Friedrich-Alexander-Universität
Erlangen-Nürnberg

T.cs

Chair for Computer Science 8
Theoretical Computer Science

Transforming OWL ontologies with bounded self-reference to plain OWL ontologies

Bachelor Thesis in Computer Science

Thorsten Wißmann

Advisors:

Prof. Dr. Lutz Schröder

Dr. Daniel Gorín

Erlangen, February 28th, 2013

Erklärung

Ich versichere, dass ich die vorliegende Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, 28. Februar 2013

Thorsten Wißmann

Acknowledgments

This thesis was written under the supervision of the Chair for Computer Science 8 (Theoretical Computer Science) at the Friedrich-Alexander-Universität Erlangen-Nürnberg. I want to thank my advisors Prof. Dr. Lutz Schröder and Dr. Daniel Gorin for their support for writing this thesis and implementing the according software. Finally, I want to thank my family for encouraging me personally.

Abstract

There are many description logics, each with different expressiveness. We will focus on $\mathcal{ALCQ}\mathsf{me}_2$, which is obtained by extending $\mathcal{ALCQ}$ with the $\mathsf{I\text{-}me}$-construct by Marx in a bounded way. Its relation to $\mathcal{ALCHIQ}$ is shown: on the one hand role inverses and hierarchies can be encoded in $\mathcal{ALCQ}\mathsf{me}_2$ and on the other hand an encoding of an arbitrary $\mathcal{ALCQ}\mathsf{me}_2$ TBox in an $\mathcal{ALCHIQ}$ TBox and RBox is given in a consistency-preserving way, exploiting something similar to the tree model property known from other description logics. This implicitly gives a consistency checking algorithm for $\mathcal{ALCQ}\mathsf{me}_2$ TBoxes, as there is a consistency checker for $\mathcal{ALCHIQ}$.

The analogous thing also is done for the widespread ontology format OWL. A method is given to effectively encode an $\mathcal{ALCHIQ}\mathsf{me}_2$ TBox and RBox in OWL. The implementation of the reduction – written as a part of this thesis – can be used to convert such an $\mathcal{ALCHIQ}\mathsf{me}_2$ ontology consistency preserving into a plain OWL ontology whose satisfiability then can be checked with any OWL reasoner. Empirical tests show that the blowup regarding consistency checking time is acceptable.

Contents

1 Introduction

The need for representing knowledge arises in many different fields. Let us look at three of them: the semantic web, medical applications and social networking.

The field of the Semantic Web – sometimes considered as the Web 3.0 – purposes to combine and connect the information distributed all over the World Wide Web, to provide a better user experience. Imagine you have to feed your cat but also want to save money by only buying food with discount. Querying a search engine for stores that offer cat food at a discount then also lists shops which announce "20% off on pet food". Of course, this is only possible if it is known to the database that cats are pets. If another web page provides the information that cats can eat beef, the response to the query by the knowledge database also would contain butchers that sell beef, currently on discount.

In the second field, the medical sector, different institutions for health care want to share their research results with each other, e.g. which disease occurs at which organ under which circumstances.

The third field, very prominent in the media nowadays, is social networking. The main knowledge there is the relation between people and their attitudes, for example their gender or who is kin to whom.

All these different kinds of information need to be made persistent in some way. The solution is the use of ontologies: the task of an ontology is to describe a situation of some entities and the relations in between them. In the first field, the entities could be different kinds of food. Additional relations could describe where it is sold and which kind of species consumes it. In the second field the entities are human organs and diseases. Relations would be how those affect each other. In the field of human relations the entities could be people and attitudes of them. E.g. being friendly can be seen as an attitude when defining that only those people are friendly who are liked by someone. Being a father also can be an entity when saying that a person is a father if he is male and there is at least one person he is the parent of.

The intention behind that is not just to encode the situation but also to be able to infer more knowledge about the situation by a computer program; this process is called *reasoning*. Interesting knowledge about an ontology is for example, whether the description is contradictory or not, or e.g. how the terms imply each other: is a father also a parent? Is a parent also a father? Are the terms parent and father equivalent?

Thus it is necessary that an ontology is expressed in a formal and syntactically well-defined way. The most common ontology file format is the *Web Ontology Language*

(OWL)[1], defined by the W3C. For OWL, many reasoning applications are available. The reasoning problems for OWL are all decidable. Even if the theoretical worst-case complexity of most of the reasoning problems is very hard, the time needed to solve the problems for ontologies which appear in practice is acceptable.

But beside all these nice properties, OWL allows describing self-reference only in a very restricted manner. Self-referential constructs occur often in natural language, so we are interested in the use of those in ontologies. An example for a self-referential construct in the context of relationships between people are narcissists: they are those people who love themselves. Further examples for descriptions using self-reference are given by statements like "I only like other people who also like me" or "each person meets another person who is smarter". From these examples the narcissist is the only one expressible in OWL, i.e. it only allows the restricted way of directly stating how a certain group of individuals is related to itself.

The reason for this restriction is the following: it is known that allowing self-reference in general makes reasoning problems undecidable. But as it turned out recently, it stays decidable under conditions, which are weaker than those stipulated by OWL. The conditions are met if we allow saying "I only like other people who also like me" but prohibit saying "I only like people who like other people who happen to like me". So when phrasing the description in a first person perspective, the nesting depth of the relative clause in which the *me* occurs is the essential criterion. So it is only allowed to nest the sentences at most 2 levels deep – the main clause and one relative clause.

We will extend the OWL format with this bounded version of self-reference, for which it stays decidable. To be able to perform reasoning, such an OWL ontology with self-reference is transformed into a plain OWL ontology by the implementation written along with this thesis. Then the actual reasoning task can be done by any OWL reasoning software.

The theoretical background of this transformation is described by this thesis, together with the adaption to OWL and some benchmarks to see that it is applicable in practice.

[1] The acronym was changed from WOL to OWL [1]

2 Preliminaries

The family of description logics consists of many different formal languages used for knowledge representation. Each of them has own properties and a certain expressiveness. To give a solid base for what is used in the following chapters, first some mathematical definitions and then an introduction to description logics and ontologies is given. After defining these basics, an outline of the thesis is given in Section 2.7.

This chapter will give the definitions needed for this work. They are mostly based on the description logic primer by Krötzsch, Simancik, and Horrocks [2].

2.1 Basic Definitions

To standardize the notation the following basics are defined.

Definition 2.1 (Boolean operations)**.** Boolean values will be expressed as members of the set $2 := \{0, 1\} = \{true, false\}$ together with the binary operators $\wedge, \vee : 2 \times 2 \to 2$ and unary operator $^{-}: 2 \to 2$, which are defined by Table 2.1 ■

a	b	$a \wedge b$	$a \vee b$	$\bar{a}$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Table 2.1

Definition 2.2 (Disjoint union)**.** $A \dot\cup B$ denotes the disjoint union of two sets A and B. It behaves like the conventional union while emphasizing that A and B are considered to be disjoint. If A and B intersect, then the name clash is resolved by renaming the according elements properly.

We will typically use $\dot\cup$ when extending a set S with an *additional* element x: $S' := S \dot\cup \{x\}$. ■

Definition 2.3 (Power set)**.** The power set – the set of all subsets – of a given set S is denoted by $\wp(S)$. ■

Definition 2.4. For a given – not necessarily finite – set X, a relation $R \subseteq X^2$ is called a *tree* if there is an $r \in X$ and a function $f : X \setminus \{r\} \to X$ such that:

- $R = f^{-1}$.
- For each $x \in X \setminus \{r\}$, there is a $n \in \mathbb{N}$ with: $f^n(x) = r$.

The node r is called the root of the tree and f is the father function which maps each node to its father in the tree. ■

2.2 Description Logics and Ontologies

An *Ontology* is a finite set of *names* and *axioms*. The names are the terms of a certain domain to talk about and the axioms are facts in this domain formally describing the relation of these terms.

Axioms represent knowledge in a formal way. Having the syntactical context defined later, we will be able to form axioms which represents the following knowledge for the terms *parent*, *is parent of*, *grandparent*, *mother* and *female*:

- A *parent* is someone who *is the parent of* someone.
- A *grandparent* is someone who *is the parent of* someone who *is the parent of* somebody else.
- A *mother* is a *female parent*.

The usecase for ontologies is to store and to automatically infer further knowledge from the axioms in the ontology. E.g. in the above example the following can be inferred:

- A *grandparent* is also a *parent*.
- A *mother is the parent of* someone.

In this example, the terms can be divided into two different kind of groups: each of *parent*, *grandparent*, *mother* and *female* describes a kind of individuals. So given a concrete set of relatives, each individual is either female or not, is either a parent or not, and so on. Such terms are called *concepts*. That means that in a concrete situation, each concept can be interpreted as a subset of all individuals.

In general, concepts can also be composed together to form complex expressions. E.g. *those who are parents and not female* is also a concept and stands for a subset of all individuals, even though it is not explicitly given a compact name.

The term *is parent of* describes the relation between individuals: in a concrete set of relatives, for each pair a, b of individuals, a either is the parent of b or not. Such a kind of term is called *role* and in a concrete situation each role is to be interpreted as binary relation in the set of individuals. We have seen already, that concepts can be defined using roles, e.g. *those who are parent of someone*.

The remaining question is how to exactly express these axioms, concepts and roles as

formulas. To be able to describe the meaning of concepts precisely and to do things automatically, a formal syntax for axioms is needed. On the one hand we want the expressive power to model ideas like *grandparent*, but on the other hand problems like inferring knowledge still should be computable, preferably with a good complexity. As there is not one perfect compromise between expressiveness and complexity, there is not *the description logic* but a family of languages with different flavors, how axioms can be built.

2.3 The Description Logic $\mathcal{ALC}$

The most basic description logic is considered to be $\mathcal{ALC}$. Many other description logics are extensions of $\mathcal{ALC}$. Let us look at $\mathcal{ALC}$ first:

Definition 2.5 (Syntax of $\mathcal{ALC}$)**.** Let N_C, N_R be disjoint finite sets of concept names and role names. In the description logics we consider, complex concepts and roles are formulas and can be built by combining the names with certain operators. The sets $\mathfrak{C}$ and $\mathfrak{R}$ denote the set of concept and role expressions that can be built. In the base case of $\mathcal{ALC}$, no complex role names are allowed, so the set $\mathfrak{R}$ of role expressions only contains role names:

$$\mathfrak{R} = N_R$$

Concept expressions can be built using concept names and many operators. The set of concept expressions $\mathfrak{C}$ is defined as follows:

$$\begin{aligned} &\mathfrak{C} \supseteq N_C, \\ &\mathfrak{C} \ni C \sqcap D, \neg C, \exists R.C \qquad\qquad \text{for all } C, D \in \mathfrak{C}, R \in \mathfrak{R}. \end{aligned}$$

If a concept expression $C \in \mathfrak{C}$ is also in N_C, then C is called an *atomic concept*, otherwise it is called *complex*.

In $\mathcal{ALC}$ it can be expressed, that for two concepts $C, D \in \mathfrak{C}$, C is a subconcept of D, syntactically expressed by the axiom:

$$C \sqsubseteq D$$

Two concepts $C, D \in \mathfrak{C}$ also can be marked as being equivalent, expressed by this axiom:

$$C \equiv D$$

These both type of axioms which are talking about the relation of concepts are called *TBox* axioms. For concrete N_C and N_R, a concrete finite set of TBox axioms is called *TBox*. Later we also will define other kind of axioms, i.e. different type of "Boxes". ■

One may have noticed that the intersection $\sqcap$ of two concepts is defined as a formula constructor but not the union. The first reason for that is to keep the syntax as compact

as possible. Secondly it is still possible two define the union and many other operators implicitly:

$$C \sqcup D :\equiv \neg(\neg C \sqcap \neg D), \qquad \forall R.C :\equiv \neg\exists R.\neg C, \tag{2.1}$$

$$C \to D :\equiv \neg C \sqcup D, \qquad C \leftrightarrow D :\equiv (C \to D) \sqcap (D \to C), \tag{2.2}$$

$$\bot :\equiv N \sqcap \neg N \qquad \top :\equiv \neg\bot \tag{2.3}$$

for $C, D \in \mathfrak{C}$, $R \in \mathfrak{R}$ and for any atomic concept $N \in N_C$.

The syntax already gives an intuitive meaning of the formulas. So e.g. bottom $\bot$ contains no individuals and its counterpart top $\top$ contains all individuals. To encode the example from before, we need suitable concept and role names N_C and N_R:

$$N_C = \{\text{Parent}, \text{Grandparent}, \text{Female}\}$$
$$N_R = \{\text{ParentOf}\}$$

The axioms, including the inferred facts, from the example in Section 2.2 are phrased as follows:

$$\begin{aligned} \text{Parent} &\equiv \exists\text{parentOf}.\top \\ \text{Grandparent} &\equiv \exists\text{parentOf}.\exists\text{parentOf}.\top \\ \text{Mother} &\equiv \text{Female} \sqcap \text{Parent} \\ \text{Grandparent} &\sqsubseteq \text{Parent} \\ \text{Mother} &\sqsubseteq \exists\text{parentOf}.\top \end{aligned}$$

Let us look at the $\exists$ term more closely. $\exists R.C$ is a concept and describes all individual that are in an R-relation to someone in C. For example

$$\exists \underbrace{\text{parentOf}}_{R}.\underbrace{\top}_{C}$$

are exactly those individuals that are the parent of someone. The concept C can be a complex concept expression again, so the concept

$$\exists \underbrace{\text{parentOf}}_{R}.\underbrace{\exists\text{parentOf}.\top}_{C}$$

contains those who are the parent of a parent.

The only non-intuitive construct is the derived $\forall R.C$: it is defined as those individuals for which no individual exists which they are in R relation to and which is not in C. I.e. it contains exactly those individuals which are only related via R to individuals that are in C. An example is the individuals who only have daughters, formally:

$$\forall\text{parentOf}.\text{female}$$

Similar to the universal quantifier in logics, this concept also comprehends individuals who do not have any children: all their children are female.

All this already gave an intuition about the meaning of the formulas. The exact interpretation is given by the semantics.

Definition 2.6 (Semantics of $\mathcal{ALC}$)**.** For a given – not necessarily finite – set of individuals $\Delta^{\mathcal{I}}$, the operation $\cdot^{\mathcal{I}}$ maps each atomic concept to a subset of individuals and each role to a binary relation between individuals. So for an atomic concept $N \in N_C$, $N^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ describes the set of individuals being in the concept N. For an atomic role $R \in N_R$, $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ contains those pairs of individuals which are connected via R. The semantics of complex $\mathcal{ALC}$ concepts is defined inductively for every operator by Table 2.2. Note that the semantics definitions for the implicitly defined constructs $\sqcup$, $\top$, $\bot$ and $\forall$ are redundant here and are given for the sake of intuition.

The semantics of TBox axioms is defined by Table 2.3. Any interpretation $\cdot^{\mathcal{I}}$ mapping the concepts to a concrete domain $\Delta^{\mathcal{I}}$ must fulfill all the listed laws and each axiom. ■

Syntax	Semantics
N	$N^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$\top$	$\Delta^{\mathcal{I}}$
$\bot$	$\emptyset$
$\exists R.C$	$\{x \mid \text{it exists some } (x, y) \in R^{\mathcal{I}} \text{ with } y \in C^{\mathcal{I}}\}$
$\forall R.C$	$\{x \mid \text{for all } (x, y) \in R^{\mathcal{I}} \text{ it is } y \in C^{\mathcal{I}}\}$
with $N \in N_C, C, D \in \mathfrak{C}, R \in \mathfrak{R}$	

Table 2.2: Semantics of $\mathcal{ALC}$ concepts

Syntax	Semantics
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$
with $C, D \in \mathfrak{C}$	

Table 2.3: Semantics of $\mathcal{ALC}$ axioms

Definition 2.7 (Interpretation, Model, TBox consistency)**.** For a TBox $\mathcal{T}$, such a *non-empty* set of individuals $\Delta^{\mathcal{I}}$ with the mapping of concepts and roles and with the semantics of all axioms in $\mathcal{T}$ being fulfilled is called *interpretation* for $\mathcal{T}$, or also *model* for $\mathcal{T}$.

If a TBox $\mathcal{T}$ has at least one model, it is called *consistent* (or also *satisfiable*). Otherwise it is called *inconsistent*.

The consistency checking problem is the problem of determining if a TBox is consistent or not. ■

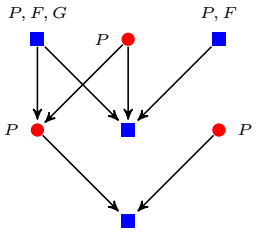
Example 2.8. Let us extend the original example a little bit:

$$N_C = \{\text{Parent}, \text{Father}, \text{Grandfather}, \text{Male}, \text{Female}\},$$
$$N_R = \{\text{parentOf}\}.$$

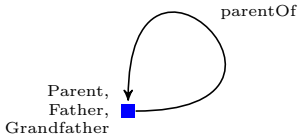
Then the following axioms are well-formed $\mathcal{ALC}$ axioms:

$$\begin{aligned} \bot &\equiv \text{Male} \sqcap \text{Female} \\ \text{Father} &\equiv \text{Male} \sqcap \text{Parent} \\ \text{Parent} &\equiv \exists \text{parentOf}.\top \\ \text{Grandfather} &\equiv \text{Male} \sqcap \exists \text{parentOf}.\exists \text{parentOf}.\top \end{aligned}$$

Given these axioms as the TBox $\mathcal{T}$, an example model is given by Figure 2.1a. Note that the formal rules stated above also allow the model in Figure 2.1b. In particular, $\mathcal{T}$ is consistent. ▲



(a) An artificial family tree; Grandfather = G, Father = F, Parent = P, parentOf is the only role.



(b) A cyclic family bloodline

Figure 2.1: Some family bloodlines. Male individuals are marked by ■, females are marked by ●.

Example 2.9. For $N_C = \{X, Y\}$, consider the TBox

$$X \equiv Y$$
$$\top \equiv X \sqcap \neg Y$$

This TBox is inconsistent, because the axiom saying that each individual fulfills X and $\neg Y$ can not be satisfied together with the axiom stating that X and Y are equivalent. ▲

2.4 Extensions of $\mathcal{ALC}$

$\mathcal{ALC}$ is considered to be the most basic description logic, because it contains the basic Boolean operators $\sqcap$, $\neg$, $\exists$. The consistency of an ontology for an extension is defined analogously to the consistency of an $\mathcal{ALC}$ TBox.

Some commonly known extensions relevant for this thesis are defined in this section. Each of them is named by some character X. $\mathcal{ALC}X$ then denotes the extension of $\mathcal{ALC}$ with this particular feature X.

2.4.1 Role Inverses

Definition 2.10 (Syntax of $\mathcal{I}$)**.** In the role inverses extension of a description logic the set of role expressions is

$$\mathfrak{R}' = \mathfrak{R} \cup \{R^- \mid R \in \mathfrak{R}\},$$

where $\mathfrak{R}$ is the set of role expression in the original description logic and $\mathfrak{R}'$ the set of role expressions in the extended one. ■

So in case of $\mathcal{ALCI}$, the set complex roles is

$$\mathfrak{R} = \{R \mid R \in N_R\} \cup \{R^- \mid R \in N_R\}.$$

Definition 2.11 (Semantics of $\mathcal{I}$)**.** The semantics of the extended description logic is compatible to the original one, with one rule added:

$$(a, b) \in R^{-\mathcal{I}} \Leftrightarrow (b, a) \in R^{\mathcal{I}},$$

for any pair of individuals $a, b \in \Delta^{\mathcal{I}}$. ■

Example 2.12. Extending the sets N_C, N_R in Example 2.8 with the role name *childOf*, the following is a well-formed $\mathcal{ALCI}$ TBox axiom:

$$\text{Parent} \equiv \exists \text{childOf}^-.\top$$

▲

2.4.2 Role Hierarchies

Definition 2.13 (Syntax of $\mathcal{H}$)**.** In the role hierarchies extension, another type of axioms are allowed to be phrased: *RBox* axioms. They describe inclusion relations between the roles. For some roles $R, S \in \mathfrak{R}$, expressing that R is a subrole of S is done by the following axiom:

$$R \sqsubseteq S$$

■

Definition 2.14 (Semantics of $\mathcal{H}$)**.** The semantics of the extended description logic is compatible to the original one, with the rule in Table 2.4 added. Given an interpretation $\mathcal{I}$, it means: for any individuals $a, b \in \Delta^{\mathcal{I}}$ being connected via R, $(a, b) \in R^{\mathcal{I}}$, they also have to be connected via S, i.e. $(a, b) \in S^{\mathcal{I}}$. ■

Example 2.15. Extending the sets N_C, N_R from Example 2.12 with the role name *fatherOf*, the following is an example $\mathcal{ALCH}$ RBox-axiom:

$$\text{fatherOf} \sqsubseteq \text{parentOf}$$

It tells: if a is the father of b, then a is also the parent of b. An example for combining role hierarchies with inverses is given by the following $\mathcal{ALCHI}$ RBox axioms:

$$\text{parentOf}^- \sqsubseteq \text{childOf}, \quad \text{childOf}^- \sqsubseteq \text{parentOf}$$

▲

Syntax	Semantics
$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
with $R, S \in \mathfrak{R}$	

Table 2.4

2.4.3 Qualified Number Restrictions

Definition 2.16 (Syntax of $\mathcal{Q}$)**.** In the qualified number restriction extension the set of concept expressions is

$$\mathfrak{C}' = \mathfrak{C} \cup \{\geq_n R.C \mid R \in \mathfrak{R}, C \in \mathfrak{C}', n \in \mathbb{N}_0\},$$

where $\mathfrak{R}$ is the set of role expression and $\mathfrak{C}$ is the original set of concept expressions. ■

Definition 2.17 (Semantics of $\mathcal{Q}$)**.** The semantics of the extended description logic is compatible to the original one, with the rule in Table 2.5 added. ■

Syntax	Semantics
$\geq_n R.C$	$\{x : \#\{y : (x, y) \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\} \geq n\}$
with $n \in \mathbb{N}_0, C \in \mathfrak{C}, R \in \mathfrak{R}$	

Table 2.5

Based on $\geq_n R.C$ the concept $\leq_n R.C$ is defined as:

$$\neg \geq_{n+1} R.C \tag{2.4}$$

Furthermore, even $\exists R.C$ and $\forall R.C$ can be expressed by $\geq$:

$$\exists R.C :\equiv \geq_1 R.C, \qquad \forall R.C :\equiv \leq_0 R.\neg C$$

The advantage of this linkage is, that in description logics including the $\mathcal{Q}$ extension it is sufficient to consider $\geq_n R.C$ in case distinctions to cover $\exists$ and $\forall$ implicitly.

Example 2.18. Given the sets N_C, N_R like in Example 2.12, the following is an example $\mathcal{ALCQ}$ TBox axiom:

$$\top \equiv \leq_2 \text{childOf.Parent} \sqcap \geq_2 \text{childOf.Parent}$$

It says, that everybody is the child of at most two parents and is also the child of at least one parents, i.e. everybody is the child of exactly two parents. ▲

2.5 More about $\mathcal{ALCHIQ}$

$\mathcal{ALCHIQ}$ and its properties take an important role in the following chapters, so $\mathcal{ALCHIQ}$ is considered here in more detail.

Definition 2.19. A concept is in *negated normal form* (NNF) if negations only occur directly in front of atomic concepts and only the following operators are used: $\exists$, $\forall$, $\geq$, $\leq$, $\sqcup$, $\sqcap$, $\neg$.

Every concept formula can be transformed into an *equivalent* formula in NNF using the definition of $\exists$, $\leq$, $\sqcup$ including the DeMorgan rule in (2.1) on page 16 as identities for pushing the negation towards the atomic concepts, as well as (2.4) to resolve $\neg \leq_n$ and $\neg \geq_n$. This mapping is done by $\mathrm{NNF} : \mathfrak{C} \to \mathfrak{C}$, defined as

$$\begin{aligned}
\mathrm{NNF}(N) &= N, \quad \mathrm{NNF}(\neg N) = \neg N,\\
\mathrm{NNF}(\neg\neg C) &= \mathrm{NNF}(C),\\
\mathrm{NNF}(\neg(C \sqcup D)) &= \mathrm{NNF}(\neg C) \sqcap \mathrm{NNF}(\neg D),\\
\mathrm{NNF}(\neg(C \sqcap D)) &= \mathrm{NNF}(\neg C) \sqcup \mathrm{NNF}(\neg D),\\
\mathrm{NNF}(\neg(\exists R.C)) &= \forall R.\mathrm{NNF}(\neg C),\\
\mathrm{NNF}(\neg(\forall R.C)) &= \exists R.\mathrm{NNF}(\neg C),\\
\mathrm{NNF}(\neg(\leq_n R.C)) &= \geq_{n+1} R.\mathrm{NNF}(C),\\
\mathrm{NNF}(\neg(\geq_n R.C)) &= \leq_{n-1} R.\mathrm{NNF}(C),
\end{aligned}$$

for $N \in N_C$, $C, D \in \mathfrak{C}$, $R \in \mathfrak{R}$ and $\leq_{n-1} R.C$ denoting $\bot$ for $n = 0$. In all other cases, NNF commutes with the operator, e.g.:

$$\mathrm{NNF}(C \sqcap D) = \mathrm{NNF}(C) \sqcap \mathrm{NNF}(D)$$

■

Definition 2.20. Given a concept $C \in \mathfrak{C}$, the set of *subconcepts* or also called *subformulas* of C is denoted by $\mathrm{Sub}(C)$. $\mathrm{Sub} : \mathfrak{C} \to \wp(\mathfrak{C})$ maps a concept to the set of its subconcepts, defined by

$$\begin{aligned}
\mathrm{Sub}(N) &= \{N\}, & \mathrm{Sub}(\neg C) &= \{\neg C\} \cup \mathrm{Sub}(C),\\
\mathrm{Sub}(C \sqcup D) &= \{C \sqcup D\} \cup \mathrm{Sub}(C) \cup \mathrm{Sub}(D), & \mathrm{Sub}(C \sqcap D) &= \{C \sqcap D\} \cup \mathrm{Sub}(C) \cup \mathrm{Sub}(D),\\
\mathrm{Sub}(\geq_n R.C) &= \{\geq_n R.C\} \cup \mathrm{Sub}(C), & \mathrm{Sub}(\leq_n R.C) &= \{\leq_n R.C\} \cup \mathrm{Sub}(C),
\end{aligned}$$

for $C, D \in \mathfrak{C}$, $N \in N_C$, $R \in \mathfrak{R}$. ■

Definition 2.21. For a model $\mathcal{I}$ of some ontology with the role names N_R, the globalized relation $\mathcal{I}$ is the union of all relations $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}, r \in N_R$:

$$x\mathcal{I}y :\Leftrightarrow (x, y) \in \bigcup_{r \in N_R} r^{\mathcal{I}}.$$

■

Definition 2.22. A description logic has the tree model property, if the following conditions are equivalent for each ontology $\mathcal{O}$:

(i) $\mathcal{O}$ has a model.

(ii) $\mathcal{O}$ has a model $\Delta^{\mathcal{I}}$ with the shape of a tree, i.e. the globalized relation $\mathcal{I}$ is a tree in $\Delta^{\mathcal{I}}$.

■

Note that for any description logic, the direction "(i)⇐(ii)" of the equivalence is trivially true, "(i)⇒(ii)" is the essential direction. $\mathcal{ALCHIQ}$ has this property[1] [3].

2.6 Reasoning

The basic task on ontologies is inferring information from the given axioms of an ontology, which is called *reasoning*. A program that performs reasoning is called *reasoner*.

There are different types of reasoning tasks, but many of them can be reduced to the basic problem of TBox and RBox consistency checking. Common reasoning tasks are:

- Check if a given axiom is the consequence of a given ontology.
- Find out all pairs of atomic concepts (A, B) of an ontology with $A \sqsubseteq B$. This is called classification.
- Check the consistency of a given ontology.

A consistency checker gets N_C, N_R, a TBox and an RBox and then tells if the given ontology is consistent.

Many reasoners accept the input ontologies in the OWL format [4] which can be considered as an extension to $\mathcal{ALCHIQ}$. This very handy because any OWL reasoner can be used as a consistency checker for $\mathcal{ALCHIQ}$, which will be exploited later.

2.7 Outline of the Thesis

One can use the introduced techniques for modelling different kind of scenarios in a formal way. Many constructs we know from natural speech have an operator in $\mathcal{ALCHIQ}$. But there are many concepts that can be described in prose with an self-referential phrase, which can not be adapted to $\mathcal{ALCHIQ}$. In the beginning a *grandparent* was defined by this wording:

> A grandparent is someone who is the parent of someone who is the parent of somebody else.

[1] We do not consider ABox axioms here. When taking an ABox into account, $\mathcal{ALCHIQ}$ has the weaker forest model property only.

So a grandparent is not just a parent of any parent but the parent of someone who is the parent of "somebody else". For example the graph given by Figure 2.1b does not fulfil the "somebody else" part, as there only is one individual who is both father and grandfather of himself. To prohibit interpretations like that, we need a formal syntax to model such concepts.

This can be achieved by using the I-me-construct by Marx [5], which is described in detail in Chapter 3 as an extension to $\mathcal{ALCQ}$. With that it is possible to remember a certain point in a formula, go to other points along roles and then refer to the originally remembered point.

As allowing such a general construct leads to undecidability regarding consistency checking, the investigations after that restrict the usage of the new construct: $\mathcal{ALCQ}\mathsf{me}_2$ is obtained by only allowing the remembered and referred point to have a distance of at most two hops via qualified number restrictions. Even with this restriction it is seen that $\mathcal{ALCQ}\mathsf{me}_2$ still is expressive enough to encode role inverses and role hierarchies as they are known from $\mathcal{ALCHIQ}$.

On the other hand, it is shown in Chapter 4, that the TBox consistency checking problem for $\mathcal{ALCQ}\mathsf{me}_2$ can be polynomially reduced to $\mathcal{ALCHIQ}$ by exploiting something similar to the tree model property. As there are consistency checkers for $\mathcal{ALCHIQ}$, this gives an consistency checker for $\mathcal{ALCQ}\mathsf{me}_2$ implicitly.

The practical point of view is stressed afterwards in Chapter 5. The widespread ontology format OWL is used to effectively encode an $\mathcal{ALCHIQ}\mathsf{me}_2$ TBox and RBox. The implementation of the reduction can be used to convert such an $\mathcal{ALCHIQ}\mathsf{me}_2$ ontology to a plain OWL ontology whose satisfiability then can be checked with any OWL reasoner. Benchmarking the conversion shows that independent from the increasing file size, the blowup regarding consistency checking time turns out to be nearly unchanged.

Finally, the thesis is concluded by summarizing the main results.

3 Self-Reference in Description Logics

The aspect of self-reference is implicitly used very often in natural language however, is immediately not expressible with the description logics considered previously. In $\mathcal{ALCHIQ}$ it is impossible to define the concept of a narcissist:

> A narcissist is a person who loves herself.[1]

So in the model given by Figure 3.1, the informally defined concept of a narcissist holds exactly at the individuals marked by N.

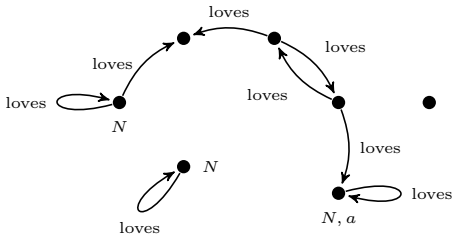


Figure 3.1: A model containing some narcissists and an individual a.

It can be tested manually where N holds and where it does not. When validating a concept formula C at a certain point in a graph more or less two different types of checks are done:

- If C is composed by Boolean operators $\sqcap$, $\sqcup$, $\neg$, it is checked if the subformulas hold and thus it is known whether C holds at the current point.
- If C is of the kind $\exists R.D$, we have to look for a R successor where the validation of D succeeds. Analogously for $\forall R.D$, we have to validate D at each R successor.

In the latter case, the current point of evaluation changes because of following the R-link between to points. In common description logics it is not possible to reference the original point after following the link. Even when having role inverses, we know, the original point is reachable by an R^- link, but not which link it is. But in the extension of $\mathcal{ALC}$ given by Marx [5] this is enabled by adding new constructs: the current point of evaluation is remembered by an I before following a link with $\exists$,$\forall$ and is then referable by a me. In other words, me behaves like a concept that holds only at the point where the I was evaluated. Using I and me, narcissists can be defined by the concept:

$$N \equiv \mathsf{I}.\exists \mathsf{loves}.\mathsf{me}$$

[1] Not implying that a narcissist is necessarily female.

N holds exactly at those points fulfilling the condition: there is a loves-link to the point itself.

The difference with $\mathcal{ALC}$ is: there is a new concept me from which you can not tell if it holds at a certain point or not, because it depends on the context. It does not make sense to ask, where the concept me holds in the model in Figure 3.1. But if the remembered point by I is e.g. a, then me definitely only holds at a, i.e. me in this case is interpreted as the set $\{a\}$. If another individual is remembered by the I, the interpretation changes completely. Intuitively $\mathsf{I}.\mathsf{me}$ holds everywhere but $\mathsf{I}.\neg\mathsf{me}$ nowhere.

A more complex example is the concept of a sibling – a person whose parent has another child. Assume the roles p and c behave naturally as the roles "parent of" and "child of", i.e. assume p is the inverse of c and define a sibling as:

$$S \equiv \mathsf{I}.\exists c.\exists p.\neg\mathsf{me} \equiv \mathsf{I}.\exists p^{-}.\exists p.\neg\mathsf{me}$$

S holds at those points from which a path first along $c \equiv p^{-}$, then along p exists that does not lead to the original point. An example for the sibling concept is given by Figure 3.2, where individual 2 is a sibling as its parent 1 has another child 3. So individual 3 also is a sibling, whereas the topmost individual 1 does not have a parent and so is not a sibling at all. The only child of the parent of individual 4 is 4, so it is not a sibling.

This description even matches the intuition concerning its opposite: the contrary concept of a sibling – an only child O – can be written as

$$O :\equiv \neg S \equiv \neg\mathsf{I}.\exists c.\exists p.\neg\mathsf{me} \equiv \mathsf{I}.\forall c.\forall p.\neg\neg\mathsf{me} \equiv \mathsf{I}.\forall c.\forall p.\mathsf{me},$$

a person whose each parent's each child is the person herself. In Figure 3.2 individual 4 satisfies O, because the only child of 2 is individual 4. But also individual 1 satisfies O even though it is not anyone's child.

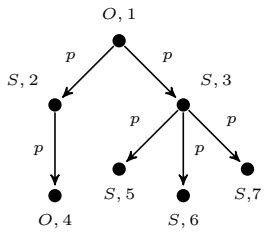


Figure 3.2: Example containing the sibling concept S and only children O

3.1 $\mathcal{ALCQ}$ with Self-Reference

Now the formal description of an extension of $\mathcal{ALCQ}$ which allows self-reference is given. Originally, Marx [5] extended $\mathcal{ALC}$. The approach here is the straightforward adaption to $\mathcal{ALCQ}$.

Definition 3.1 (Syntax of $\mathcal{ALCQ}\mathsf{me}$)**.** Based on $\mathcal{ALCQ}$, the set of concept expressions $\mathfrak{C}$ is extended as follows:

$$\mathfrak{C} \ni N, \mathsf{me}, \neg C, C \sqcap D, \mathsf{I}.C, \geq_n R.D \qquad N \in N_C, C, D \in \mathfrak{C}, R \in \mathfrak{R}, n \in \mathbb{N}_0$$

Like in $\mathcal{ALCQ}$, the operators $\exists$, $\forall$, $\sqcup$, etc. are implicitly defined in the usual way.

As we already examined, an occurrence of me without the guard of a I does not have unique meaning. Hence we do not want this to happen in a TBox axiom, which can by formalized by partitioning concept expressions into open and closed concepts. $C \in \mathfrak{C}$ is called *open* if there is a me which is not under the scope of an I; such a me is named *free*. Otherwise the concept C is called *closed*. In detail, $\text{open}, \text{closed}: \mathfrak{C} \to 2$, with $\text{open}(C) = 1 - \text{closed}(C)$, is defined as follows:

$$\begin{aligned} \text{closed}(N) &= 1 \text{ for } N \in N_C, & \text{closed}(\mathsf{I}.C) &= 1, \\ \text{closed}(\mathsf{me}) &= 0, & \text{closed}(\neg C) &= \text{closed}(C), \\ \text{closed}(C \sqcap D) &= \text{closed}(C) \wedge \text{closed}(D), & \text{closed}(\geq_n R.C) &= \text{closed}(C). \end{aligned}$$

The set of well-formed TBox axioms contains axioms of the kind

$$C \sqsubseteq D,$$

where C and D are *closed* concepts. Also axioms of the form $C \equiv D$ can be expressed implicitly by stating $C \sqsubseteq D$ and $D \sqsubseteq C$. ■

As already discussed, I "remembers" the current point of evaluation in $\mathsf{I}.C$ so that each me in C under the scope of this I denotes a concept which only contains the remembered point. This means that the memorized point needs to be explicitly handled in the semantics of $\mathcal{ALCQ}\mathsf{me}_2$.

Definition 3.2 (Semantics of $\mathcal{ALCQ}\mathsf{me}$)**.** For every $a \in \Delta^{\mathcal{I}}$ and $C \in \mathfrak{C}$ the expression $C_a^{\mathcal{I}}$ denotes the interpretation of C where a is the remembered point:

$$\begin{aligned} \bot_a^{\mathcal{I}} &= \emptyset, & N_a^{\mathcal{I}} &= N^{\mathcal{I}}, & (\mathsf{I}.C)_a^{\mathcal{I}} &= \{b \in \Delta^{\mathcal{I}} \mid b \in C_b^{\mathcal{I}}\}, \\ (C \sqcap D)_a^{\mathcal{I}} &= C_a^{\mathcal{I}} \cap D_a^{\mathcal{I}}, & (\neg C)_a^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C_a^{\mathcal{I}}, & \mathsf{me}_a^{\mathcal{I}} &= \{a\}, \\ (\geq_n R.C)_a^{\mathcal{I}} &= \{b \in \Delta^{\mathcal{I}} \mid \#\{(b,c) \in R^{\mathcal{I}} : c \in C_a^{\mathcal{I}}\} \geq n\} \end{aligned}$$

for all $N \in N_C, D \in \mathfrak{C}', R \in \mathfrak{R}$. The interpretation for TBox axioms is similar to $\mathcal{ALC}$ and given by Table 3.1. ■

Syntax	Semantics
$C \sqsubseteq D$	$C_a^{\mathcal{I}} \subseteq D_b^{\mathcal{I}}$
with $C, D \in \mathfrak{C}, a, b \in \Delta^{\mathcal{I}}$	

Table 3.1

Example 3.3. The behaviour of the remembered state is visualized by the example given in Figure 3.3. For the concept $D :\equiv \mathsf{I}.\exists R.\exists R.\mathsf{me}$ the interpretation only consists of $\{a', b'\}$, which is shown by:

$$\begin{aligned} \mathsf{me}^{\mathcal{I}}_{a'} &= \{a'\} \\ (\exists R.\mathsf{me})^{\mathcal{I}}_{a'} &= \{x \in \Delta^{\mathcal{I}} \mid \text{exists } (x,y) \in R^{\mathcal{I}} \text{ with } y \in \mathsf{me}^{\mathcal{I}}_{a'} = \{a'\}\} \overset{(b',a') \in R^{\mathcal{I}}}{=} \{b'\} \\ (\exists R.\exists R.\mathsf{me})^{\mathcal{I}}_{a'} &= \{x \in \Delta^{\mathcal{I}} \mid \text{exists } (x,y) \in R^{\mathcal{I}} \text{ with } y \in (\exists R.\mathsf{me})^{\mathcal{I}}_{a'} = \{b'\}\} = \{a'\} \end{aligned}$$

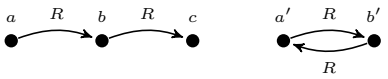


Figure 3.3

So $a' \in (\exists R.\exists R.\mathsf{me})^{\mathcal{I}}_{a'}$ implies $a' \in (\mathsf{I}.\exists R.\exists R.\mathsf{me})^{\mathcal{I}}_{y}$ for *any* $y \in \Delta^{\mathcal{I}}$; analogously we can conclude $b' \in D^{\mathcal{I}}_{y}$. The only points where D holds are a' and b', because any other point $p \in \Delta^{\mathcal{I}} \setminus \{a', b'\}$ can not be contained in $(\exists R.\exists R.\mathsf{me})^{\mathcal{I}}_{p}$, e.g.:

$$\begin{aligned} a &\notin (\exists R.\exists R.\mathsf{me})^{\mathcal{I}}_{a} = \emptyset \\ c &\notin (\exists R.\exists R.\mathsf{me})^{\mathcal{I}}_{c} = \{a\} \end{aligned}$$

▲

Remark. The semantics of "$\sqsubseteq$" is well-defined because for $C \sqsubseteq D$ the concepts C, D must be closed and it holds $E^{\mathcal{I}}_{a} = E^{\mathcal{I}}_{b}$ for any closed concept E and any $a, b \in \Delta^{\mathcal{I}}$.

$E^{\mathcal{I}}_{a} = E^{\mathcal{I}}_{b}$ follows recursively from the structure of complex concepts. Trivially it is $(\mathsf{I}.C)^{\mathcal{I}}_{a} = \{c \in \Delta^{\mathcal{I}} \mid c \in C^{\mathcal{I}}_{c}\} = (\mathsf{I}.C)^{\mathcal{I}}_{b}$ and $N^{\mathcal{I}}_{a} = N = N^{\mathcal{I}}_{b}$ for all $C \in \mathfrak{C}$, $N \in N_C$. These are the base-cases for closed concepts. As the remembered points a, b are disregarded and left unmodified by the other constructs, it follows $E^{\mathcal{I}}_{a} = E^{\mathcal{I}}_{b}$ for any closed concept E.

Analogous to the consistency checking problem in Section 2.6, the TBox consistency checking problem for $\mathcal{ALCQ}\mathsf{me}$ is defined as follows:

> Given an $\mathcal{ALCQ}\mathsf{me}$ TBox $\mathcal{T}$. Is there any model for $\mathcal{T}$?

In contrast to the previously discussed decidable description logic $\mathcal{ALCHIQ}$, this problem is undecidable for $\mathcal{ALC}\mathsf{me}$ [5]. $\mathcal{ALC}\mathsf{me}$ equals $\mathcal{ALCQ}\mathsf{me}$ without the extension quantified number restrictions, so the undecidability also follows for the strictly more expressive $\mathcal{ALCQ}\mathsf{me}$.

It turned out that one important criterion for decidability is the maximum number of qualified number restrictions between a me and its corresponding I. For example this distance for

$$\mathsf{I}.\exists R.\exists S.\forall T.\mathsf{me}$$

is 3. But this distance is 2 for the concepts

$$\mathsf{I}.\forall R.\exists S.\mathsf{I}.\forall T.\mathsf{me}, \quad \mathsf{I}.\forall R.\exists S.\mathsf{me}, \quad \mathsf{I}.\forall R.\exists S.(\mathsf{me} \sqcap \forall T.\top).$$

Let $\mathcal{ALLC}\mathsf{me}_d$, $d \in \mathbb{N}_0$, be the fragment of $\mathcal{ALLC}\mathsf{me}$ with the maximum distance of at most d in any concept expression. Initially it was proved that the problem is undecidable for $\mathcal{ALLC}\mathsf{me}_8$ [5]. Then this was lowered to $d = 4$, i.e. it is undecidable for $\mathcal{ALLC}\mathsf{me}_4$ [6]. Later it was shown that $\mathcal{ALLC}\mathsf{me}_3$ is undecidable, and hence $\mathcal{ALLCQ}\mathsf{me}_3$, too [7].

Theorem 3.4. Considering $\mathcal{ALLC}\mathsf{me}_d$ for $d = 0$, the decidable description logic $\mathcal{ALLCQ}$ is obtained using the identities

$$\mathsf{I}.\mathsf{me} \equiv \top, \quad \mathsf{I}.(\neg C) \equiv \neg\mathsf{I}.C, \quad \mathsf{I}.(C \sqcap D) \equiv \mathsf{I}.C \sqcap \mathsf{I}.D, \quad \mathsf{I}.N \equiv N \text{ for } N \in N_C.$$

These identities directly result from the semantics definition of $\mathcal{ALLCQ}\mathsf{me}$. □

The question for the maximum $0 \leq d < 3$ that keeps $\mathcal{ALLCQ}\mathsf{me}_d$ decidable concerning consistency checking finally was turned out to be $d = 2$: the decidability of the consistency checking problem for $\mathcal{ALLCQ}\mathsf{me}_2$ – implicitly also for $\mathcal{ALLCQ}\mathsf{me}_1$ – can be shown by a reduction to $\mathcal{ALLCHIQ}$ [8, 9].

So in the following we will focus on $\mathcal{ALLCQ}\mathsf{me}_2$, will look at its expressiveness in comparison to $\mathcal{ALLCHIQ}$, starting with the normalization of $\mathcal{ALLCQ}\mathsf{me}_2$ concept formulas.

Definition 3.5. An $\mathcal{ALLCQ}\mathsf{me}_2$ formula is considered normalized if it is in NNF and the I is as close to the atomic concepts as possible. I.e. negations occur only in front of atomic concepts – analogous to Definition 2.19 with treating me as an atomic concept. A concept C is put in NNF as described in Definition 2.19 with the additional rule that both $\neg$ and NNF commute with I:

$$\mathrm{NNF}(\mathsf{I}.C) = \mathsf{I}.\mathrm{NNF}(C), \quad \mathrm{NNF}(\neg\mathsf{I}.C) = \mathsf{I}.\mathrm{NNF}(\neg C).$$

Pushing the I as close to the me is achieved by the following mapping:

$$\begin{aligned} \mathsf{I}.\mathsf{me} &\mapsto \top, & \mathsf{I}.(C \star D) &\mapsto \mathsf{I}.C \star \mathsf{I}.D, & \star &\in \{\sqcap, \sqcup\}, \\ \mathsf{I}.(\neg C) &\mapsto \neg\mathsf{I}.C, & \mathsf{I}.\mathsf{I}.C &\mapsto \mathsf{I}.C, & & \\ \mathsf{I}.E &\mapsto E, & & & C, D &\in \mathfrak{C}, E \in N_C. \end{aligned}$$

No mapping changes the concept semantics, but after applying these rules, a concept in NNF is obtained. Furthermore, for each me in the normalized concept, the distance to its related I is still at most 2, but there is no pair of I and me with distance 0, because each I appears directly in front of a qualified number restriction. ■

3.2 Role Inverses in $\mathcal{ALLCQ}\mathsf{me}_2$

In $\mathcal{ALLCQ}\mathsf{me}_2$ no role inverses are allowed to be used explicitly. But it is possible to make two roles interact as inverses to each other. Let $R, S \in N_R$ be two roles that shall

be marked as inverses to each other. Then consider the following TBox axiom:

$$\top \equiv \mathsf{I}.\forall R.\exists S.\mathsf{me}$$

It states that for each R link, there always is an S link back, i.e. for any interpretation $\mathcal{I}$ it is:

$$R^{\mathcal{I}} \subseteq (S^{-})^{\mathcal{I}}$$

Considering the axiom with the roles R and S swapped, you get the TBox Axiom

$$\top \equiv \mathsf{I}.\forall S.\exists R.\mathsf{me},$$

which supplementary postulates:

$$S^{\mathcal{I}} \subseteq (R^{-})^{\mathcal{I}},$$

which is equivalent to $(S^{-})^{\mathcal{I}} \subseteq R^{\mathcal{I}}$ because of the semantics of role inverses.

So both axioms together enforce $R^{\mathcal{I}} = (S^{-})^{\mathcal{I}}$. This directly gives a consistency preserving reduction from $\mathcal{ALCIQ}$ to $\mathcal{ALCQ}\mathsf{me}_2$: for each role $R \in N_R$ introduce a new role $S_R \in N'_R$, add the TBox axiom

$$\top \equiv \mathsf{I}.\forall R.\exists S_R.\mathsf{me} \sqcap \mathsf{I}.\forall S_R.\exists R.\mathsf{me}$$

and replace very occurrence of R^{-} in the original TBox by S_R [8].

3.3 Role Hierarchies in $\mathcal{ALCQ}\mathsf{me}_2$

The encoding of role hierarchies in $\mathcal{ALCQ}\mathsf{me}_2$ is very similar. Any RBox axiom of the form

$$V \sqsubseteq W$$

can be expressed as a single TBox axiom in $\mathcal{ALCQ}\mathsf{me}_2$ assuming that role inverses already are encoded as described before. The semantics of "$\sqsubseteq$" requires that for every interpretation $\mathcal{I}$ it holds:

$$V^{\mathcal{I}} \subseteq W^{\mathcal{I}} = (S_W^{-})^{\mathcal{I}},$$

where S_W is the inverse role of W as encoded in the section before. Now the same trick as for the inverses can be done. Put an TBox axiom for each RBox axiom $V \sqsubseteq W$:

$$\mathsf{I}.\forall V.\exists S_W.\mathsf{me}$$

It says: if there is a V-successor then there is an S_W-link back to the original node. I.e. if there is an V-link between two nodes, then there is also a W-link. So $\mathcal{ALCHQ}\mathsf{me}_2$ – and also $\mathcal{ALCHIQ}$ – can be reduced to $\mathcal{ALCQ}\mathsf{me}_2$ [8].

Example 3.6. Consider the following definition of a barber, which may be known from other fields of mathematics in a similar way.

> A barber is someone who shaves those that do not shave themselves. Everyone is shaved – by himself or by a barber.

This description can be adapted directly to an $\mathcal{ALCQ}\mathsf{me}_2$ TBox. Let B be the concept of a barber, S the role standing for "shaves" and R standing for "shaved by". We first have to let R and S as inverse roles:

$$\top \equiv \mathsf{I}.\forall S.\exists R.\mathsf{me}, \quad \top \equiv \mathsf{I}.\forall R.\exists S.\mathsf{me}$$

A barber should only shave people that do not shave themselves and secondly, everybody is shaved by exactly one person, by himself or by a barber:

$$\begin{aligned} B &\equiv \ \forall S.\neg\mathsf{I}.\exists S.\mathsf{me} \\ \top &\equiv \geq_1 R.\top \sqcap \ \leq_1 R.\top \\ \top &\equiv \ \mathsf{I}.\forall R.(\mathsf{me} \sqcup B) \end{aligned}$$

Given the $\mathcal{ALCQ}\mathsf{me}_2$ TBox $\mathcal{T}$ with these axioms. Is the concept B satisfiable, i.e. is there a model $\mathcal{I}$ for $\mathcal{T}$ with $B^{\mathcal{I}} \neq \emptyset$? The answer is yes, because there may be two barbers shaving each other as it is illustrated by Figure 3.4. ▲

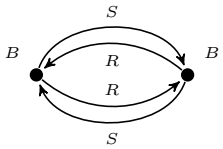


Figure 3.4: A minimal model for the barber TBox

Example 3.7. In other fields, the original barber example normally is known as a contradictory example. Consider the previous TBox with the following axiom added:

$$\bot \equiv B \sqcap \mathsf{I}.\exists R.(B \sqcap \neg\mathsf{me})$$

This new TBox $\mathcal{T}'$ explicitly forbids the model in Figure 3.4 by saying: there is no barber that is shaved by a barber that is not himself. So the consideration whether the barber shaves himself or not leads to the contradiction making the concept unsatisfiable. This means it is $B^{\mathcal{I}} = \emptyset$ for any model $\mathcal{I}$. ▲

4 Transforming $\mathcal{ALCQ}\mathsf{me}_2$ to $\mathcal{ALCHIQ}$

In this chapter a transformation is provided that maps an $\mathcal{ALCQ}\mathsf{me}_2$ TBox $\mathcal{T}$ to an $\mathcal{ALCHIQ}$ TBox $\mathcal{T}'$ with the property: $\mathcal{T}$ is consistent iff $\mathcal{T}'$ is consistent. The $\mathcal{ALCHIQ}$ TBox $\mathcal{T}'$ will be such that it only accepts tree models. With this transformation, a consistency checker for $\mathcal{ALCHIQ}$ is given implicitly.

In this chapter we assume that $\forall$ and $\exists$ are expressed using $\geq$ and $\leq$, all formulas are in NNF and each I is being pushed as close as possible to each me under its scope, according to Definition 3.5. Since the encoding is complex, we start with the simpler case of encoding $\mathcal{ALCHIQ}$ in $\mathcal{ALCHIQ}$ itself.

4.1 Encoding $\mathcal{ALCHIQ}$ in itself

As an introduction to the idea of reduction of TBoxes, this section will provide a reduction of $\mathcal{ALCHIQ}$ to $\mathcal{ALCHIQ}$ itself. Knowing that reducing a problem to itself is trivial in general, this reduction only is given to show the basic reduction steps, which will be extended for the reduction of $\mathcal{ALCQ}\mathsf{me}_2$ later.

4.1.1 Subformula Construction

To be able to encode the $\mathcal{ALCQ}\mathsf{me}_2$ axioms later, the basic idea is to add new concepts for every subformula contained in a complex concept. Then the relationship between these concepts are expressed with $\mathcal{ALCHIQ}$ axioms to simulate the behaviour of the original concepts. This kind of transformation is now described for an $\mathcal{ALCHIQ}$ TBox, which operates on $\mathcal{ALCHIQ}$ subformulas. From now on we assume all concept formulas to be in NNF. So for $C \in \mathfrak{C}$, its negation $\neg C$ stands for the negation in NNF, i.e. for $\mathrm{NNF}(\neg C)$.

Definition 4.1. For a set Σ of $\mathcal{ALCHIQ}$ concepts, the *closure* $Cl(\Sigma)$ is the smallest set extending Σ, consisting concepts that is closed under negation and subformula construction. $\mathrm{Cl}(\Sigma)$ is constructed explicitly by

$$\mathrm{Cl}(\Sigma) := \bigcup_{\sigma \in \Sigma} \{x \mid x \in \mathrm{Sub}(\sigma)\} \cup \{\neg x \mid x \in \mathrm{Sub}(\sigma)\},$$

where Sub corresponds to the set of subformulas for a given concept, as defined by Definition 2.20. For a TBox $\mathcal{T}$, $\mathrm{Cl}(\mathcal{T})$ is the closure of the set of concepts occurring in $\mathcal{T}$. ■

The subformula reduction of an $\mathcal{ALCHIQ}$ TBox $\mathcal{T}$ constructs a new TBox $\mathcal{T}'$ from scratch, containing atomic concepts for each subformula from $\mathcal{T}$,

$$N'_C := \{A_D \mid D \in \mathrm{Cl}(\mathcal{T})\},$$

while keeping the role names unchanged: $N'_R = N_R$. For every axiom $C \sqsubseteq D$ in $\mathcal{T}$ the corresponding axiom added to $\mathcal{T}'$:

$$A_C \sqsubseteq A_D$$

Of course we have to add more axioms to $\mathcal{T}'$, to ensure that former complex concepts behave like their name suggests – $A_{C \sqcup D}$ shall interact with A_C and A_D just like $C \sqcup D$ did with C and D. In general we want a one-to-one correspondence of the models for $\mathcal{T}$ and for $\mathcal{T}'$:

1. If $\mathcal{I}$ is a model for $\mathcal{T}$, then there is a model $\mathcal{I}'$ for $\mathcal{T}'$ with

$$\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}} \tag{4.1}$$
$$R^{\mathcal{I}'} = R^{\mathcal{I}} \qquad \text{for each } R \in N'_R = N_R \tag{4.2}$$
$$A_C^{\mathcal{I}'} = C^{\mathcal{I}} \qquad \text{for each } C \in N'_C \tag{4.3}$$

2. If $\mathcal{I}'$ is a model for $\mathcal{T}'$, then there is a model $\mathcal{I}$ for $\mathcal{T}$ such that (4.1), (4.2) and (4.3) are fulfilled.

To get both directions met, the A_C concepts have to be given a meaning based on the structure of C. Effectively the semantics definition of $\mathcal{ALCHIQ}$ is encoded again for every single concept $A_C \in N'_C$ by adding the following axioms to $\mathcal{T}'$:

$$A_\top \equiv \top, \qquad A_\bot \equiv \bot, \qquad \neg A_C \equiv A_{\neg C}$$
$$A_{C \sqcup D} \equiv A_C \sqcup A_D \qquad A_{C \sqcap D} \equiv A_C \sqcap A_D$$
$$A_{\geq_n R.C} \equiv \geq_n R.A_C \qquad A_{\leq_n R.C} \equiv \leq_n R.A_C$$

All equations are the intuitive adaption of the semantics of the operators. So there is a one-to-one correspondence between models for $\mathcal{T}$ and $\mathcal{T}'$. In particular $\mathcal{T}'$ is consistent iff $\mathcal{T}$ is consistent.

Example 4.2. Consider $N_C = \{C, D\}$, $N_R = \{R\}$ and the TBox $\mathcal{T}$ consisting of one axiom:

$$D \sqsubseteq \exists R.(C \sqcup D)$$

Then N'_C contains these formulas:

$$\begin{array}{ccccc} A_\top & A_C & A_D & A_{C \sqcup D} & A_{\exists R.(C \sqcup D)} \\ A_\bot & A_{\neg C} & A_{\neg D} & A_{\neg C \sqcap \neg D} & A_{\forall R.(\neg C \sqcap \neg D)} \end{array}$$

The transformed TBox $\mathcal{T}'$ has the following axioms. For every pair of axioms $\neg A_C \equiv A_{\neg C}$ and $\neg A_{\neg C} \equiv A_C$, one of them is omitted because it is apparently redundant.

$$
\begin{array}{ccc}
A_D \sqsubseteq A_{\exists R.(C \sqcup D)} & A_\top \equiv \top & A_\bot \equiv \bot \\
A_{C \sqcup D} \equiv A_C \sqcup A_D & A_{\neg C \sqcap \neg D} \equiv A_{\neg C} \sqcap A_{\neg D} & \neg A_C \equiv A_{\neg C} \\
A_{\exists R.(C \sqcup D)} \equiv \exists R.(A_C \sqcup A_D) & A_{\forall R.(\neg C \sqcap \neg D)} \equiv \forall R.(A_{\neg C} \sqcap A_{\neg D}) & \neg A_D \equiv A_{\neg D} \\
\neg A_{C \sqcup D} \equiv A_{\neg C \sqcap \neg D} & \neg A_{\exists R.(C \sqcup D)} \equiv A_{\forall R.(\neg C \sqcap \neg D)} &
\end{array}
$$

▲

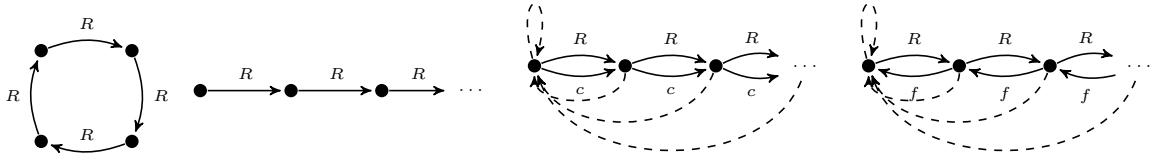
4.1.2 Tree Model Property

Now the tree model property of $\mathcal{ALCHIQ}$ is exploited; it is a property which is featured by many decidable description logics and which lets them stay decidable under many extensions [10]. As already stated along with Definition 2.22, $\mathcal{ALCHIQ}$ has this property. For the encoding of $\mathcal{ALCQ}\mathsf{me}_2$ later, we will heavily use some similar property of $\mathcal{ALCQ}\mathsf{me}_2$.

Example 4.3. Consider an $\mathcal{ALCHIQ}$ TBox $\mathcal{T}$ consisting of one axiom:

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

$\mathcal{T}$ is satisfiable; an example model is given in Figure 4.1a, which is not a tree. The axiom says: each point has at least one R-successor. So there can not be a finite tree shaped model for $\mathcal{T}$, but there are some infinite models, e.g. the natural numbers $\Delta^\mathcal{I} = \mathbb{N}$ with the ordinary successor relation: $R^\mathcal{I} = \{(x, x+1) \mid x \in \mathbb{N}\}$. This infinite tree with root 1 is indicated by Figure 4.1b. ▲



(a) (b) (c) The dashed relation describes *root* (d) Replacing c by f^- in Figure 4.1c

Figure 4.1: Different models for $\top \sqsubseteq \exists R.\top$

Because of the tree model property, we can enforce an $\mathcal{ALCHIQ}$ TBox to have only tree shaped models, without changing the consistency of the TBox. This is done by adding an additional role c, which is intended to point from a node to its children, i.e. $(x, y) \in c^\mathcal{I}$ iff y is a child of x. As requested by Definition 2.4, c^- has to be functional, which means that each point must be the child of at most one node. This is achieved with an additional TBox axiom:

$$\top \sqsubseteq \leq_1 c^-.\top$$

The globalized relation $\mathcal{I}$, given by Definition 2.21, of each model has to be forced to be a tree. This is done by only allowing the existing roles to connect nodes with their children in the tree. To achieve this formally, an RBox needs to be introduced, to which the following axiom is added for each role $r \in N_R$:

$$r \sqsubseteq c$$

To give that tree a root, the concept $\neg\exists c^-.\top$ has to hold at some point in the model. The typical approach for enforcing a concept to not be empty, is to introduce another role *root* and add another TBox axiom:

$$\top \sqsubseteq \exists \text{root}.\neg\exists c^-.\top$$

Strictly speaking, once *root* is added, the globalized relation $\mathcal{I}$ is not tree-shaped anymore. But ignoring *root*, the remaining relation $c^{\mathcal{I}}$ is a forest, i.e. a collection of trees, each with a root fulfilling $\neg\exists c^-.\top$ and $\exists \text{root}^-.\top$. A model for the transformed version of Example 4.3 is given by Figure 4.1c. All in all we can enforce an $\mathcal{ALCHIQ}$ TBox to have only tree shaped models – ignoring *root* regarding the globalized relation – by setting the role names to

$$N_R' := N_R \dot\cup \{c, \text{root}\}$$

and extending the TBox as well as the RBox as described above.

If a tree-shaped model exists for the original TBox, then there is also a model for the new TBox with the new role root pointing to each tree's root. On the other hand, an inconsistent TBox still is inconsistent after this transformation, because we only added more and more axioms. So the transformation is consistency preserving.

To improve the readability of the formulas later we will replace the role pointing from a node to all its child nodes by the inverse role f, which points to each node's father:

$$f := c^-, \quad f^- := c$$

As this can be grasped as just renaming roles, the resulting transformation is still consistency preserving. Deploying the entire transformation on the TBox given by Example 4.3 leads to a TBox with the example model in Figure 4.1d.

4.2 Encoding $\mathcal{ALCQ}\mathsf{me}_2$ in $\mathcal{ALCHIQ}$

The logic $\mathcal{ALCQ}\mathsf{me}_2$ does not have the tree model property. This can be seen, because both the $\mathcal{ALCQ}\mathsf{me}_2$ axioms

$$\top \sqsubseteq \mathsf{I}.\exists R.\mathsf{me} \text{ and } \top \sqsubseteq \mathsf{I}.\exists R.(\neg\mathsf{me} \sqcap \exists R.\mathsf{me})$$

are satisfiable but do not have a tree model. Thus $\mathcal{ALCQ}\mathsf{me}_2$ can not have the tree model property. But a slightly weaker criterion of a quasi-tree model property can be found for $\mathcal{ALCQ}\mathsf{me}_2$.

Definition 4.4. A relation $S \subseteq X^2$ is a quasi-tree with root r if there is a tree $R \subseteq X^2$ with root r and $R \subseteq S \subseteq R \cup R^{-1} \cup \mathrm{Id}_X$.

$\mathcal{I}$ is a *quasi-tree model* with root r if its globalized relation is a quasi-tree with root r. ■

Informally speaking, a quasi-tree is a tree extended with self-loops and links to the father, like the example given by Figure 4.2.

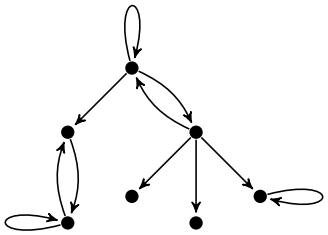


Figure 4.2: An example quasi-tree

Theorem 4.5 (Gorín and Schröder [8])**.** If C is a satisfiable closed $\mathcal{ALCQ}\mathsf{me}_2$ concept, then there exists a quasi-tree model that satisfies C at its root. □

Let $\mathcal{T}$ be an $\mathcal{ALCQ}\mathsf{me}_2$ TBox to be checked for satisfiability. We know that if $\mathcal{T}$ is consistent then there must be a quasi-tree model for $\mathcal{T}$. This allows a transformation to an $\mathcal{ALCHIQ}$ TBox $\mathcal{T}'$ following the same ideas as in Section 4.1 combined with the additional effort needed to encode the quasi-tree:

- Force the model to be a tree
- Encode self-loops of R as new concepts $\circlearrowleft_R$
- Encode uplinks of R as new concepts $\uparrow_R$
- Encode the behaviour of $\geq, \leq, \sqcup, \sqcap, \neg$ as axioms in $\mathcal{ALCHIQ}$

With the concepts $\circlearrowleft_R$ and $\uparrow_R$, the quasi-tree is reduced to a simple tree, which allows us to adapt the transformation from Subsection 4.1.2 to $\mathcal{ALCQ}\mathsf{me}_2$. Knowing that the model always is a tree, we can encode the $\mathcal{ALCQ}\mathsf{me}_2$ semantics in $\mathcal{ALCHIQ}$ using the subformula construction similar to Subsection 4.1.1. The subformulas of an $\mathcal{ALCQ}\mathsf{me}_2$ concept are defined as follows:

Definition 4.6. For concepts C and D, $C(D)$ is the concept obtained by replacing every free occurrence of me outside the scope of a qualified number restriction in C by D. ■

E.g. $(\mathsf{me} \sqcap \leq_4 R.\mathsf{me})(D)$ is $D \sqcap \leq_4 R.\mathsf{me}$ because the second me is under the scope of "$\leq_4$".

Definition 4.7. For a set Σ of closed $\mathcal{ALCQ}\mathsf{me}_2$ concepts, the *closure* $Cl(\Sigma)$ is the smallest set extending Σ consisting of concepts, normalized according to Definition 3.5,

that is closed under negation, subformulas and – in addition to $\mathcal{ALCHIQ}$ – pseudo-subformulas; i.e., if $C \in \mathrm{Cl}(\Sigma)$ and D is a subformula of C, then $D(\top)$ and $D(\bot)$ are in $\mathrm{Cl}(\Sigma)$ too. Aside from the subconcepts specified by $\mathrm{Sub} : \mathfrak{C} \to \wp(\mathfrak{C})$ in Definition 4.1, it is

$$\begin{aligned}
\mathrm{Sub}(\mathsf{I}.\ \geq_n R.C) &= \{\mathsf{I}.\ \geq_n R.C\} \cup \mathrm{Sub}(C(\top)) \cup \mathrm{Sub}(C(\bot)),\\
\mathrm{Sub}(\mathsf{I}.\ \leq_n R.C) &= \{\mathsf{I}.\ \leq_n R.C\} \cup \mathrm{Sub}(C(\top)) \cup \mathrm{Sub}(C(\bot)),\\
\mathrm{Sub}(\geq_n R.C) &= \{\geq_n R.C\} \quad \cup \mathrm{Sub}(C(\top)) \cup \mathrm{Sub}(C(\bot)),\\
\mathrm{Sub}(\leq_n R.C) &= \{\leq_n R.C\} \quad \cup \mathrm{Sub}(C(\top)) \cup \mathrm{Sub}(C(\bot)),
\end{aligned}$$

i.e. $\geq_n R.C$ is *not* a subformula of $\mathsf{I}.\ \geq_n R.C$. Because of that definition, any free occurrence of me in an open concept $C \in \mathrm{Cl}(\Sigma)$ is guarded by an qualified number restriction.

It will be important to distinguish between open and closed subformulas, so $\mathrm{Cl}^*(\Sigma)$ denotes the set of closed concepts in $\mathrm{Cl}(\Sigma)$. For the sake of simplicity Cl and Cl^* denote the closure for the concepts in $\mathcal{T}$. ■

All required definitions for the actual transformation are given now. The result of this encoding of $\mathcal{T}$ defines an $\mathcal{ALCHIQ}$ ontology – with the TBox $\mathcal{T}'$, the RBox $\mathcal{R}'$, the role names N'_R, and the concept names N'_C. Additionally to the roles $R_1, \ldots, R_K$ already in N_R we insert a role $f \in N'_R$, whose interpretation shall behave like the "father of" relation in a tree.

$$\mathcal{T}' \ni \top \sqsubseteq \leq_1 f.\top \tag{4.4}$$

$$\mathcal{R}' = \{R_1^- \sqsubseteq f, \ldots, R_1^- \sqsubseteq f\} \tag{4.5}$$

So each individual in the model has at most one father. Due to (4.5) the roles in N_R are only allowed to connect neighboured nodes within the tree given by f. So the globalized relation still is a tree (i.e. not a true quasi-tree). When doing this limitation e.g. for Figure 4.2 we will lose the self-loops and uplinks and get the tree shown in Figure 4.3.

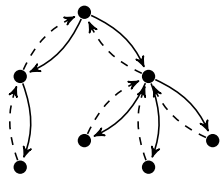


Figure 4.3: Figure 4.2 as a tree without self-loops or uplinks. $f^{\mathcal{I}}$ is the dashed relation.

To be able to encode the semantics of $\mathcal{ALCQ}\mathsf{me}_2$, more concepts are needed to indicate which node stands for me. Because of the quasi-tree model property, there are very few possibilities for the remembered node. So the set of atomic concepts contains all subformulas $\mathrm{Cl}(\mathcal{T})$ annotated with the information of whether there is a node remembered by I and if yes, which one it is:

$$A(\mathcal{T}) = \{A_{\ell:C} \mid \ell \in L, A \in \mathrm{Cl}(\mathcal{T})\}, \quad L = \{*, **, f*, *f\}$$

In the following we will add TBox axioms to enforce the behaviour given by Table 4.1 of the concepts in $A(\mathcal{T})$. E.g. we want $A_{f*:C}$ to hold at a node, iff C holds here when its

father has been remembered by I. So in the end, for any quasi-tree-shaped interpretation $\mathcal{I}$ for $\mathcal{T}$, the contexts should allow an interpretation $\mathcal{J}$ for $\mathcal{T}'$ with $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$, fulfilling:

$$\begin{array}{lll} x \in (A_{*:C})^{\mathcal{J}} & \iff & x \in C^{\mathcal{I}}_z \text{ for any } z \in \Delta^{\mathcal{I}} \\ x \in (A_{**:C})^{\mathcal{J}} & \iff & x \in C^{\mathcal{I}}_x \\ x \in (A_{f*:C})^{\mathcal{J}} & \iff & x \in C^{\mathcal{I}}_y \text{ with } (x,y) \in f^{\mathcal{J}} \\ x \in (A_{*f:C})^{\mathcal{J}} & \iff & y \in C^{\mathcal{I}}_x \text{ with } (x,y) \in f^{\mathcal{J}} \end{array}$$

Note that $(x,y) \in f^{\mathcal{J}}$ says both that x has a father and y is that uniquely determined father. E.g. the last condition says that we want $A_{*f:C}$ to hold at a node, iff remembering this node as I implies that C holds at its father.

ℓ	Remembered node	Node where C would hold in $\mathcal{ALCQ}\mathsf{me}_2$
$*$	does not matter	the closed concept C holds here.
$**$	this node	this node
$f*$	the father node	this node
$*f$	this node	the father node

Table 4.1: Behaviour of subformula concepts $A_{\ell:C} \in A(\mathcal{T})$

We say that $A_{\ell:C}$ describes C in the *context* ℓ. It obviously does not make sense to consider open concepts in the $*$ context, so the set $L(C)$ of possible contexts for a concept C is defined as:

$$L : \mathfrak{C} \to \wp(L), \quad L(C) := \begin{cases} \{*, **, f*, *f\}, & \text{if } C \text{ is closed,} \\ \{\ \ **, f*, *f\}, & \text{if } C \text{ is open.} \end{cases}$$

As a tree has neither self-loops or uplinks to the father, we have to encode the presence or absence of those in extra concepts for each role:

$$B(\mathcal{T}) = \{\circlearrowleft_R |\ R \in N_R\} \cup \{\uparrow_R |\ R \in N_R\}$$

We want $\circlearrowleft_R$ to represent an R-self-loop and $\uparrow_R$ an uplink. The latter is only possible if a node has a father at all, so the following axiom is added to $\mathcal{T}'$ for each $R \in N_R$:

$$\uparrow_R \sqsubseteq \exists R.\top$$

The two sets $A(\mathcal{T})$ and $B(\mathcal{T})$ form the concept names $N'_C = A(\mathcal{T}) \dot\cup B(\mathcal{T})$. Taking the quasi-tree in Figure 4.2 and encoding it as a tree using the concepts in $B(\mathcal{T})$ gives Figure 4.4.

Each TBox axiom $B \sqsubseteq C$ from $\mathcal{T}$ is adapted to the axiom

$$A_{*:B} \sqsubseteq A_{*:C}$$

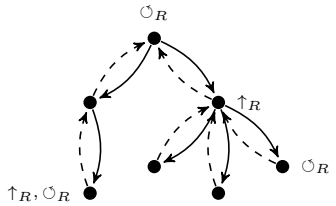


Figure 4.4: Encoding Figure 4.2 as a tree and annotating it with axioms from $B(\mathcal{T})$

in $\mathcal{T}'$. But to ensure that the concepts of the form $A_{*:B}$ behave like the original concept B in $\mathcal{T}$, its semantics is encoded by adding the following axioms to $\mathcal{T}'$ for all concepts $C \in \mathrm{Cl}$.

$$A_{\ell:\bot} \equiv \bot \qquad \text{for } \ell \in L, \tag{4.6}$$
$$A_{\ell:\top} \equiv \top \qquad \text{for } \ell \in \{*, **\}, \tag{4.7}$$
$$\bot \equiv A_{\ell:C} \sqcap A_{\ell:\neg C} \qquad \text{for } \ell \in L(C), \tag{4.8}$$
$$\top \equiv A_{\ell:C} \sqcup A_{\ell:\neg C} \qquad \text{for } \ell \in \{*, **\} \cap L(C), \tag{4.9}$$
$$\exists f.\top \sqsubseteq A_{\ell:C} \sqcup A_{\ell:\neg C} \qquad \text{for } \ell \in \{f*, *f\}. \tag{4.10}$$

It is clear that we want $A_{\ell:\bot}$ to hold nowhere. As there can not be a node in $\mathcal{ALCQ}\mathsf{me}_2$ where both C and $\neg C$, for $C \in \mathrm{Cl}$ hold, we also want $A_{\ell:C}$ and $A_{\ell:\neg C}$ not to hold at the same node. In contrast to that, the positive statements can only be stipulated if the context allows it: the root node of the tree does not have a father, so $A_{*f:C}$ or $A_{f*:C}$ should not hold there for any concept C. So (4.7) and (4.9) do not cover $*f$, $f*$. But at any other node in the tree exactly one of them must hold to ensure that the counting for qualified number restrictions later is done correctly. Therefore (4.10) makes exactly one of $A_{\ell:C}$ and $A_{\ell:\neg C}$ hold if there is a father, i.e. if the node is not the root.

If concepts are composed with $\sqcap$, $\sqcup$, the correct behaviour is enforced by:

$$A_{\ell:C\sqcup D} \sqsubseteq A_{\ell:C} \sqcup A_{\ell:D}, \quad A_{\ell:C\sqcap D} \sqsubseteq A_{\ell:C} \sqcap A_{\ell:D} \tag{4.11}$$

Here, only one direction of the implication – "$\sqsubseteq$" instead of "$\equiv$" – is added, because for contexts $\ell \in \{f*, *f\}$ both $A_{\ell:C\sqcup D}$ and $A_{\ell:\neg(C\sqcup D)}$ do not hold at the root. Anywhere else in the tree, the other direction "$\sqsupseteq$" is implicitly added for the concept $A_{\ell:\neg(C\sqcup D)} = A_{\ell:\neg C\sqcap\neg D}$, because $\mathrm{NNF}(\neg(C \sqcup D))$ also is in $\mathrm{Cl}(\mathcal{T})$ – it is closed under NNF negation. The axiom added for $A_{\neg(C\sqcup D)}$ leads to the following chain of implications at any point with a father:

$$\begin{aligned} & A_{\neg(C\sqcup D)} = A_{\neg C\sqcap\neg D} && \sqsubseteq A_{\neg C} \sqcap A_{\neg D} \\ \Rightarrow \quad & \neg A_{\neg C\sqcap\neg D} && \sqsupseteq \neg(A_{\neg C} \sqcap A_{\neg D}) \\ \overset{(4.10)}{\Rightarrow} \quad & A_{C\sqcup D} && \sqsupseteq (\neg A_{\neg C} \sqcup \neg A_{\neg D}) \equiv (A_C \sqcup A_D) \end{aligned}$$

So the "$\sqsubseteq$" is sufficient in (4.11).

Having the rules for Boolean operators within a context encoded, the next step is to connect the semantics between those contexts by adding the following axioms to $\mathcal{T}'$:

$$A_{\ell:C} \sqsubseteq A_{*:C} \qquad \text{for } \ell \in \{**, f*\}, C \in \mathrm{Cl}^*, \tag{4.12}$$

which says: for a closed concept, the remembered state – whether it is the father ($\ell = f*$) or the node itself ($\ell = **$) – does not matter, it has to hold here ($\ell = *$). If a closed concept holds for the context $*f$, this means it has to hold at the father:

$$A_{*f:C} \sqsubseteq \exists f.A_{*:C} \qquad \text{for } C \in \mathrm{Cl}^*. \tag{4.13}$$

The only missing operator which has not been translated yet is the qualified number restriction "$\geq$". This requires more effort because at the counting, the possible self-loops and uplinks have to be taken into account. The main idea is to transform a concept $\geq_n R.C$ to more or less $\geq_{n-d} R.C$ where d has to be chosen depending whether an R-self-loop or an R-uplink is present and C holds at these R-successors. Therefore, the following helper construct is defined:

$$\xi_R^{\geq n}(C_1, C_2, C_3) := \bigsqcap \begin{pmatrix} (\uparrow_R \sqcap C_1) & \sqcap & (\circlearrowleft_R \sqcap C_2) & \rightarrow & \geq_{n-2} R.C_3 \\ (\uparrow_R \sqcap C_1) & \leftrightarrow & \neg(\circlearrowleft_R \sqcap C_2) & \rightarrow & \geq_{n-1} R.C_3 \\ \neg(\uparrow_R \sqcap C_1) & \sqcap & \neg(\circlearrowleft_R \sqcap C_2) & \rightarrow & \geq_{n-0} R.C_3 \end{pmatrix}$$

$\xi_R^{\leq n}$ is defined analogously by replacing $\geq$ by $\leq$ in the given definition. A formula $\geq_{n-m} R.C_3$ with $m > n$ is treated as $\top$, $\leq_{n-m} R.C_3$ as $\bot$. So $\xi_R^{\geq n}$ holds at those nodes which have at least n R-successors, distributed over these nodes:

- The father if C_1 holds here.
- The node itself if C_2 holds here.
- Each child if C_3 holds at the child.

Depending how many "virtual" R-successors – self-loop or uplink – fulfill the required concept, we need 1 or 2 successors less.

As all concepts are normalized, I only can occur in front of a qualified number restriction. If a concept $\mathsf{I}.\geq_n R.C$ holds at a node – i.e. with context $*$ – there must be at least n R-successors in those:

- The father, i.e. if $C(\bot)$ holds *here* with context $*f$. All top level occurrences of me have to replaced by $\bot$, because the context is the father with the child node remembered by the I.
- The node itself, i.e. if $C(\top)$ holds here with context $**$. In that case me has to be replaced by $\top$ because both the node of interest and the remembered node is the node itself.
- Each child, i.e. if $C(\bot)$ holds at the child with context $f*$. So me has to be replaced by $\bot$ because the node of interest is a child and the remembered node is the father (not the child itself).

The consideration for the formula $\geq_n R.C$ in the context $*$ is the same, i.e. it is the case where the node does not need to be remembered at all. So the only difference to $\mathsf{I}.\geq_n R.C$ is that C is closed and so replacing the top level me does not have any effect. So the following axiom is added to the $\mathcal{ALCHIQ}$ TBox $\mathcal{T}'$ for both $A_{*:\mathsf{I}.\geq_n R.C}$

and $A_{*:\geq_n R.C}$ as well as an analogous axiom for "$\leq$":

$$A_{*:(\mathsf{I}.)\geq_n R.C} \sqsubseteq \xi_R^{\geq n}(A_{*f:C(\bot)}, A_{**:C(\top)}, A_{f*:C(\bot)}), \tag{4.14}$$

$$A_{*:(\mathsf{I}.)\leq_n R.C} \sqsubseteq \xi_R^{\leq n}(A_{*f:C(\bot)}, A_{**:C(\top)}, A_{f*:C(\bot)}), \tag{4.15}$$

Analogously to (4.11) only the direction "$\sqsubseteq$" is needed. Note that by replacing all top level mes, the resulting concept $C(\star), \star \in \{\top, \bot\}$, is again in Cl, i.e. each free me again is guarded by some qualified number restriction. So we do not need to handle the subformula me in any context. But we have to handle qualified number restrictions that may occur in such a concept C. Those qualified number restrictions at level two have to be treated depending on the context – the remembered node and the node of interest.

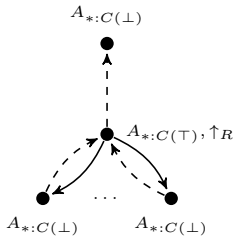
If the remembered state equals the state where the concept should hold – context $**$ – then Figure 4.5a shows for $A_{**:\geq_n R.C}$, where to check for which concept during counting of R-successors: as usual $R^{\mathcal{I}}$ is symbolized by the black arrows, $f^{\mathcal{I}}$ by the dashed ones. We now have to count how many R-successors fulfilling C there are, by taking the remembered node into account as follows.

If there is an R-uplink, then the father only has to be counted if $A_{*:C(\bot)}$ holds there. We replace me by $\bot$ because the central node is remembered, i.e. the R-uplink only is counted if $\exists f.A_{*:C(\bot)}$ holds at the evaluation point of $\geq_n R.C$.

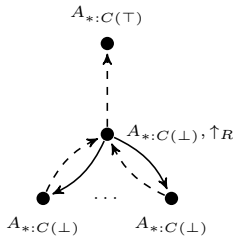
$$A_{**:\geq_n R.C} \sqsubseteq \xi_R^{\geq n}(\exists f.A_{*:C(\bot)}, A_{*:C(\top)}, A_{*:C(\bot)}) \qquad \text{for } \geq_n R.C \in \text{Cl}, \tag{4.16}$$

$$A_{**:\leq_n R.C} \sqsubseteq \xi_R^{\leq n}(\exists f.A_{*:C(\bot)}, A_{*:C(\top)}, A_{*:C(\bot)}) \qquad \text{for } \leq_n R.C \in \text{Cl}. \tag{4.17}$$

The left-hand side describes a term that already was under and $\mathsf{I}.\geq$ or $\mathsf{I}.\leq$, so C is below two qualified number restrictions. As me occurs only with the nesting level of at most 2, we can infer that $C(\top)$ and $C(\bot)$ are closed concepts, which allows the usage of context $*$ on the right-hand side.



(a) Where to check for which concept occurrences when evaluating $A_{**:\geq_n R.C}$ at the central node.



(b) Where to check for which concept occurrences when evaluating $A_{f*:\geq_n R.C}$ at the central node.

Figure 4.5

Given the same situation in context $f*$, only one difference emerges: the remembered state is the father and not the node where $A_{f*:\geq_n R.C}$ should hold. So we have to replace me by $\top$ in C when checking if we have to count the R-uplink and have to replace it by $\bot$ elsewhere. Figure 4.5b shows visually where to check for which concept. So the

counting is done by adding these TBox axioms:

$$A_{f*:\geq_n R.C} \sqsubseteq \xi_R^{\geq n}(\exists f.A_{*:C(\top)}, A_{*:C(\bot)}, A_{*:C(\bot)}) \qquad \text{for } \geq_n R.C \in \mathrm{Cl}, \tag{4.18}$$

$$A_{f*:\leq_n R.C} \sqsubseteq \xi_R^{\leq n}(\exists f.A_{*:C(\top)}, A_{*:C(\bot)}, A_{*:C(\bot)}) \qquad \text{for } \leq_n R.C \in \mathrm{Cl}. \tag{4.19}$$

The counting for the last remaining case $*f$ is different because we have to count the R-successors of the father at the child node:

$$A_{*f:\geq_n R.C} \sqsubseteq \vartheta_R^{\geq n}(C) \qquad \text{for } \geq_n R.C \in \mathrm{Cl}, \tag{4.20}$$

$$A_{*f:\leq_n R.C} \sqsubseteq \vartheta_R^{\leq n}(C) \qquad \text{for } \leq_n R.C \in \mathrm{Cl}, \tag{4.21}$$

where $\vartheta_R^{\geq n}(C)$ (and $\vartheta_R^{\leq n}(C)$ accordingly) is replaced by:

$$\bigsqcap \left(\begin{array}{rcrcl} (\exists R^-.\top & \sqcap & (A_{*:C(\top)} & \sqcap \neg A_{*:C(\bot)})) & \rightarrow \exists.f\xi_{\uparrow R}^{\geq n-1}(C) \\ (\neg\exists R^-.\top & \sqcup & (A_{*:C(\top)} & \leftrightarrow A_{*:C(\bot)})) & \rightarrow \exists.f\xi_{\uparrow R}^{\geq n+0}(C) \\ (\exists R^-.\top & \sqcap & (\neg A_{*:C(\top)} & \sqcap A_{*:C(\bot)})) & \rightarrow \exists.f\xi_{\uparrow R}^{\geq n+1}(C) \end{array} \right)$$

where $\xi_{\uparrow R}^{\geq n}(C)$ stands for $\xi_R^{\geq n}(\exists f.A_{*:C(\bot)}, A_{*:C(\bot)}, A_{*:C(\bot)})$. The following cases have to be distinguished:

- If the current node is an R-successor of its father, then it might happen that the current node is not counted as a valid R-successor fulfilling concept C, although C actually holds here with me standing for this node.

  This case can be detected if $C(\top)$ holds here but $C(\bot)$ does not: Then this node actually is a valid R-successor but the father will not recognize it properly. The fixup for this is to tell the father to require one R-successor less.

  This happens e.g. for $C = \mathsf{me}$.

- If the current node is not reachable via R from its father, then the counting of R-successors is not affected by the current node at all. It's also unaffected if the value of C is independent from the state of me.

  This happens for example if C is a closed concept.

- The last case is the inverse case of the first one: if the current point is reachable via R by the father, but C only would not hold here for me representing the current node, then the father would count it as an R-successor due to me getting replaced by $\bot$. But actually the current node should not be counted because $C(\top)$ does not hold here. $n+1$ counterbalances this miscounting.

  This happens if me is negative in C, e.g. for $C = \neg\mathsf{me}$.

With these axioms given as the $\mathcal{ALCHIQ}$-TBox $\mathcal{T}'$ and RBox $\mathcal{R}'$, the satisfiability of $\mathcal{T}$ is equivalent to the satisfiability of the following concept which – if it holds at all – will hold at the root of the tree:

$$B \equiv \neg\exists f.\top \sqcap \bigsqcap_{C \in \mathrm{Cl}\setminus\mathrm{Cl}^*} (\neg A_{f*:C} \sqcap \neg A_{*f:C}).$$

If this concept is satisfiable, then there is a model fulfilling $\mathcal{T}'$ and $\mathcal{R}'$ and being shaped as a tree with a root – a node without a father – where the contexts $f*$ and $*f$ do not hold, as they do not make sense at the root.

To ensure the non-emptiness of this root concept B, the trick known from Subsection 4.1.2 is applied: introduce a new role root $\in N'_R$ and add the TBox axiom:

$$T \sqsubseteq \exists \text{root}.B$$

Given an $\mathcal{ALCQ}\mathsf{me}_2$ TBox $\mathcal{T}$, we have just created a new $\mathcal{ALCHIQ}$ TBox $\mathcal{T}'$ and RBox $\mathcal{R}'$ as described using the concept names N'_C and role names $N'_R = N_R \dot\cup \{f, \text{root}\}$. In short, the axioms for the simple Boolean operators, for the roles R, for $\circlearrowleft_R$, $\uparrow_R$ and those listed in Table 4.2 are added to $\mathcal{T}'$.

Axiom	for each	
$A_{\ell:C} \sqsubseteq A_{*:C}$	$\ell \in \{**, f*\}, C \in \text{Cl}^*$	(4.12)
$A_{*f:C} \sqsubseteq \exists f.A_{*:C}$	$C \in \text{Cl}^*$.	(4.13)
$A_{*:(\mathsf{I}.)\geq_n R.C} \sqsubseteq \xi_R^{\geq n}(A_{*f:C(\bot)}, A_{**:C(\top)}, A_{f*:C(\bot)})$	$(\mathsf{I}.) \geq_n R.C \in \text{Cl}^*$	(4.14)
$A_{*:(\mathsf{I}.)\leq_n R.C} \sqsubseteq \xi_R^{\leq n}(A_{*f:C(\bot)}, A_{**:C(\top)}, A_{f*:C(\bot)})$	$(\mathsf{I}.) \leq_n R.C \in \text{Cl}^*$	(4.15)
$A_{**:\geq_n R.C} \sqsubseteq \xi_R^{\geq n}(\exists f.A_{*:C(\bot)}, A_{*:C(\top)}, A_{*:C(\bot)})$	$\geq_n R.C \in \text{Cl}$	(4.16)
$A_{**:\leq_n R.C} \sqsubseteq \xi_R^{\leq n}(\exists f.A_{*:C(\bot)}, A_{*:C(\top)}, A_{*:C(\bot)})$	$\leq_n R.C \in \text{Cl}$	(4.17)
$A_{f*:\geq_n R.C} \sqsubseteq \xi_R^{\geq n}(\exists f.A_{*:C(\top)}, A_{*:C(\bot)}, A_{*:C(\bot)})$	$\geq_n R.C \in \text{Cl}$	(4.18)
$A_{f*:\leq_n R.C} \sqsubseteq \xi_R^{\leq n}(\exists f.A_{*:C(\top)}, A_{*:C(\bot)}, A_{*:C(\bot)})$	$\leq_n R.C \in \text{Cl}$	(4.19)
$A_{*f:\geq_n R.C} \sqsubseteq \vartheta_R^{\geq n}(C)$	$\geq_n R.C \in \text{Cl}$	(4.20)
$A_{*f:\leq_n R.C} \sqsubseteq \vartheta_R^{\leq n}(C)$	$\leq_n R.C \in \text{Cl}$	(4.21)
$B \equiv \neg\exists f.\top \sqcap \bigsqcap_{C \in \text{Cl}\setminus\text{Cl}^*} (\neg A_{f*:C} \sqcap \neg A_{*f:C})$		
$T \sqsubseteq \exists \text{root}.B$		

Table 4.2: The main axioms at a glance

Because of the quasi-tree model property of $\mathcal{ALCQ}\mathsf{me}_2$ and of having implemented the $\mathcal{ALCQ}\mathsf{me}_2$ semantics specialized in a quasi-tree in $\mathcal{ALCHIQ}$, one can show

$$\mathcal{T} \text{ is consistent} \iff \mathcal{T}' \text{ is consistent.}$$

The basic arguments for the proof by induction over the formula structure have been sketched when introducing the new concepts and axioms. In other words, a consistency preserving reduction from $\mathcal{ALCQ}\mathsf{me}_2$ to $\mathcal{ALCHIQ}$ has been provided.

4.3 Optimizations of the Translation

In order to minimize the number of new concept names, some optimizations are done during the creation of new atomic concepts in the actual implementation of the reduction.

It concerns the creation of concepts of the kind $A_{\ell:C}$ for every pseudo-subformula $C \in \mathrm{Cl}$ and every context $\ell \in L = \{*, **, f*, *f\}$. So the mapping $A : \mathrm{Cl} \times L \to N'_C \dot\cup \{\top, \bot\}$ maps each $C \in \mathrm{Cl}$ to an atomic concept for some context, while trying to keep the number of new context names as low as possible. It is implemented as follows:

$$A(\ell, C) = \begin{cases} \lightning\,\texttt{error} & \text{if } \mathrm{open}(C) \wedge (\ell = *) \\ C & \text{if } C \in N_C \wedge (\ell \in \{*, **, f*\}) \\ A(*, C) & \text{if } \mathrm{closed}(C) \wedge (\ell \in \{**, f*\}) \\ \top & \text{if } C = \top \wedge (\ell \in \{*, **\}) \\ \bot & \text{if } C = \bot \\ A_{\ell:C} & \text{else} \end{cases}$$

So instead of writing $A_{\ell:C}$ in all the axioms in the transformation the term $A(\ell, C)$ has to be put, which inserts an normalized concept name. Let us look at the cases and see why the mapping is correct:

- If C is open, i.e. it contains an unguarded me, then the $*$-context does not make any sense. So at any point the transformation must not create such a concept in $*$. So the implementation just throws an error.
- If C is an atomic concept in the original ontology, then the remembered node does not matter, i.e. the contexts $*$, $**$, $f*$ do not have any different meaning, so we only use the original concept name C for all of them.
- If C is a closed concept, then the remembered node does not matter. So the number of new concept names is kept smaller by using one concept name $A_{*:C}$ for all concepts in $\{*, **, f*\}$.
- Instead of telling in (4.7), that $A_{*:\top}$ and $A_{**:\top}$ should hold everywhere, both concepts directly can be replaced by $\top$.
- Analogously, instead of putting (4.6), all concepts $A_{\ell:\bot}$, $\ell \in L$, are replaced by $\bot$.

The replacement by $A(\ell, C)$ also enables the minimization of the generated axioms and the formulas in them. Beside reducing the file size this also improves the readability of the formulas. The following identities are applied recursively on the formulas:

$$\begin{array}{cccc} C \sqcap \top \mapsto C & C \sqcap \bot \mapsto \bot & C \sqcup \bot \mapsto C & C \sqcup \top \mapsto \top \\ \geq_0 R.C \mapsto \top & \geq_{n'} R.\bot \mapsto \bot & \leq_n R.\bot \mapsto \top & \end{array}$$

with $C \in \mathfrak{C}$, $R \in \mathfrak{R}$, $n' \geq 1$, $n, n' \in \mathbb{N}_0$. The mappings for $\sqcap$ and $\sqcup$ apparently map the formulas to equivalent ones. The replacement for quantified number restrictions are also equivalent to the original formula because there are always at least 0 R-successors fulfilling C; and there are never more than 0 R-successors fulfilling $\bot$ or in other words there are at most n R-successors fulfilling $\bot$ for any $n \in \mathbb{N}_0$.

5 Self-Reference in OWL Ontologies

This chapter describes the implementation of an $\mathcal{ALCHIQ}\mathsf{me}_2$ consistency checker using the encoding of role inverses and hierarchies in Chapter 3 and the reduction to $\mathcal{ALCHIQ}$ discussed in Chapter 4. When dealing with ontologies in practice, the most widespread format is the *Web Ontology Language (OWL)*. So there are already reasoners which can check the consistency of an OWL ontology.

Instead of defining an ontology format with self-reference from scratch, it is more productive to extend OWL with self-reference. OWL then can be used as an container for $\mathcal{ALCHIQ}\mathsf{me}_2$ and $\mathcal{ALCQ}\mathsf{me}_2$ ontologies and as the input format for the reasoning.

As we did not need a full consistency checking algorithm but only a consistency preserving reduction to $\mathcal{ALCHIQ}$, further steps have to be done: we need a consistency checker for $\mathcal{ALCHIQ}$ ontologies. OWL is strictly more expressive than $\mathcal{ALCHIQ}$, so the solution is to put the resulting TBox into an OWL ontology and check its consistency by an OWL reasoner. Therefore the Hermit OWL Reasoner [11] is used, which is licensed under the GNU General Public License v3.

The implementation of the main reduction from OWL ontologies with self-reference to plain OWL ontologies, which accords to the transformation from $\mathcal{ALCQ}\mathsf{me}_2$ to $\mathcal{ALCHIQ}$, is now discussed. The implementation is called `i-me-owl`.

5.1 OWL Ontologies

As defined by W3C [12] there are many representation formats for OWL ontologies. The implementation of this work uses the reference implementation OWL API [13] and thus works with all official representation formats. The relevant parts of the human readable format – called Functional-Style Syntax [14] – is defined now, because it will be used for describing OWL ontologies in the following. For a more detailed description see Hitzler et al. [12]. The syntax consists of expressions of the form

```
NAME( ARGUMENTS ...)
```

So each entity has a name followed by a list of whitespace separated arguments; the list is put in parentheses. With that in mind the syntax is described intuitively by the examples.

An OWL ontology is an accumulation of axioms, each of them talking about roles and concepts. In the context of OWL ontologies, concepts are called classes and roles are

called object properties. The name of such a class or object property is a full URI followed by `#` and the actual name. To simplify typing class names, the URI can be abbreviated by prefixes. So e.g. by defining a prefix `p`

```
Prefix(p:=<http://www8.cs.fau.de/yet-another-example.owl#>)
```

the class name `p:SomeClass` stands for:

```
http://www8.cs.fau.de/yet-another-example.owl#SomeClass
```

The default prefix `owl` contains the classes `Thing` and `Nothing` which correspond to the concepts $\top$ and $\bot$. In the following we assume the empty prefix to be set to some reasonable prefix:

```
Prefix(:=<http://www8.cs.fau.de/yet-another-example.owl#>)
```

So an atomic concept C can be adapted to the OWL class `:C` and a role R to the OWL property `:R`. The TBox axioms "$\equiv$" and "$\sqsubseteq$" are called `EquivalentClasses` and `SubClassOf`, i.e. an axiom like

$$A \equiv \top$$

is adapted to:

```
EquivalentClasses( :A owl:Thing )
```

For each $\mathcal{ALCHIQ}$ concept constructor there is an analogous construct in OWL. E.g. $\exists R.(D \sqcup \neg C)$ is expressed in OWL as:

```
ObjectSomeValuesFrom( :R
                      ObjectUnionOf( :D
                                     ObjectComplementOf( :C )))
```

The full mapping from $\mathcal{ALCHIQ}$ concept constructors to OWL class constructors is given by Table 5.1. The reverse mapping of an OWL ontology to $\mathcal{ALCHIQ}$ constructs as unicode text also has been implemented for debugging purposes. E.g. if the actual content of the ontology is as follows.

```
Prefix(:=<http://example.com/yet-another-example.owl#>)
Ontology( <http://example.com/yet-another-example.owl>
  SubClassOf( :A ObjectComplementOf( :B ))
  SubClassOf( :B ObjectComplementOf( :C ))
  SubClassOf( :C ObjectIntersectionOf(ObjectComplementOf(:A)
      ObjectComplementOf(:B)))
)
```

Then the pretty printer gives the following:

```
A ⊆ ¬B
B ⊆ ¬C
C ⊆ ¬A ⊓ ¬B
```

Notion	$\mathcal{ALCHIQ}$ construct	OWL construct	Parameters
Top	$\top$	`owl:Thing`	
Bottom	$\bot$	`owl:Nothing`	
Negation	$\neg C$	`ObjectComplementOf`	`(:C)`
Intersection	$C \sqcap D$	`ObjectIntersectionOf`	`(:C :D)`
Union	$C \sqcup D$	`ObjectUnionOf`	`(:C :D)`
Exists	$\exists R.C$	`ObjectSomeValuesFrom`	`(:R :C)`
Forall	$\forall R.C$	`ObjectAllValuesFrom`	`(:R :C)`
At least	$\geq_n R.C$	`ObjectMinCardinality`	`(`n `:R :C)`
At most	$\leq_n R.C$	`ObjectMaxCardinality`	`(`n `:R :C)`

Table 5.1: Mapping $\mathcal{ALCHIQ}$ constructs to OWL constructs

5.2 OWL Ontologies with Self-Reference

OWL itself – like the very similar and commonly known description logic $\mathcal{SROIQ}$ [2] – does not provide a way to express self-reference up to distance 2 of I and me in concepts[1]. To be able to use the OWL API for parsing of the input files with self-reference anyway, it requires a OWL compatible encoding of the I and me construct. Therefore, the role name `:I` and class name `:me` both with prefix

```
http://www8.cs.fau.de/yet-another-example.owl#
```

are used. To encode $\mathsf{I}.C$ both `ObjectSomeValuesFrom( :I :C )` and `ObjectAllValuesFrom( :I :C)` are recognized by the implementation. E.g. a concept like $\mathsf{I}.\geq_4 \neg\mathsf{me}$ can be encoded as:

```
ObjectSomeValuesFrom( :I ObjectMinCardinality( 4 ObjectComplementOf( :me )))
```

Note that this encoding of self-reference in OWL is only recognized by the implementation of this work! It is not any standardized format. This kind of extension of OWL is done, because it allows us to run the implementation with real OWL ontologies in circulation later – even though they do not use self-reference at all.

The pretty printer of the implementation also recognizes I and me in this format and gives an appropriate output. E.g. for the OWL ontology

[1] Although $\mathcal{SROIQ}$ allows a concept $\exists R.\text{Self}$ standing for the $\mathcal{ALCQ}\mathsf{me}_2$ concept $\mathsf{I}.\exists R.\mathsf{me}$.

```
Prefix(:=<http://www8.cs.fau.de/yet-another-example.owl#>)
Ontology( <http://www8.cs.fau.de/yet-another-example.owl>
  EquivalentClasses( :Celebrity
      ObjectSomeValuesFrom( :I
      ObjectAllValuesFrom( :seenBy ObjectSomeValuesFrom( :
         knows :me))))
)
```

the output of the pretty printer is:

```
Celebrity ≡ I.∀seenBy.(∃knows.me)
```

5.3 Usage

Assuming the implementation `i-me-owl` is set up like described in Appendix A, the calling schema is described by this usage line:

`./i-me-owl` *action* [*input* [*output*]]

It performs a certain *action*, reads the input data from the file specified by *input* or from stdin if the string "`-`" is given. The result is written to the file specified by *output* or to stdout if the string "`-`" is given. *input* and *output* are optional arguments and both default to the string "`-`". The most important *actions* can be found in Table 5.2. The

action	Behaviour
`--help`	List all possible actions and their meaning.
`print`	Pretty print the given ontology using unicode characters.
`normalize`	Do the normalization described in Chapter 3 and resolve role inverses and role hierarchies as described in Chapter 3.
`convert`	First normalize, then do the actual transformation using the optimizations from Section 4.3 and output a plain OWL ontology.

Table 5.2: The usage of `i-me-owl`

results of the axioms generated by the implementation can be visualized very well.

Example 5.1 (`chain.owl`)**.** Consider the $\mathcal{ALCQ}\mathsf{me}_2$ TBox which is encoded as follows in OWL:

```
T ≡ I.∃S.¬me
T ≡ I.∀R.(∃S.me)
```

It states that every node has an S-successor which is not itself and that the inverse of R is a subrole of S.

After converting the ontology using `./i-me-owl convert`, the classification – the hierarchy of concepts regarding concept inclusion – gives Figure 5.1. There is a vertex for each class of equivalent concepts and an edge from C to D iff it is $C \sqsupseteq D$ and there is no atomic concept between them.

The concept *Root* denotes the concept that should hold at the root of the tree in any model. The other concepts are the self-loop and uplink concepts as well as the $A_{\ell:C}$ concepts known from the translation. It is worth mentioning, that many concept inclusions we intuitively see in the TBox are reflected in the hierarchy. Only some of those – together with their intuitive explanations – are:

- $\circlearrowleft_R$ is a subconcept of $\circlearrowleft_S$, because R is the inverse subrole of S.
- $A_{**:\geq_1 S.\mathsf{me}}$ is equivalent to $\circlearrowleft_S$, it directly follows from the $\mathcal{ALCQ}\mathsf{me}_2$ semantics. Note that $A_{**:\geq_1 R.\mathsf{me}}$ does not occur in the classification because $\geq_1 R.\mathsf{me}$ is not a pseudo-subformula in the given TBox.
- $\uparrow_R$ is a subconcept of $A_{*f:\leq_1 S.\mathsf{me}}$, because R^- is a subrole of S.

This confirms that the $\mathcal{ALCQ}\mathsf{me}_2$ formulas are encoded correctly by the implementation. ▲

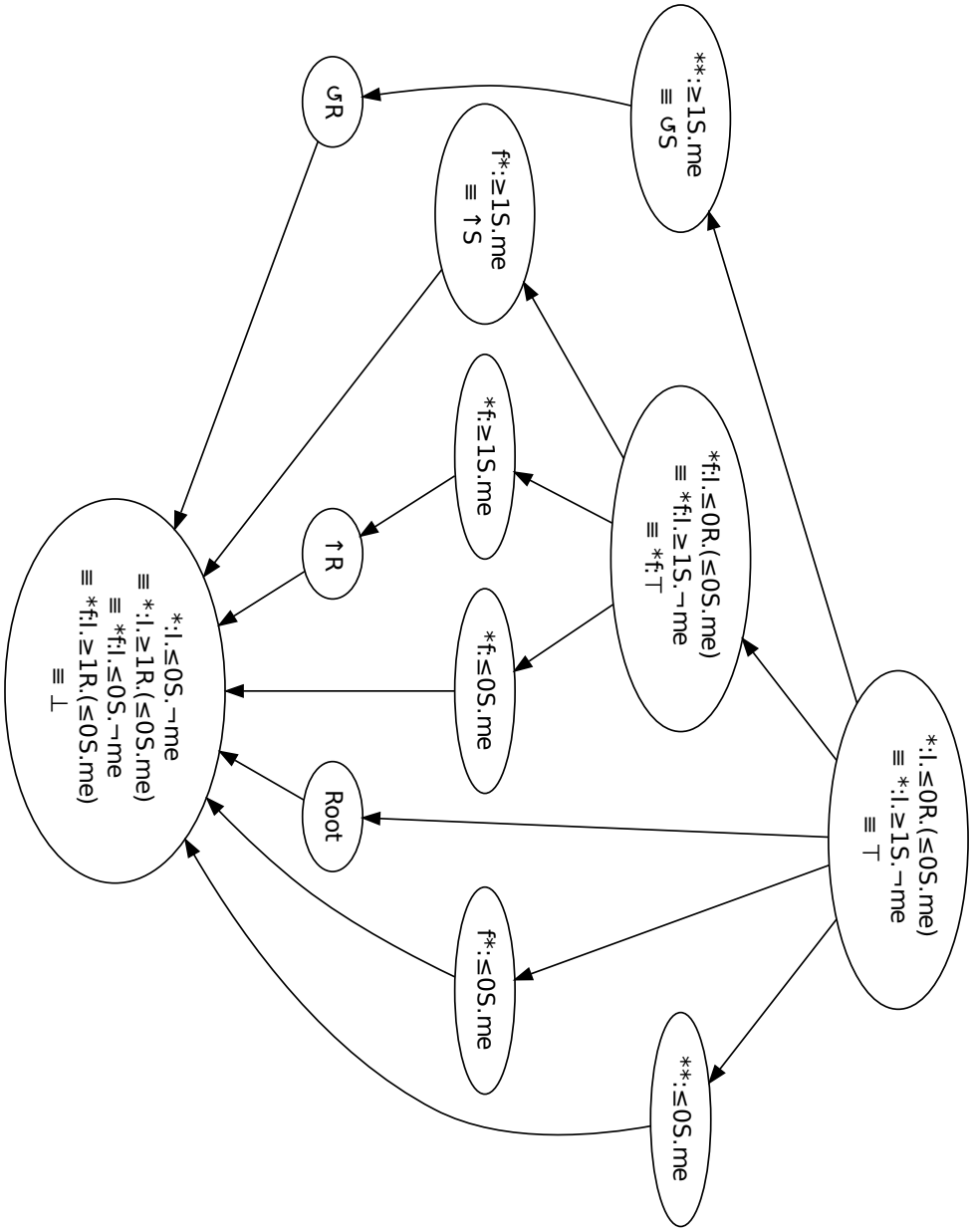


Figure 5.1: Classification for the chain.owl ontology

5.4 Benchmarking

For any type of reduction, the blowup is a characteristic property. It describes how large the output of the reduction is compared to the input. In case of ontologies, two types of measurements are analyzed:

- The file size: how much more memory does it take to save the ontology?
- The reasoning time: how longer does it take to check the consistency of the ontology?

The blowup of the file size already can be estimated theoretically in the $\mathcal{O}$-notation: let the original file have the size of n bytes. The original ontology is now expressed using constantly many atomic concepts for each subformula. Each atomic concept representing a subformula needs constantly many axioms describing its behaviour, so this part takes $\mathcal{O}(n)$. But for each subformula, a new atomic concept with a new name has to be created, i.e. it is needed to encode $\mathcal{O}(n)$ new concept names – each of the length $\mathcal{O}(\log n)$ – which in total takes $\mathcal{O}(n \cdot \log n)$ disk space.

Additionally a role and a concept has to be added to describe the tree. This description consists of the "father of"-relation and the concept which holds at the root. But the number of concept names in the formula for the root is linearly bounded in the number of subformulas in the original ontology, i.e. the entire formula again needs in $\mathcal{O}(n \cdot \log n)$ space.

So in total it is expected that the generated ontology has a file size bounded by $\mathcal{O}(n \cdot \log n)$. But it is likely not to see the logarithmic factor for ontologies of some reasonable size because the extra space needed for encoding the concept names with e.g. up to 20 digits is much lower than the constant factor for each axiom.

For the benchmarking, an OWL ontology containing the axioms $F_1, \ldots, F_m$ is tested as follows. Consider the ontologies O_i consisting of the axioms $F_1, \ldots, F_i$. For each – or every tenth – O_i, measure its file size s_i in bytes, transform it using the implementation to some ontology O_i' with some file size s_i'. Then plot the graphs for $i \mapsto s_i$, $i \mapsto s_i'$, $i \mapsto \frac{s_i'}{s_i}$ in logarithmic scale and $i \mapsto \frac{s_i'}{s_i}$ also in linear scale. The blowup in size is $\frac{s_i'}{s_i}$.
The graph for the facts $F_i = (C_i \equiv D_i)$ can be seen in Figure 5.2. As the graph visualizes, the blowup nearly is constant, which is not surprising, because there are many independent axioms. For a more realistic example like the pizza ontology[15] it takes some steps until the constant factor is reached, as shown by Figure 5.3.

Testing the time for consistency checking has been done similar to the file size measurement: instead of measuring the file size s_i, the time t_i for checking the consistency of O_i has been plotted. Therefore, t_i is the average CPU time[2] in milliseconds from doing the consistency checking 4 times by the Hermit reasoner.

[2]Time spent in user and kernel mode, without the time waiting for a time slice or for IO

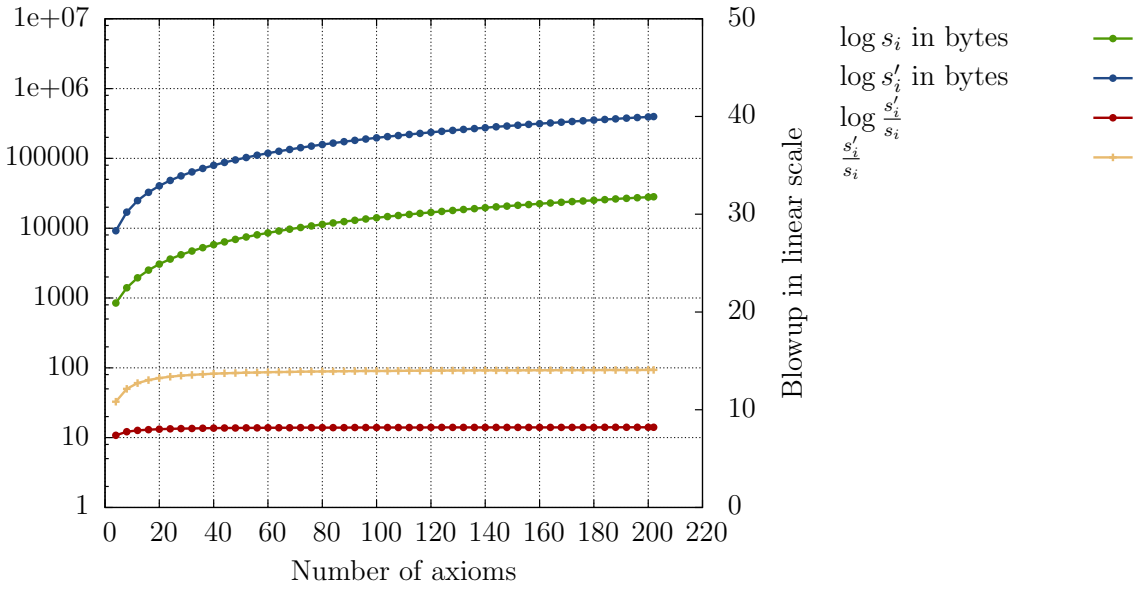


Figure 5.2: Size blowup for the axioms $F_i = (C_i \equiv D_i)$

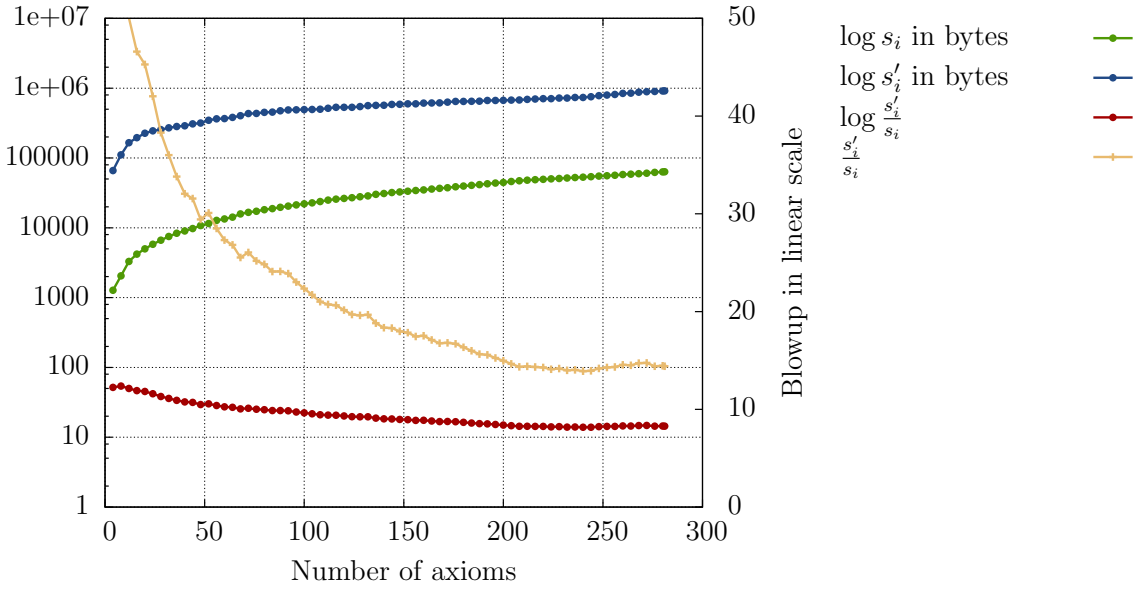


Figure 5.3: Size blowup of the pizza ontology

Measuring the time blowup for the same ontologies as before shows, that the blowup for consistency checking time is very small, as shown by Figure 5.4 and Figure 5.5.

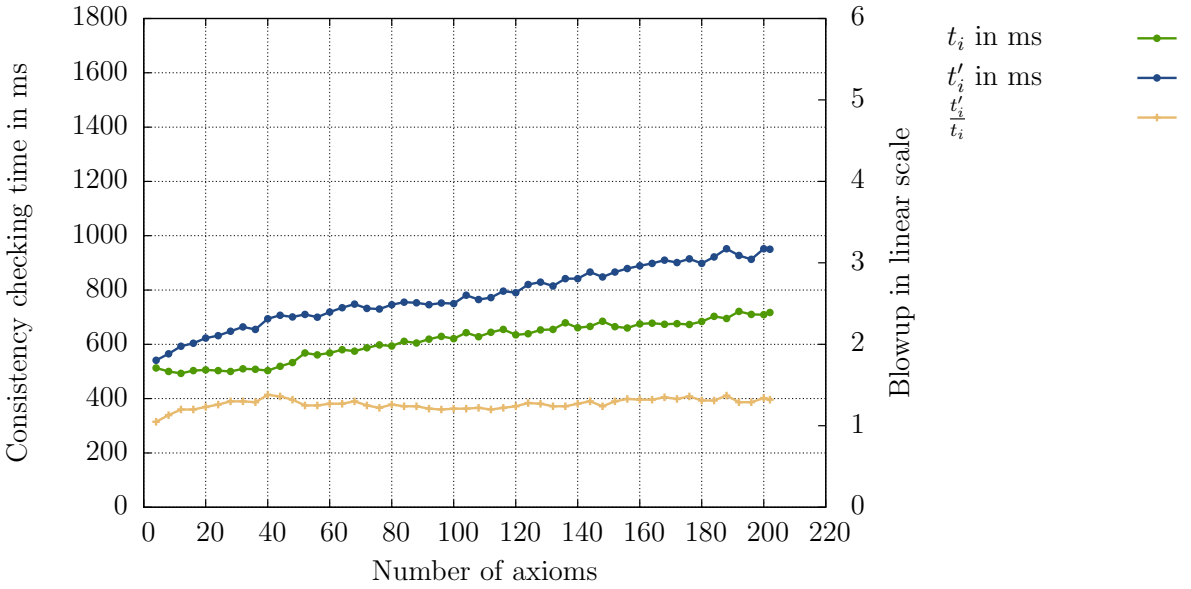


Figure 5.4: Time blowup for the axioms $F_i = (C_i \equiv D_i)$

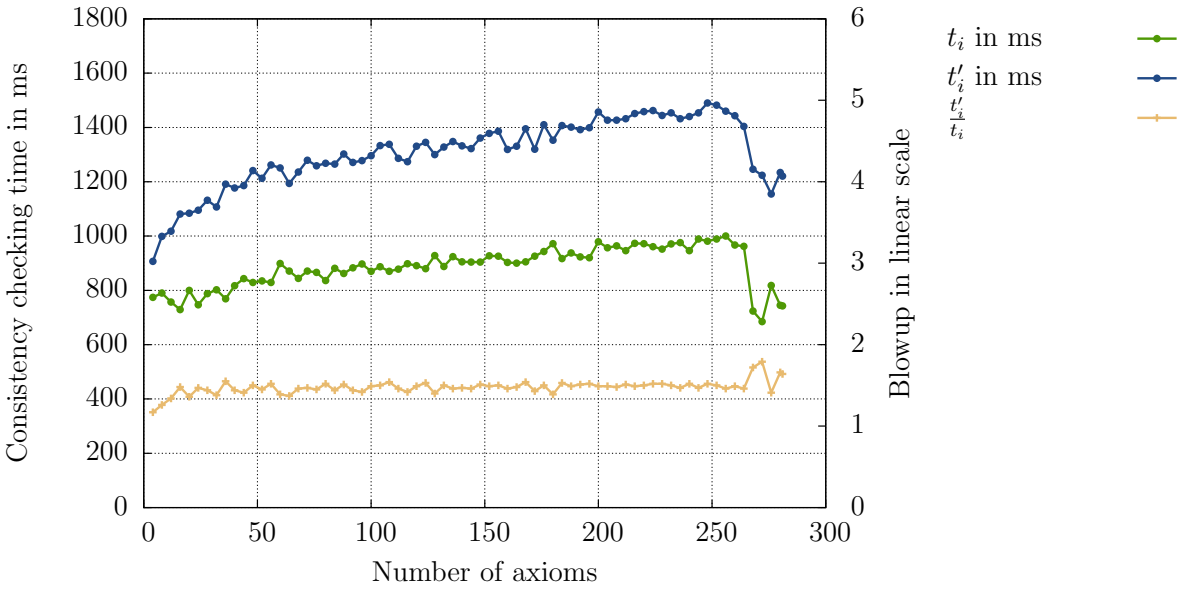


Figure 5.5: Time blowup of the pizza ontology

6 Conclusion

We have seen that $\mathcal{ALCQ}\mathsf{me}_2$ is decidable which was shown by the polynomial reduction to $\mathcal{ALCHIQ}$. As TBox and RBox consistency checking for the latter one is EXPTIME-complete [16], consistency checking for $\mathcal{ALCQ}\mathsf{me}_2$ is EXPTIME-complete, too.

The reduction enabled a conversion from OWL ontologies with the self-referential construct to plain OWL ontologies, which can be fed to any reasoner compatible to OWL. This implementation also is usable in practice, as the blowup in reasoning time is applicable. So additional to modelling a narcissist in OWL, reasoning is now possible about those who say "I only like people who also like me".

The practical view on performance has to be distinguished from the theoretical complexity class. As shown, in practice the blowup in file size is determined by possibly large factors that may disappear in the $\mathcal{O}$-notation completely.

In future work it may be investigated how self-reference interacts with other type of axioms like ABoxes, which are not considered in this thesis. It also can be examined how bounded self-reference interacts e.g. with transitive roles, which is an additional feature of $\mathcal{SROIQ}$ compared to $\mathcal{ALCHIQ}$.

A Setting up i-me-owl

These are the system requirements for using the implementation i-me-owl:

- scala $\geq$2.9.2
- bash $\geq$4.2.0
- GNU Make (Only for building)
- git (Only when fetching the sources from the internet)

Having the requirements installed, one can setup the implementation by doing the following steps in a shell.

The sources for `i-me-owl` can be fetched via git:

```
git clone git://git.code.sf.net/p/imeowl/code i-me-owl
cd i-me-owl/
```

During git clone, an internet connection is required. Current information about `i-me-owl` can be found on the project page [17]. For build instructions and further information see the `README` file in the project directory.

Then `./i-me-owl` can be used as described in Section 5.3. The project also features many examples for both $\mathcal{ALCHIQ}$ and $\mathcal{ALCHIQ}\mathsf{me}_2$ ontologies, which can be found in the `examples/` subdirectory.

Bibliography

[1] www-webont-wg@w3.org mailing list, "SWOL versus WOL". http://lists.w3.org/Archives/Public/www-webont-wg/2001Dec/0169.html, 2001.

[2] Markus Krötzsch, Frantisek Simancik, and Ian Horrocks. A description logic primer. *CoRR*, abs/1201.4089, 2012.

[3] Sebastian Rudolph. Foundations of description logics. In Axel Polleres, Claudia d'Amato, Marcelo Arenas, Siegfried Handschuh, Paula Kroner, Sascha Ossowski, and Peter F. Patel-Schneider, editors, *Reasoning Web*, volume 6848 of *Lecture Notes in Computer Science*, pages 76–136. Springer, 2011. ISBN 978-3-642-23031-8.

[4] W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 27 October 2009. Available at http://www.w3.org/TR/owl2-overview/.

[5] Maarten Marx. Narcissists, stepmothers and spies. In *In Proc. of DL'02, volume 53. CEUR*, 2002.

[6] Balder Ten Cate and Massimo Franceschet. On the complexity of hybrid logics with binders. In *Proc. of the 19th CSL, 2005, LNCS 3634 (2005)*, pages 339–354. Springer, 2005.

[7] Daniel Gorín and Lutz Schröder. Narcissists are easy, stepmothers are hard. In Lars Birkedal, editor, *FoSSaCS*, volume 7213 of *Lecture Notes in Computer Science*, pages 240–254. Springer, 2012. ISBN 978-3-642-28728-2.

[8] Daniel Gorín and Lutz Schröder. Celebrities don't follow their followers: Binding and qualified number restrictions.

[9] Daniel Gorín and Lutz Schröder. Extending ALCQ with bounded self-reference. In Silvio Ghilardi and Lawrence Moss, editors, *Proc. Advances in Modal Logic 2012, AiML 2012*. College Publications, 2012.

[10] Moshe Y. Vardi. Why is modal logic so robustly decidable? In Neil Immerman and Phokion G. Kolaitis, editors, *Descriptive Complexity and Finite Models, Proceedings of a DIMACS Workshop, January 14-17, 1996, Princeton University*, volume 31 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 149–184. American Mathematical Society, 1996. ISBN 0-8218-0517-7.

[11] Hermit owl reasoner. `http://www.hermit-reasoner.com/`, Status as of february 2013.

[12] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph, editors. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-primer/`.

[13] The owl api. `http://owlapi.sourceforge.net/`, Status as of february 2013.

[14] Peter F. Patel-Schneider Boris Motik and Bijan Parsia, editors. *OWL 2 Web Ontology Language, Structural Specification and Functional-Style Syntax (Second Edition)*. W3C Recommendation, 11 December 2012.

[15] Pizza ontology v1.5 (2007/02/12). `http://www.co-ode.org/ontologies/pizza/2007/02/12/`, Status as of february 2013.

[16] Stephan Tobies. Complexity results and practical algorithms for logics in knowledge representation. *CoRR*, cs.LO/0106031, 2001.

[17] I-me-owl project page on sourceforge. `http://sourceforge.net/p/imeowl`.

Index