satisfiable with respect to $\mathcal{T}'_{ex}$: any model $\mathcal{I}$ of $\mathcal{T}_{ex}$ in which $\mathsf{Professor}^{\mathcal{I}} \subseteq \mathsf{Teacher}^{\mathcal{I}}$ holds and $\textit{taught-by}^{\mathcal{I}} = \emptyset$ is a model of $\mathcal{T}'_{ex}$.

In order to relate roles such as *teaches* and *taught-by* in the desired way, DLs can be extended with *inverse roles*. The fact that a DL provides inverse roles is normally indicated by the letter $\mathcal{I}$ in its name. Since we will discuss and name many different DLs (e.g., $\mathcal{ALC}$, $\mathcal{ALCO}$, $\mathcal{ALCOI}$, $\mathcal{SHIQ}$), we will use $\mathcal{L}$ as a placeholder for the name of a DL.

Definition 2.18. For R a role name, R^- is an *inverse role*. The set of $\mathcal{I}$ *roles* is $\mathbf{R} \cup \{R^- \mid R \in \mathbf{R}\}$.

Let $\mathcal{L}$ be a description logic. The set of $\mathcal{LI}$ *concepts* is the smallest set of concepts that contains all $\mathcal{L}$ concepts and where $\mathcal{I}$ roles can occur in all places of role names.

In addition to what is said in Definition 2.1, an interpretation $\mathcal{I}$ maps inverse roles to binary relations as follows:

$$(r^-)^{\mathcal{I}} = \{(y,x) \mid (x,y) \in r^{\mathcal{I}}\}.$$

Following this definition, in the DL $\mathcal{ALCI}$, inverse roles can occur in existential and universal restrictions, for example, in the following concept:

$$\exists r^-.(\forall s.(\exists r.A \sqcap \forall s^-.B)).$$

In $\mathcal{ALCI}$, we now have indeed that $(x,y) \in r^{\mathcal{I}}$ if and only if $(y,x) \in (r^-)^{\mathcal{I}}$, and we can thus rephrase our new constraints using *teaches*$^-$

instead of *taught-by*:

$$\begin{array}{rcl}\mathsf{Professor} & \sqsubseteq & \mathsf{Teacher},\\ \mathsf{Course} & \sqsubseteq & \forall \mathit{teaches}^{-}.\neg\mathsf{Professor}.\end{array}$$

We use $\mathcal{T}''_{ex}$ for the extension of the TBox $\mathcal{T}_{ex}$ from Figure 2.3 with the above two GCIs. Please note that Professor is indeed unsatisfiable with respect to $\mathcal{T}''_{ex}$: assume we had an interpretation $\mathcal{I}$ with $p \in \mathsf{Professor}^{\mathcal{I}}$. Again, this implies that $p \in \mathsf{Teacher}^{\mathcal{I}}$, and hence $\mathcal{T}_{ex}.4$ implies that there exists some c with $(p, c) \in \mathit{teaches}^{\mathcal{I}}$ and $c \in \mathsf{Course}^{\mathcal{I}}$. Now $(c, p) \in (\mathit{teaches}^{-})^{\mathcal{I}}$, and thus the second GCI above implies that $p \in (\neg\mathsf{Professor})^{\mathcal{I}}$, contradicting our assumption.

In any system based on a DL with inverse roles, it would clearly be beneficial to allow the user to introduce names for inverse roles, such as *taught-by* for $\mathit{teaches}^{-}$, *child-of* for $\mathit{has\text{-}child}^{-}$, or *part-of* for $\mathit{has\text{-}part}^{-}$. Indeed, as we will see in Chapter 8, state-of-the-art ontology languages do this.

The above line of reasoning has been repeated numerous times in DL related research:

- we want to express something, e.g., that courses are not taught by professors;
- this seems to be not possible in a satisfactory way: in contrast to our intuition, Professor was satisfiable with respect to $\mathcal{T}'_{ex}$;
- we extend our DL with a new constructor, e.g., inverse roles, which involves extending the syntax (i.e., allowing roles r^{-} in the place of role names r) *and* the semantics (i.e., fixing $(r^{-})^{\mathcal{I}}$).

2.5.2 Number restrictions

Next, assume we want to restrict the number of courses attended by students to, say, at least three and at most seven: so far, we have only said that each student attends at least one course – see $\mathcal{T}_{ex}.8$ in Figure 2.3. Again, we can try hard, e.g., using the following GCI:

$$\begin{array}{rcl}\mathsf{Student} & \sqsubseteq & \exists \mathit{attends}.(\mathsf{Course} \sqcap A) \sqcap\\ & & \exists \mathit{attends}.(\mathsf{Course} \sqcap \neg A \sqcap B) \sqcap\\ & & \exists \mathit{attends}.(\mathsf{Course} \sqcap \neg A \sqcap \neg B).\end{array}$$

This will ensure that any element in the extension of Student attends at least three courses due to the usage of the mutually contradictory concepts A, $\neg A \sqcap B$, and $\neg A \sqcap \neg B$. We will see in Section 3.2 that we cannot use a similar trick to ensure that a Student attends at most seven courses. As a consequence, (*qualified*) *number restrictions* were introduced in DLs. The fact that a DL provides number restrictions (respectively qualified number restrictions) is normally indicated by the letter $\mathcal{N}$ (respectively $\mathcal{Q}$) in its name.

Definition 2.19. For n a non-negative number, r an $\mathcal{L}$ role and C a (possibly compound) $\mathcal{L}$ concept description, a *number restriction* is a concept description of the form $(\leqslant n\, r)$ or $(\geqslant n\, r)$, and a *qualified number restriction* is a concept description of the form $(\leqslant n\, r.C)$ or $(\geqslant n\, r.C)$, where C is the qualifying concept.

Let $\mathcal{L}$ be a description logic. The description logic $\mathcal{LN}$ (respectively $\mathcal{LQ}$) is obtained from $\mathcal{L}$ by, additionally, allowing number restrictions (respectively qualified number restrictions) as concept constructors.

For an interpretation $\mathcal{I}$, its mapping $\cdot^{\mathcal{I}}$ is extended as follows, where $\#M$ is used to denote the cardinality of a set M:

$$\begin{array}{rcl}
(\leqslant n\, r)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \#\{e \mid (d,e) \in r^{\mathcal{I}}\} \leq n\},\\
(\geqslant n\, r)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \#\{e \mid (d,e) \in r^{\mathcal{I}}\} \geq n\},\\
(\leqslant n\, r.C)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \#\{e \mid (d,e) \in r^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}}\} \leq n\},\\
(\geqslant n\, r.C)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \#\{e \mid (d,e) \in r^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}}\} \geq n\}.
\end{array}$$

Concept descriptions $(=nr)$ and $(=n\, r.C)$ may be used as abbreviations for $(\leqslant n\, r) \sqcap (\geqslant n\, r)$ and $(\leqslant n\, r.C) \sqcap (\geqslant n\, r.C)$ respectively.

A qualified number restriction allows us to restrict the number of r-fillers that are *in the extension of* a concept C. In contrast, an unqualified number restriction only allows us to restrict the number of r-fillers, regardless of which concepts' extensions they belong to; this is equivalent to a qualified number restriction where the qualifying concept is $\top$, i.e., $(\leqslant n\, r) \equiv (\leqslant n\, r.\top)$ and $(\geqslant n\, r) \equiv (\geqslant n\, r.\top)$. Naming conventions are such that, in $\mathcal{ALCIQ}$, both role names and inverse roles can occur in number restrictions whereas, of course, in $\mathcal{ALCQ}$, only role names can.

In the example interpretation $\mathcal{I}$ in Figure 2.2, $m \in (\leqslant 2\, \mathit{teaches}.\mathsf{Course})^{\mathcal{I}}$ and $m \in (\geqslant 2\, \mathit{teaches}.\mathsf{Course})^{\mathcal{I}}$ because m has exactly two *teaches*-fillers in $\mathsf{Course}^{\mathcal{I}}$. The element et is in $(\geqslant 1\, \mathit{teaches}.\mathsf{Course})^{\mathcal{I}}$, but not in $(\geqslant 2\, \mathit{teaches}.\mathsf{Course})^{\mathcal{I}}$. Finally, every element in every interpretation $\mathcal{I}$ is in $(\geqslant 0\, r.C)^{\mathcal{I}}$; the concept $\exists r.C$ is equivalent to $(\geqslant 1\, r.C)$; and $\forall r.C$ is equivalent to $(\leqslant 0\, r.\neg C)$.

2.5.3 Nominals

So far, we have used individual names in ABoxes, where we have used concepts and roles to constrain their interpretation. Now, assume we want to use individual names inside concepts, e.g., we want to define the class CourseOfMary as those Courses that are taught by Mary. Clearly, we could try the following $\mathcal{ALCI}$ concept definition

$$\mathsf{CourseOfMary} \equiv \mathsf{Course} \sqcap \exists \mathit{teaches}^{-}.\mathsf{Mary}, \tag{2.4}$$

but this would not work for the following two reasons. First, when combined with the ABox from Figure 2.4, the Concept Definition 2.4 would lead to a syntax error since, in Definition 2.6, we have said that individual names are disjoint from concept names, hence Mary cannot occur both as an individual and as a concept name. Second, if we were to allow Mary to occur in place of a concept, we would need to say what this means for Mary's interpretation: in every interpretation $\mathcal{I}$, $\mathsf{Mary}^{\mathcal{I}}$ is an element of the interpretation domain, but concepts are interpretated as *sets* of elements. To enable the use of individual names in concepts and avoid these problems, *nominals* have been introduced. The fact that a DL provides nominals is normally indicated by the letter $\mathcal{O}$ in its name, for the "o" in nominal and because $\mathcal{N}$ is already used for unqualified number restrictions.

Definition 2.20. For b an individual name in $\mathbf{I}$, $\{b\}$ is called a *nominal*.

Let $\mathcal{L}$ be a description logic. The description logic $\mathcal{LO}$ is obtained from $\mathcal{L}$ by allowing nominals as additional concepts.

For an interpretation $\mathcal{I}$, its mapping $\cdot^{\mathcal{I}}$ is extended as follows:

$$(\{a\})^{\mathcal{I}} = \{a^{\mathcal{I}}\}.$$

Hence in $\mathcal{ALCOI}$, we can define the above mentioned concept using the following $\mathcal{ALCOI}$ concept definition:

$$\mathsf{CourseOfMary} \equiv \mathsf{Course} \sqcap \exists \mathit{teaches}^{-}.\{\mathsf{Mary}\}. \tag{2.5}$$

So, by putting curly brackets around the individual name Mary, we have turned it into a concept and can therefore use it inside a concept. To see the additional expressive power provided by $\mathcal{ALCOI}$ over $\mathcal{ALCI}$, please note that, for example, CS600 is an instance of CourseOfMary with respect to Concept Definition 2.5 and $\mathcal{A}_{ex}$ from Figure 2.4.

2.5.4 Role hierarchies

Coming back to our example on lazy students, assume that we want to define lazy students as those who do not actively attend anything (and thus also no course):

$$\mathsf{LazyStudent} \equiv \mathsf{Student} \sqcap \forall \mathit{attendsActively}.\bot. \tag{2.6}$$

Let $\mathcal{T}'_{ex}$ be the TBox from Figure 2.3 extended with the above definition, and consider the ABox $\mathcal{A} = \{(\mathsf{Bob}, \mathsf{CS600}) : \mathit{attendsActively}\}$; then the KB $\mathcal{K} = (\mathcal{T}'_{ex}, \mathcal{A})$ should entail that Bob is a student, but not a lazy one. However, we find that $\mathcal{K} \not\models \mathsf{Bob} : \mathsf{Student}$ since we did not capture the intended relationship between $\mathit{attendsActively}$ and $\mathit{attends}$, namely that the former implies the latter. *Role inclusion axioms* are the DL constructors that can capture this implication. Their availability is indicated by the letter $\mathcal{H}$ in the logic's name.

Definition 2.21. Let $\mathcal{L}$ be a description logic.

A *role inclusion axiom* (RIA) is an axiom of the form $r \sqsubseteq s$ for r, s $\mathcal{L}$ roles.[5]

The DL $\mathcal{LH}$ is obtained from $\mathcal{L}$ by allowing, additionally, role inclusion axioms in TBoxes.

For an interpretation $\mathcal{I}$ to be a *model of* a role inclusion axiom $r \sqsubseteq s$, it has to satisfy

$$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}.$$

An interpretation is a *model of* a TBox if it satisfies all concept and role inclusion axioms in it.

Continuing our lazy student example, we can now add the following role inclusion axiom to $\mathcal{T}'_{ex}$ and call the result $\mathcal{T}''_{ex}$:

$$\mathit{attendsActively} \sqsubseteq \mathit{attends}. \tag{2.7}$$

We will then find that $(\mathcal{T}''_{ex}, \mathcal{A}) \models \mathsf{Bob} : \mathsf{Student}$, while $\mathsf{LazyStudent}$ is satisfiable with respect to $\mathcal{T}''_{ex}$.

2.5.5 Transitive roles

As a last extension, we consider transitive roles. Consider our running example $\mathcal{T}_{ex}$ in Figure 2.3 extended with the following axioms that in-

[5] That is, $r, s \in \mathbf{R}$ if $\mathcal{L}$ does not support inverse roles, and r, s possibly inverse roles if $\mathcal{L}$ supports inverse roles.

troduce the notion of a section:

$$\begin{array}{rcl}\mathsf{Course} & \sqsubseteq & \exists \mathsf{\textit{hasPart}}.\mathsf{Section} \sqcap \forall \mathsf{\textit{hasPart}}.\mathsf{Section},\\ \mathsf{Section} & \sqsubseteq & \forall \mathsf{\textit{hasPart}}.\mathsf{Section},\\ \mathsf{TeachableCourse} & \equiv & \mathsf{Course} \sqcap \forall \mathsf{\textit{hasPart}}.\mathsf{Ready}.\end{array}$$

Given that sections of a course can contain other sections which, in turn, can contain sections, the question arises what we mean by a teachable course. Consider the following example interpretation $\mathcal{I}$:

$$\begin{array}{rcl}\Delta^{\mathcal{I}} & = & \{c, s_1, s_2, s_3, \ldots\},\\ \mathsf{Section}^{\mathcal{I}} & = & \{s_1, s_2, s_3\},\\ \mathsf{Ready}^{\mathcal{I}} & = & \{s_1, s_2\},\\ \mathsf{Course}^{\mathcal{I}} & = & \{c\},\\ \mathsf{\textit{hasPart}}^{\mathcal{I}} & = & \{(c, s_1), (c, s_2), (s_1, s_3)\}.\end{array}$$

Now $c \in (\mathsf{TeachableCourse})^{\mathcal{I}}$ because it is a Course and all of its (immediate) sections are *Ready*. Intuitively, however, we might not expect this, because c *hasPart* s_1, s_1 *hasPart* s_3, and s_3 is not Ready. We might try to address this issue by using the following stricter definition of TeachableCourse:

$$\mathsf{TeachableCourse} \equiv \mathsf{Course} \sqcap \forall \mathsf{\textit{hasPart}}.(\mathsf{Ready} \sqcap \forall \mathsf{\textit{hasPart}}.\mathsf{Ready}).$$

This would work for the above interpretation $\mathcal{I}$, but not for others where we have longer *hasPart*-paths. In particular, if we wanted to define TeachableCourse as those courses all of whose *direct and indirect* sections are ready, regardless of the lengths of paths that relate a course to its (direct or indirect) sections, then we need to consider *transitive roles*.

Definition 2.22. Let $\mathcal{L}$ be a description logic. A *role transitivity axiom* is an axiom of the form $\mathsf{Trans}(r)$ for r an $\mathcal{L}$ role.[6]

The name of the DL that is the extension of $\mathcal{L}$ by allowing, additionally, transitivity axioms in TBoxes, is usually given by replacing $\mathcal{ALC}$ in $\mathcal{L}$'s name with $\mathcal{S}$.

For an interpretation $\mathcal{I}$ to be a *model of* a role transitivity axiom $\mathsf{Trans}(r)$, $r^{\mathcal{I}}$ must be transitive.

An interpretation $\mathcal{I}$ is a *model of* a TBox $\mathcal{T}$ if $\mathcal{I}$ satisfies each of the axioms in $\mathcal{T}$.

Naming conventions are slightly more complicated for transitive roles.

[6] That is, $r \in \mathbf{R}$ if $\mathcal{L}$ does not support inverse roles, and r is a possibly inverse role if $\mathcal{L}$ supports inverse roles.

In order to avoid having longer and longer names for DLs, the extension of $\mathcal{ALC}$ with role transitivity axioms is usually called $\mathcal{S}$ (due to similarities with the modal logic **S4**); e.g., the extension of $\mathcal{ALCIQ}$ with transitive roles is called $\mathcal{SIQ}$, and the extension of $\mathcal{ALCHIQ}$ with transitive roles is called $\mathcal{SHIQ}$. However, in some cases $_{R^+}$ is used to indicate transitive roles; using this naming scheme, $\mathcal{SHIQ}$ would be written $\mathcal{ALCHIQ}_{R^+}$.

It is important to understand the difference between transitive roles and the transitive closure of roles. Transitive closure is a role *constructor*: given a role r, transitive closure can be used to construct a role r^+, with the semantics being that $(r^+)^\mathcal{I} = (r^\mathcal{I})^+$. In a logic that includes both transitive roles and role inclusion axioms, e.g., $\mathcal{SH}$, adding axioms $\mathsf{Trans}(s)$ and $r \sqsubseteq s$ to a TBox $\mathcal{T}$ ensures that in every model $\mathcal{I}$ of $\mathcal{T}$, $s^\mathcal{I}$ is transitive, and $r^\mathcal{I} \subseteq s^\mathcal{I}$. However, we cannot enforce that s is the smallest such transitive role: s is just *some* transitive role that includes r. In contrast, the transitive closure r^+ of r is, by definition, the *smallest* transitive role that includes r; thus we have:

$$\{\mathsf{Trans}(s), r \sqsubseteq s\} \models r \sqsubseteq r^+ \sqsubseteq s.$$

This finishes our overview of various extensions of $\mathcal{ALC}$. Although we have covered several of the most prominent extensions, the overview is far from exhaustive, and many other extensions have been studied in the literature, including concrete domains (see Section 5.3.2), role value maps (see Section 5.3.1) and role boxes (see Section 8.1.2); the Appendix summarises the syntax and semantics of the DL constructors and axioms used in this book.

2.6 DLs and other logics

This section explains the close relationship between DLs and other logics, namely first-order logic (also known as first-order predicate logic or first-order predicate calculus) and modal logic. It is aimed at those readers who know one or both of these logics, and should provide these readers with a deeper understanding of the material and of the field. We suggest that readers not familiar with these logics skip this section.

2.6.1 DLs as decidable fragments of first-order logic

Most DLs can be seen as decidable fragments of first-order logic, although some provide operators such as transitive closure of roles or fixpoints that make them decidable fragments of second-order logic [Bor96]. Viewing role names as binary relations and concept names as unary relations, we can translate TBox axioms and ABox assertions into first-order logic formula, e.g.,

$$\begin{aligned} \exists \mathit{attends}.\top &\sqsubseteq \mathsf{Person}, \\ \mathsf{Teacher} &\equiv \mathsf{Person} \sqcap \exists \mathit{teaches}.\mathsf{Course}, \\ \mathsf{Mary} : \mathsf{Teacher} & \end{aligned}$$

can be translated into

$$\begin{aligned} &\forall x.(\exists y.\mathit{attends}(x, y) \Rightarrow \mathsf{Person}(x)), \\ &\forall x.(\mathsf{Teacher}(x) \Leftrightarrow \mathsf{Person}(x) \wedge \exists y.(\mathit{teaches}(x, y) \wedge \mathsf{Course}(y))), \\ &\mathsf{Teacher}(\mathsf{Mary}). \end{aligned}$$

Please note how TBox axioms correspond to universally quantified (bi-) implications without free variables, and how ABox assertions correspond to ground facts.

To formalise this translation, we define two translation functions, π_x and π_y, that inductively map $\mathcal{ALC}$ concepts into first-order formulae with one free variable, x or y:[7]

$$\begin{aligned} \pi_x(A) &= A(x), & \pi_y(A) &= A(y), \\ \pi_x(C \sqcap D) &= \pi_x(C) \wedge \pi_x(D), & \pi_y(C \sqcap D) &= \pi_y(C) \wedge \pi_y(D), \\ \pi_x(C \sqcup D) &= \pi_x(C) \vee \pi_x(D), & \pi_y(C \sqcup D) &= \pi_y(C) \vee \pi_y(D), \\ \pi_x(\exists r.C) &= \exists y.r(x, y) \wedge \pi_y(C), & \pi_y(\exists r.C) &= \exists x.r(y, x) \wedge \pi_x(C), \\ \pi_x(\forall r.C) &= \forall y.r(x, y) \Rightarrow \pi_y(C), & \pi_y(\forall r.C) &= \forall x.r(y, x) \Rightarrow \pi_x(C). \end{aligned}$$

Next, we translate a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ as follows, where $\psi[x \mapsto a]$ is the formula obtained from ψ by replacing all free occurrences of x with a:

$$\begin{aligned} \pi(\mathcal{T}) &= \forall x. \bigwedge_{C \sqsubseteq D \in \mathcal{T}} (\pi_x(C) \Rightarrow \pi_x(D)), \\ \pi(\mathcal{A}) &= \bigwedge_{a\,:\,C \in \mathcal{A}} \pi_x(C)[x \mapsto a] \wedge \bigwedge_{(a,b)\,:\,r \in \mathcal{A}} r(a, b). \end{aligned}$$

This translation clearly preserves the semantics: we can easily view DL interpretations as first-order interpretations and vice versa.

Theorem 2.23. *Let $(\mathcal{T}, \mathcal{A})$ be an $\mathcal{ALC}$-knowledge base, C, D possibly compound $\mathcal{ALC}$ concepts and b an individual name. Then*

(i) *$(\mathcal{T}, \mathcal{A})$ is satisfiable if and only if $\pi(\mathcal{T}) \wedge \pi(\mathcal{A})$ is satisfiable,*

[7] This definition is inductive (or recursive), with π_x calling π_y and vice versa, from compound concepts to their constituent parts.

(ii) $C \sqsubseteq_{\mathcal{T}} D$ *if and only if* $\pi(\mathcal{T}) \Rightarrow \forall x.(\pi_x(C) \Rightarrow \pi_x(D))$ *is valid, and*

(iii) b *is an instance of* C *with respect to* $(\mathcal{T}, \mathcal{A})$ *if and only if* $(\pi(\mathcal{T}) \wedge \pi(\mathcal{A})) \Rightarrow \pi_x(C)[x \mapsto b]$ *is valid.*

This translation not only provides an alternative way of defining the semantics of $\mathcal{ALC}$, but also tells us that all reasoning problems for $\mathcal{ALC}$ knowledge bases are decidable: the translation of a knowledge base uses only variables x and y, and thus yields a formula in the *two-variable fragment of first-order logic*, for which satisfiability is known to be decidable in nondeterministic exponential time [GKV97]. Similarly, the translation of a knowledge base uses quantification only in a rather restricted way, and therefore yields a formula in the *guarded fragment* [ANvB98], for which satisfiability is known to be decidable in deterministic exponential time [Grä99]. As we can see, the exploration of the relationship between DLs and first-order logics even gives us upper complexity bounds for free.

The translation of more expressive DLs may be straightforward, or more difficult, depending on the additional constructs: inverse roles can be captured easily in both the guarded and the two-variable fragments by simply swapping the variable places; e.g., $\pi_x(\exists r^-.C) = \exists y.r(y,x) \wedge \pi_y(C)$. Number restrictions can be captured using (in)equality or so-called *counting quantifiers*. It is known that the two-variable fragment with counting quantifiers is still decidable in nondeterministic exponential time [PST00]. The guarded fragment, when restricted carefully to the so-called *action guarded fragment* [GG00], can still capture a variety of features such as number restrictions, inverse roles and fixpoints, while remaining decidable in deterministic exponential time.

2.6.2 DLs as cousins of modal logic

Description logics are closely related to modal logic, yet they have been developed independently. This close relationship was discovered only rather late [Sch91], and has been exploited quite successfully to transfer complexity and decidability results as well as reasoning techniques [Sch94, DGL94a, HPS98, Are00]. It is not hard to see that $\mathcal{ALC}$ concepts can be viewed as syntactic variants of formulae of multi-modal $\mathbf{K}_{(\mathbf{m})}$: Kripke structures can easily be viewed as DL interpretations, and vice versa; we can then view concept names as propositional variables, and role names as modal parameters, and realise this correspondence through the mapping π as follows:

$$\begin{aligned}
\pi(A) &= A, \text{ for concept names } A,\\
\pi(C \sqcap D) &= \pi(C) \wedge \pi(D),\\
\pi(C \sqcup D) &= \pi(C) \vee \pi(D),\\
\pi(\neg C) &= \neg\pi(C),\\
\pi(\forall r.C) &= [r]\pi(C),\\
\pi(\exists r.C) &= \langle r\rangle\pi(C).
\end{aligned}$$

The translation of DL knowledge bases is slightly more tricky: a TBox $\mathcal{T}$ is satisfied only in those structures where, for each $C \sqsubseteq D$, $\neg\pi(C) \vee \pi(D)$ holds *globally*, i.e., in each world of our Kripke structure (or equivalently, in each element of our interpretation domain). We can express this using the *universal modality*, that is, a special modal parameter U that is interpreted as the total relation in all interpretations. Before we discuss ABoxes, let us first state the properties of our correspondence so far.

Theorem 2.24. *Let $\mathcal{T}$ be an $\mathcal{ALC}$ TBox and E, F possibly compound $\mathcal{ALC}$ concepts. Then:*

(i) *F is satisfiable if and only if $\pi(F)$ is satisfiable;*
(ii) *F is satisfiable with respect to $\mathcal{T}$ if and only if $\bigwedge_{C\sqsubseteq D\in\mathcal{T}}[U](\pi(C) \Rightarrow \pi(D)) \wedge \pi(F)$ is satisfiable;*
(iii) *$E \sqsubseteq_{\mathcal{T}} F$ if and only if $\bigwedge_{C\sqsubseteq D\in\mathcal{T}}[U](\pi(C) \Rightarrow \pi(D)) \Rightarrow [U](\pi(E) \Rightarrow \pi(F))$ is valid.*

Like TBoxes, ABoxes do not have a direct correspondence in modal logic, but they can be seen as a special case of a modal logic constructor, namely *nominals*.[8] These are special propositional variables that hold in exactly one world; they are the basic ingredient of *hybrid logics* [ABM99].[9] Usually, modal nominals come with a special modality, the @-operator, that allows one to refer to the (only) world in which a nominal a holds. For example, $@_a\psi$ holds if, in the world where a holds, ψ holds as well. Hence an ABox assertion of the form $a : C$ corresponds to the modal formula $@_a\pi(C)$, and an ABox assertion $(a, b) : r$ corresponds to $@_a\langle r\rangle b$. In this latter formula, we see how nominals can act both as a parameter to the @ operator, like a, and as a propositional variable, like

[8] Description logic nominals as introduced in Section 2.5.3 have received their name from modal logic.

[9] Please note that modal nominals are special propositional variables, whereas DL nominals are concepts constructed from individual names by enclosing them in curly brackets.

b. In DLs that provide nominals, such as $\mathcal{ALCO}$, there is traditionally no counterpart to the @ operator: for example, the concept $A \sqcap \exists r.\{\mathsf{b}\}$ corresponds to the modal formula $A \wedge \langle r \rangle b$, where b is a nominal, but the formula $A \wedge \langle r \rangle (B \vee @_b B)$ does not have an $\mathcal{ALCO}$ counterpart since it uses the @ operator to say "B holds at the place where b holds".

2.7 Historical context and literature review

This chapter tries to introduce the basic, standard notions relevant in Description Logic: concepts, GCIs and assertions, TBoxes and ABoxes, interpretations and models, entailments and reasoning services. We left out the history of the area, and will only sketch it very briefly here. Description logics have had various other names in the past, e.g., terminological knowledge representation systems, concept languages, frame languages etc. They were developed to give a well-defined semantics to knowledge representation systems, in particular KL-ONE [BS85]. When it turned out that the DL underlying KL-ONE was undecidable [Sch89], a lot of work was carried out in trying to understand the sources of undecidability (see, e.g., [PS89]) and to identify useful, decidable DLs: the DL underlying the CLASSIC system [PSMB+91] was designed to be tractable, and the CLASSIC system was the first one to be used by non-experts (in an application that supported engineers in configuring communication systems). With $\mathcal{ALC}$ [SS91], the first propositionally closed DL was introduced and proven to be decidable, but of a computational complexity that was believed to preclude practically useful implementations (see Chapter 5). Research in the 1990s and 2000s saw a plethora of results regarding decidability and computational complexity for a wide range of DLs that differed in

- the constructors they allowed for forming concepts and roles, e.g., inverse role, number restrictions, concrete domains,
- the kind of axioms they allowed, in particular regarding roles, e.g., role hierarchies, but also axioms that expressed default or probabilistic statements and
- the semantics they employed, e.g., least or greatest fixpoint semantics for cyclic TBoxes, fuzzy or probabilistic semantics,

with the interest in a specific DL usually driven by some specific knowledge representation application. This period also saw the design of various algorithms to decide subsumption, satisfiability, consistency etc., together with proofs of their correctness, i.e., together with proofs that they always terminate and never give the wrong answer. It also saw implementations and optimisations of these algorithms, and further algorithms for TBox classification.

In this way, DLs developed into a rich family of logics for which sources of complexity and undecidability are well understood. In parallel, researchers in this area noticed the close relationship between DLs and other logics: the relationship between modal and description logics is first discussed in [Sch91] and explored further in [Sch94, DGL94a, DGL94b]. The relationship between a wide range of DLs and first-order logic was first described in [Bor96].

There is a huge and still growing body of work describing these results, far too big to list here in a suitable way. We suggest consulting *The Description Logic Handbook* [BCM$^+$07] for a general overview, the informal proceedings of the annual International Workshop of Description Logics, which are almost all available electronically at `dl.kr.org`, as well as the proceedings of meetings on artificial intelligence, knowledge representation and reasoning such as AAAI (the conference of the Association for the Advancement of Artificial Intelligence), CADE (the International Conference on Automated Deduction), ECAI (the European Conference on Artificial Intelligence), IJCAI (the International Joint Conference on Artificial Intelligence), IJCAR (the International Joint Conference on Automated Reasoning), and KR (the International Conference on Principles of Knowledge Representation and Reasoning). Furthermore, journals that are often used by researchers in Description Logic to publish their work include AIJ (*Artificial Intelligence – an International Journal*), JAIR (*Journal of Artificial Intelligence Research*), and JLC (*Journal of Logic and Computation*).