Standard and Non-standard Reasoning in Description Logics

Dissertation

zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von
Dipl.-Inform. Sebastian-Philipp Brandt
geboren am 25. April 1974 in Düsseldorf

verteidigt am
5. April 2006

Gutachter:
Prof. Dr.-Ing. Franz Baader,
Technische Universität Dresden
Prof. Dr.-Ing. Diego Calvanese,
Freie Universität Bozen-Bolzano
Prof. Dr. rer.-nat. habil. Bernhard Ganter,
Technische Universität Dresden

Dresden, im Juni 2006

Contents

CHAPTER 1

INTRODUCTION

1.1 Knowledge representation with Description Logics

In general, the term knowledge representation (KR) stands for the approach to store knowledge about a given domain of discourse in explicit form in a *knowledge base*, and to automatically *infer* all implicit consequences of the information stored. The algorithm performing this inference hereby depends only on the general *syntactic format* of the knowledge base but is independent of its actual domain-specific content. Such a *KR system*, i.e., a knowledge base together with an inference algorithm, is distinguished by the underlying representation formalism.

Origin

Description Logics are one class of such representation formalisms that evolved from earlier logic based KR formalisms, especially *semantic networks* [Qui67] and *frames* [Min81]. Though very dissimilar on the surface, both predecessors aim to represent classes of individuals and relations between such classes.

In the case of semantic networks, this is accomplished by labeled directed graphs in which labeled vertices represent concepts or individuals and labeled edges represent relations between them. More precisely, 'is-a'-edges are used either to denote subconcept-superconcept relations between concepts or to specify the type of an individual by a concept. Edges with other labels, e.g., 'has colour', can be used to denote properties of concepts or individuals.

In basic frame systems, concepts are represented by *frames* for each of which a name, a list of direct super-frames, and a list of *slots* can be specified. Slots are used to denote properties of concepts by linking to other frames, e.g., a slot 'colour' might be filled by the name of a frame representing a colour.

Both semantic networks and frame systems lacked a formally well-defined semantics. As a consequence, two different KR systems based on one of these formalisms could give contradictory answers upon the same input. In order to overcome this problem, the meaning of a knowledge base had to be defined independently of any specific reasoning algorithm. In the case of frame systems, this was accomplished by means of first-order logics [Hay79]. More precisely, it was observed that restricted semantic networks and basic frame systems could be translated into relatively small fragments of first-order logics [BL85], in which reasoning problems are decidable. Furthermore, simpler specialized algorithms instead of theorem provers could be used to reason w.r.t. these fragments. These findings gave rise to so-called 'concept languages' which allow to define concepts by means of formulae

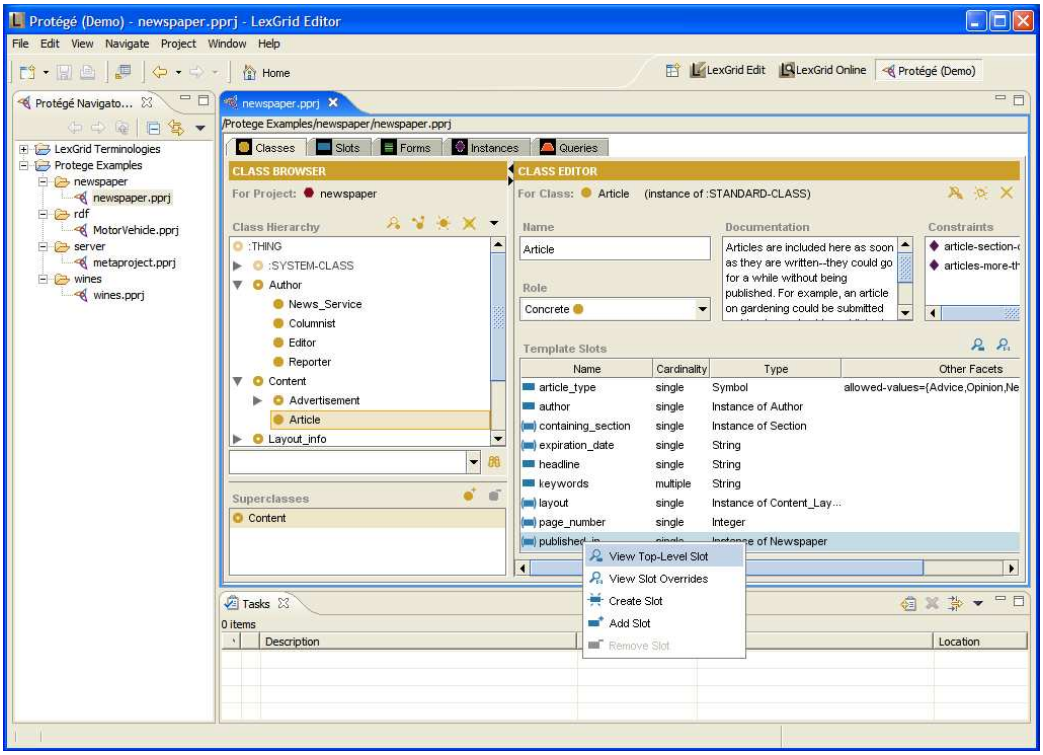


Figure 1.1.1: A typical editor for DL knowledge bases

over fragments of first-order logics. Probably in order to emphasize the fact that formulae are used to *describe* concepts with an underlying semantics based on first-order *logics*, the name gradually changed from 'concept languages', e.g., [HB91], 'concept description languages', e.g., [Sch92], or 'terminological logics', e.g., [Bra91], to *Description Logics (DL)*.

Usually, KR systems based on DLs are called *DL systems*. The earliest system that is today called a DL system was KL-ONE [BS85] by Ron Brachman, presented 1985. Although it was shown later on that reasoning w.r.t. KL-ONE knowledge bases is undecidable [SS89], KL-ONE anticipated many aspects of syntax, semantics, and reasoning services of modern DL systems. The most well-known modern DL reasoners are FaCT [Hor98], Racer [HM01b], and, more recently, Pellet [SP04]. Modern DL systems are typically modularized, comprising a DL reasoner and an independent knowledge editor, communicating to each other via a standardized protocol, the DIG[1] interface [BMC03]. An example of a modern knowledge editor is Protégé [GMF+03] shown in Figure 1.1.1, which will be discussed in more detail after the introduction of the syntax of DL.

DL Syntax

When introducing DL syntax, it should be clarified that the term DL does not denote exactly one KR formalism. Instead, there exist many DLs of different expressive power, mostly[2] corresponding to fragments of first-order logics. Each DL is characterized by the constructors available to build complex terms, called *concept descriptions*, from a fixed

[1] Description Logics Implementation Group

[2] Some DLs provide constructors, e.g., transitive closure of relations, that cannot be characterized by first-order logics.

set of atomic concepts. Apart from basic propositional constructors, most DLs provide so-called *roles* by which binary relations between concepts can be expressed.

In a DL system, the knowledge base typically comprises two sets, the *terminological box (TBox)* and the *assertional box (ABox)*. In its most basic form, the TBox contains concept *definitions* of the form $A \equiv C$ which define a concept *name* A by a concept *description* C. Concept descriptions are terms built from primitive concepts and roles by means of constructors provided by the DL. If present, the ABox on the other hand contains assertions about individuals, either assigning individuals to concepts or establishing binary relations between individuals via roles. The following example shows how a simple DL knowledge base, i.e., a TBox together with an ABox, might look like.

Example 1.1.1 Assume that we have atomic concepts Person and Female and a role has_child. In a DL providing conjunction ($\sqcap$) and existential restriction ($\exists$), we can now define a TBox $\mathcal{T}$ as follows.

$$\mathcal{T} := \{\ \mathit{Parent} \equiv \mathsf{Person} \sqcap \exists \mathsf{has\_child}.\mathsf{Person},\\ \mathit{Mother} \equiv \mathit{Parent} \sqcap \mathsf{Female}\ \}$$

Hence, a parent is a person in has-child relation to another person and a mother is a parent and is female. Given individuals Alice, Bob, and Charles, an ABox $\mathcal{A}$ using the above definitions might look as follows.

$$\mathcal{A} := \{\ \mathit{Mother}(\mathsf{Alice}),\ \mathsf{Person}(\mathsf{Charles}),\ \mathsf{Person}(\mathsf{Bob}),\ \mathsf{has\_child}(\mathsf{Bob}, \mathsf{Charles})\ \}$$

In this example, $\mathcal{A}$ states that Alice is a mother and that the person Charles is in has-child relation to the person Bob. □

In order to see how a DL knowledge base appears from the perspective of the user of a modern knowledge editor, consider Figure 1.1.1. In the column on the left labeled 'Protégé Navigato...', the user has selected a knowledge base named 'newspaper.pprj' which is opened for editing in the large sub-window on the right labeled 'newspaper.pprj'. There, a sub-window labeled 'Classes' shows a 'Class Browser' and a 'Class Editor'. The class browser shows a list containing the concepts defined in the TBox of 'newspaper.pprj', e.g., *Content*, *Advertisement*, *Article*, etc. In our case, the user has selected the concept *Article* for inspection in the class editor, where the main part of the definition of *Article* is shown in the list labeled 'Template Slots'. For example, the line 'author – single – Instance of Author' corresponds to the concept description $\exists \mathsf{author}.\mathit{Author}$ with the additional restriction that every article has at most one author. Note that *Author* also occurs in the class browser. Thus, every line in the list 'Template Slots' corresponds to a conjunct in the definition of *Article*. The ABox of 'newspaper.pprj' is hidden in the sub-window labeled 'Instances'. One reason for not presenting a knowledge base in the actual DL syntax is to support domain experts not familiar with the underlying formalism.

DL Semantics

As mentioned above, the distinguishing feature of DL over older KR formalisms is a well-defined semantics. The meaning of concepts and roles is defined w.r.t. an *interpretation* $\mathcal{I}$ which consists of a *universe* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$. By means of the interpretation function, every concept occurring in a DL-TBox is interpreted as a subset of $\Delta^{\mathcal{I}}$, every role as a binary relation on $\Delta^{\mathcal{I}}$, and every individual occurring in the ABox as one element of $\Delta^{\mathcal{I}}$. The interpretation of complex concept descriptions is inductively defined by the semantics of the constructors provided by the DL under consideration.

For instance, the conjunction $\mathit{Parent} \sqcap \mathsf{Female}$ is interpreted by the intersection of the interpretations of *Parent* and Female. An interpretation $\mathcal{I}$ *satisfies* a TBox $\mathcal{T}$ if and only if left-hand side and right-hand side of every definition in $\mathcal{T}$ are interpreted identically. In this case, $\mathcal{I}$ is called a *model* of $\mathcal{T}$. Similarly, $\mathcal{I}$ satisfies an ABox if and only if for every assertion $A(a)$ in $\mathcal{A}$, the interpretation of a is contained in the interpretation of A, and for each $r(a, b)$ in $\mathcal{A}$, the interpretation of r contains the pair of the interpretations of a and b. Moreover, often the *unique name assumption* holds, i.e., distinct individuals do not have the same interpretation. This kind of semantics is usually called *descriptive* semantics [Neb91]. Note that another kind of semantics relevant for the present work will be introduced later on, namely *greatest-fixedpoint* semantics.

Example 1.1.2 Consider the knowledge base from Example 1.1.1. Let $\mathcal{I}$ be an interpretation with universe $\Delta^{\mathcal{I}} := \{a, b, c, d\}$ and an interpretation function that interprets the primitive concepts, roles, and individuals from $\mathcal{T}$ and $\mathcal{A}$ as follows: $\mathsf{Person}^{\mathcal{I}} = \{a, b, c, d\}$, $\mathsf{Female}^{\mathcal{I}} = \{a, d\}$, $\mathit{Parent}^{\mathcal{I}} = \{a, b\}$, $\mathit{Mother}^{\mathcal{I}} = \{a\}$, $\mathsf{Alice}^{\mathcal{I}} = a$, $\mathsf{Bob}^{\mathcal{I}} = b$, $\mathsf{Charles}^{\mathcal{I}} = c$, and $\mathsf{has\_child}^{\mathcal{I}} = \{(a, d), (b, c)\}$. Then $\mathcal{I}$ is a model of $\mathcal{T}$ together with $\mathcal{A}$. Note that in $\mathcal{I}$, Bob is a parent although the ABox did not explicitly specify this. □

The main asset of KR systems is their ability to *reason* over the knowledge base, i.e., to make implicitly hidden knowledge explicit. For instance, in Example 1.1.2 one might suspect that Bob must be contained in the interpretation of *Parent* in *every* model of $\mathcal{T}$ and $\mathcal{A}$, which would make Bob being a parent an implicit consequence of the knowledge base. Questions of this nature can be answered by means of certain inference services.

Reasoning

In the context of DL systems, two classes of *inference services* are distinguished, namely *terminological* ones, taking into account only the TBox, and *assertional* ones, additionally considering the ABox. Throughout this work, we will mostly be concerned with terminological inference services. We shall also discuss ways to reduce assertional reasoning to terminological reasoning under certain circumstances.

The most basic terminological inference services supported by most DL systems are satisfiability and subsumption. A concept is *satisfiable* w.r.t. a given TBox if and only if the TBox has a model in which the interpretation of the concept is not empty. A concept A is *subsumed* by a concept B w.r.t. a TBox $\mathcal{T}$ if and only if A is more specific than B in the sense that, w.r.t. every model of $\mathcal{T}$, the interpretation of A is a subset of that of B. For instance, in Example 1.1.1, *Mother* is subsumed by *Parent* w.r.t. $\mathcal{T}$, which is in turn subsumed by Person. To *classify* a TBox $\mathcal{T}$ means to compute all subsumption relationships between concepts occurring in $\mathcal{T}$. Usually, classification is the main system service of a DL reasoner.

The two most common assertional inference services are the consistency and the instance problem. An ABox is *consistent* w.r.t. a TBox if ABox and TBox have a common model. An individual a is an *instance of* a concept description C w.r.t. a TBox and an ABox if and only if the interpretation of a is an element of the interpretation of C w.r.t. every model of the TBox together with the ABox.

In the remainder of the present subsection, we introduce a more powerful TBox formalism supported by many modern DL systems, such as FaCT, Racer, or Pellet. Moreover, we discuss quality criteria for DL terminologies.

General TBoxes

General TBoxes are sets of so-called *general concept inclusion (GCI)* axioms of the form $C \sqsubseteq D$, where both C and D are arbitrary concept descriptions, i.e., not necessarily atomic concepts. An interpretation satisfies a GCI $C \sqsubseteq D$ if and only if the interpretation of C is a subset of that of D. Hence, D is implied whenever C holds. General TBoxes extend our previous definition of TBoxes in the sense that every definition $A \equiv C$ is equivalent to the set of GCIs $\{A \sqsubseteq C, C \sqsubseteq A\}$. The utility of GCIs for KR applications has been examined in depth; see, e.g., [RNG93, Rec03, HRG96]. Apart from constraining models of a terminology further without explicitly changing all definitions in the terminology, using GCIs can lead to smaller, more readable TBoxes, and can facilitate the re-use of data in applications of different levels of detail. As a consequence, GCIs are supported by most modern DL systems. An example giving an impression how GCIs can be used in practice will be discussed in Section 1.2.

This completes our overview of the basic features of DL systems. In the remainder of the present work, we will call a DL language together with a TBox formalism, e.g., ordinary, cyclic, or general TBoxes, a *DL formalism*.

Concerning the way domain knowledge is represented by a TBox, we have so far only stated that all terms 'relevant' to the domain of discourse should be contained. The question of what distinguishes an adequate terminology from an inadequate one constitutes a research area on its own, especially the field of *formal ontology*. In the context of formal ontology, the term *ontology* is defined as a *specification of a conceptualization* [Gru93b]. In this context, *conceptualization* usually stands for the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships among them [GN87]. A *specification* gives names to the relevant objects, concepts, entities, and relationships, and constrains the possible interpretations of these names by formal axioms [Gru93b]. Hence, a terminology in the DL sense, i.e., a TBox, can be seen as a special form of an ontology in the formal ontology sense.

In order to cast some light on the question of the adequacy of a terminological representation of domain knowledge, we conclude the present section by presenting some quality criteria for DL-based terminologies from formal ontology.

Quality criteria for terminologies

In [Gru93a], five basic quality criteria for formal ontologies have been proposed, which also apply to DL terminologies. Since these are of interest for the remainder of the present chapter, we summarize the relevant points from [Gru93a].

- *Clarity*: an ontology should effectively communicate the meaning of defined terms. To this end, the meaning of terms defined by the ontology should be fixed independently of computational (or social) contexts, if possible by means of logical axioms. Complete definitions providing all necessary and sufficient conditions are preferred over incomplete ones providing only necessary conditions.
- *Coherence*: the ontology as a whole should be logically consistent, i.e., no implicit consequence derived from explicit information should contradict this information.
- *Extensibility*: it should be possible to extend an ontology or refine parts of it monotonically, i.e., preserving the meaning of already existing terms and preserving existing implicit consequences.
- *Minimal encoding bias*: the ontology should be specified without dependency on a specific symbol-level encoding. For instance, the concept of a physical quantity

should not be defined as a 'double-float value' together with a 'string' but rather as a real number together with a unit of measure. Furthermore, language features of the underlying representation formalism should, if possible, not be selected purely because of computational properties or notational convenience. For instance, language constructs that make reasoning more complex should not be removed from the representation formalism without ascertaining that their expressive power is not vitally needed.

- *Minimal ontological commitment*: the ontology should be minimal in that only terms essential for the intended use of the ontology are defined. As few additional restrictions as possible about the domain of discourse should be included. Moreover, the terms defined in the ontology should be interpreted by the weakest theory, thus admitting the largest number of potential models.

Note that DL terminologies naturally promote several of the above criteria. Concepts can be defined by means of logical axioms, and are interpreted independently of an inference algorithm by a model-theoretic semantics. Since reasoning w.r.t. the majority of DLs is decidable, consistency of DL-TBoxes can be checked automatically. As a fragment of first-order logics, reasoning w.r.t. DLs is monotonic by definition. Furthermore, the semantics of DL knowledge bases is an open-world semantics which is weaker than closed-world semantics. In this sense, minimal ontological commitment is respected.

In the following section, we introduce an application domain in which KR systems are increasingly popular, namely the life sciences and healthcare.

1.2 Knowledge representation for the Life Sciences

History

The term Life Sciences denotes all natural sciences directly related to the study of living organisms, especially medicine, biomedicine, biochemistry, and biology. Given the complexity of the human anatomy, or physiological processes even in single cells, it is not surprising that, long before the advent of computers, these branches of science encountered the problem of representing knowledge in a systematic way.

This can be illustrated in epidemiology, the branch of medicine studying the distribution and determinants of diseases in human populations [RG98]. At first glance, the task seems a quantitative one: to determine how many individuals in a given domain contracted, or died of, which diseases. Without a proper qualitative classification of diseases, however, the data obtained thus is of little value. For instance, in order to assess the incidence of inflammatory diseases it must be known beforehand for every disease occurring in epidemiological records whether it falls into this class or not.

One of the first classification schemes of diseases known internationally was the 'Bertillon Classification of Causes of Death' from 1893 by Jacques Bertillon [Chu00]. It was adopted in revised form by the World Health Organization as the 'International Classification of Diseases (ICD)' which today, in its 10th revision, serves as a main standard for epidemiological data in research and healthcare. Similarly, the College of American Pathologists in 1965 released the 'Systematized Nomenclature of Pathology (SNOP)' which was extended to the 'Systematized Nomenclature of Medicine (SNOMED)' [CRP+93] in 1974. Intended for use not only in pathology, the SNOMED classification was not limited to diseases but also contained anatomical structures and medical procedures. An electronic version was released in 1977, which might be regarded as an ancestor of general-purpose medical KR systems. Since then, SNOMED has been revised continually and nowadays exists in the

form of a DL terminology [Spa01]. In its current release, SNOMED contains 379,691 concepts and 52 roles [BLS05].

In 1992, the European project GALEN[3] has been launched in order to facilitate the integration of medical information systems by means of a common reference model for medical terminology. In contrast to the College of American Pathologists, which translated an already existing classification system into DL, the strategy of the GALEN project was to develop a suitable KR formalism *before* building the actual terminology. To this end, requirements specific to the medical domain were assessed and an adequate terminology language was developed, the 'GALEN Representation and Integration Language (GRAIL)', based on a DL [RH97]. As the underlying requirements are relevant to KR in the Life Sciences in general, we shall come back to the major points observed by the GALEN project. A unique feature of the GALEN terminology is that it makes use of GCIs instead of purely relying on concept definitions to represent medical knowledge. Currently, GALEN comprises 2,740 concept definitions and 1,214 GCIs [BLS05].

In biology, one motivation behind classifying relevant terms by means of a KR system arose from the need to share information among large genome databases. For instance, in 2000, the genome projects FlyBase [Con98], Mouse Genome Informatics [BER+98], and the Saccharomyces Genome Database [CAB+98] founded the Gene Ontology (GO) Consortium with the goal to produce a '*structured*, precisely *defined*, common, *controlled vocabulary* for describing the roles of genes and gene products in any organism' [Con00]. By means of the GO, gene products in multiple organisms can be annotated in such a way as to enable uniform queries over their properties, and to combine knowledge stored in separate genome databases. The GO corresponds to a DL knowledge base of 18,803 concepts with only one transitive role [BLS05].

KR requirements for the Life Sciences

The above mentioned biomedical KR projects have cast light on potential KR requirements specific to the biomedical domain. In the following, we present the lessons learnt, especially in the context of the GALEN project.

Firstly, a KR formalism for the biomedical domain should support general axioms, i.e., GCIs in the DL case, which proved useful especially for three purposes.

- *To indicate the status of objects*: instead of introducing several definitions for the same concept in different states, e.g., normal insulin secretion, abnormal but harmless insulin secretion, and pathological insulin secretion, only insulin secretion is defined while the status, i.e., normal, abnormal but harmless, and pathological, is implied by GCIs of the form $\ldots$ (necessary condition) $\ldots \sqsubseteq \exists$has_status.pathological.

- *To bridge levels of granularity and to add implied meaning*: A classical example [HRG96] is to use a GCI like

$$\mathsf{ulcer} \sqcap \exists \mathsf{has\_loc.stomach} \sqsubseteq \mathsf{ulcer} \sqcap \exists \mathsf{has\_loc.}(\mathsf{lining} \sqcap \exists \mathsf{is\_part\_of.stomach})$$

  to render the description of a defined concept for 'ulcer of stomach' more precisely to 'ulcer of lining of stomach' if it is known that ulcer of the stomach is specific to the lining of the stomach.

- *To add spacial information*: For instance, in order to define that any hollow body structure defines a cavity without changing the definitions of all structures in ques-

[3]Generalized Architecture for Languages, Enzyclopaedias and Nomenclatures in Medicine, see also `http://www.OpenGALEN.org`.

tion, a GCI similar to the following is added.

$$\mathsf{BodyStructure} \sqcap \exists \mathsf{has\_topology.has\_absolute\_state.Hollow} \sqsubseteq \exists \mathsf{defines\_space.BodyCavity}$$

Note that in GALEN, the topology of a body structures can change. This is modeled by means of the roles has_absolute_state and has_change_in_state, via which a change of state can be expressed.

Moreover, it has been argued that the use of GCIs facilitates the re-use of data in applications of different levels of detail [HRG96, RBG+97]. For instance, in some contexts additional information, e.g., the precise location of ulcer in the stomach, might be unnecessary. In this case, one can simply ignore GCIs by which additional information is supplied and focus on the un-refined definitions in question. Note that distinctions of level of detail occur naturally in the medical domain. For instance, an epidemiologist usually only documents a 'fracture of the finger' when a surgeon has treated a 'fracture of the proximal phalanx of the fourth left finger'.

Secondly, a biomedical KR formalism should allow to declare relations transitive.

- Transitive relations occur naturally in the biomedical domain. Describing the anatomy of complex organisms, for instance, necessarily produces deep partonomic hierarchies that are transitive in nature, e.g., mitral valve cusp part-of mitral valve part-of heart part-of cardiovascular system. The same holds for causality relations between processes in living organisms, e.g., severe hypertension causes coronary hypertrophy causes insufficient coronary perfusion, and also for medical procedures.

- Without support of transitive relations, either crucial information is lost, e.g., mitral valve cusp part of heart, compromising the quality of reasoning, or the transitive closure of all relevant relations must be added explicitly to the knowledge base. The latter option, however, seems inexpedient for at least two reasons. Firstly, the often enormous size of biomedical knowledge bases already poses a challenge to KR systems, so that adding, e.g., all explicit parthood relations might be impracticable. Secondly, removing or inserting new concepts into a knowledge base becomes much more difficult if hundreds (or more) of relations to other concepts must be deleted or established.

- Simulating transitive relations by means of auxiliary concepts [SH02] also increases the size of a knowledge base severely.

Thirdly, a KR formalism for the biomedical domain should support so-called *right-identities*. Written in DL syntax, a right-identity is an axiom of the form $r \circ s \sqsubseteq r$, where r, s are role names and $\circ$ represents a composition operator on roles. Semantically, a model $\mathcal{I}$ satisfies a right-identity of the above form if and only if the following holds for all elements x, y, z of $\Delta^{\mathcal{I}}$: if x, y are related via $r^{\mathcal{I}}$ and y, z are related via $s^{\mathcal{I}}$ then x is related to z via $r^{\mathcal{I}}$.

As an example from pathology, consider a hematoma in the temporal lobe of the brain. The hematoma is located in the temporal lobe without being a part of it while the temporal lobe is a part of the brain and not located in it. Clearly, we would like a KR system to infer that the hematoma is located in the brain. However, using transitive relations 'has-location' and 'part-of' does not suffice because both relations occur exactly once. On the other hand, manually adding all missing relations to the knowledge base causes similar problems as manually adding the transitive closure: the knowledge base is likely to increase in size severely, and maintenance is complicated. The desired inference can easily be expressed by means of right-identities. In case of a DL knowledge base, it suffices to add

the axiom has_location ∘ part_of ⊑ has_location.[4] In the biomedical domain, many similar examples exist. For instance, a finding at a part of an anatomical structure often implies a finding at the whole, e.g., a lesion at the hand implies a lesion at the upper extremities. The relevance of right-identities for KR in the biomedical domain has also been observed in [Spa00]. Note that right-identities can express transitivity of relations. Written in DL syntax, the right-identity $r \circ r \sqsubseteq r$ makes r a transitive role.

In contrast to the above, several other constructs known from more expressive DLs proved to be of little value for biomedical classification systems. As observed in [RH97] in the context of the GALEN project, the fraction of terms in common medical nomenclatures requiring explicit cardinality restrictions in their definitions is less than 1%. For similar reasons, disjunction, negation, and universal quantification[5] have not been included in the GRAIL language.

It should be noted that some features desirable from a medical KR point of view are not supported in GRAIL. Firstly, it is not possible to faithfully represent contiguous anatomical structures [RH97]. For instance, the gastrointestinal tract of the human body is a single contiguous structure, leading from the mouth via the oesophagus to the stomach and so on, without any strict boundaries between its components. The notion of contiguity can only be represented to some extent by the weaker part-of relation or subrelations thereof. Similar problems occur with respect to the vascular system, where no structural boundaries exist at the transitions of many vessels. E.g., the external iliac artery becomes the femoral artery without any structural or topological border. Similar problems are faced when representing surface regions of the human body.

A second desirable feature not present in any of the biomedical knowledge bases cited above are same-as references in concept definitions [RH97]. For instance, the above ulcer example could be generalized to all gastrointestinal organs as follows: if an ulcer occurs in an organ of the gastrointestinal tract then it occurs in the lining of *the same* organ. This same-as constraint cannot be represented faithfully without naming all relevant organs, i.e., oesophagus, stomach, etc., and introducing specialization axioms for every single one.

To sum up our overview of the Life Sciences as an application domain for KR systems, we have seen that here, as a natural consequence of the complexity of the domain, knowledge bases usually comprise a very large number of concepts, sometimes more than 300,000, but on the other hand are defined over relatively inexpressive KR formalisms that correspond to DL terminologies. The value of general axioms, transitive relations and right-identities has been explicitly noted while many other common constructors, such as disjunction or general negation, do not offer a substantial benefit. Efficient reasoning in relatively inexpressive DLs w.r.t. GCIs is studied in depth in Chapter 3. For an overview of the existing results on reasoning w.r.t. GCIs, see Section 1.5.1. Our contributions to this topic are summarized in Section 1.6.1.

The overview of their history also shows that biomedical knowledge bases are usually developed by entire groups of domain experts over several years. In order to maintain the quality of the resulting knowledge bases, e.g., avoid inconsistencies, redundancies, modeling errors, etc., an appropriate maintenance methodology is required as well as reliable maintenance tools for their support. In Section 1.4, we will return to the topic of knowledge maintenance and argue in favor of so-called *non-standard inference services* as the tools of choice to support the construction and maintenance of DL knowledge bases in a formally well-defined way.

Apart from the Life Sciences, there is another application domain of DL based KR systems that has recently attracted considerable attention: the Semantic Web. In the following

[4] In GRAIL syntax, the axiom reads: has_location *specialisedBy* part_of.
[5] For a formal definition of these constructs, see Definitions 2.1.1 and 2.1.3.

section, we introduce the main ideas related to this domain and the connection to the main topics of our work.

1.3 Knowledge representation for the Semantic Web

In the seminal paper [BLHL01], a vision of the future World Wide Web (Web for short) has been proposed that centers around the idea to annotate Web content by a formal representation of its meaning and thereby enable Web-based software to greatly increase the complexity of tasks they can perform without human guidance.

The authors have argued that an elementary prerequisite for the realization of this vision is an ontology by means of which every Web-resource, every item of data occurring on a Web-page, and every Web-service can be assigned a formally defined meaning, or type. In the context of the Semantic Web, the term ontology is usually strongly preferred over 'terminology' or 'knowledge base'. One motive behind this might be to stress that the interesting aspect of a knowledge base is the *meaning* given to objects and their interrelations rather than the syntactical formalism by which terms can be constructed. The following example illustrates how an ontology might be used to annotate Web-content.

Example 1.3.1 The homepage of one of the authors of [BLHL01], J. Hendler, demonstrates a Semantic Web annotation of a classical Web-page. An excerpt of the source-code shows the following tags.

```
<INSTANCE KEY="http://www.cs.umd.edu/users/hendler/">
<USE-ONTOLOGY ID="cs-dept-ontology" VERSION="1.0" PREFIX="cs"
          URL=" http://www.cs.umd.edu/projects/plus/SHOE/cs.html" />
<CATEGORY NAME="cs.Professor" FOR="http://www.cs.umd.edu/users/hendler/"/>
<RELATION NAME="cs.name">
          <ARG POS=2 VALUE="Dr. James Hendler"></RELATION>
<RELATION NAME="cs.doctoralDegreeFrom">
          <ARG POS=1 VALUE="http://www.cs.umd.edu/users/hendler/">
          <ARG POS=2 VALUE="http://www.brown.edu"></RELATION>
```

The annotation states that the web page provides information about an instance of the category[6] *cs.Professor* defined in the ontology cs-dept-ontology. This instance is in relation cs.name to the string 'Dr. James Hendler' and in relation cs.doctoralDegreeFrom to an object specified by the URL 'http://www.brown.edu'. In the relevant ontology, the category cs.Professor is a sub-category of Faculty, which specializes Worker, which is a sub-category of Person. Moreover, type restrictions for the occurring relations imply that cs.name links an instance of the category Person to a string, and that cs.doctoralDegreeFrom similarly links an instance of Person to University, a sub-category of Organization. □

As argued in [BLHL01], annotations of the kind shown above enhance the capabilities of automated services offered over the Web. For instance, searching the Web for the homepage of James Hendler usually cannot be done in any smarter way than just looking for Web-pages containing the phrase 'James Hendler'. It cannot be assumed, and in fact is not the case, that the homepage in question contains the phrase 'homepage'. Such a search, however, will produce a significant number of pages on which the search phrase occurs for reasons other than designating the homepage of a person under that name.

With an annotation and an ontology of the above kind in the background, a search could be conducted for a Web-page *about a professor* under the *name* 'James Hendler'. Thus,

[6] The above mentioned ontology is represented in the frame-based formalism SHOE [HHL99], where so-called 'categories' correspond to the notion of concepts in a DL-knowledge base.

Web-pages could be excluded on which the relevant name is only mentioned in some other context, or which do not belong to a professor. Note that even in this simple case, the query system has to *infer* from the ontology that a professor is especially a person because the relation cs.name is restricted to instances of the category Person.

In the context of Semantic Web research, it is emphasized that not only Web-pages, but also Web-*services* will be described by means of ontologies [MM03, MSZ01]. Hence, search engines, Internet shops, on-line databases, and other businesses are supposed to formally describe their offered services or products. The intended benefit is that complex tasks involving data entry on Web-sites and interpretation of query results will be executed partly autonomously by Semantic Web Agents [Hen01].

The vision of the Semantic Web sketched above clearly poses the question of an adequate KR formalism for the ontologies underlying the Semantic Web. In 2004, the W3C Consortium responsible for the management of development efforts related to the Semantic Web recommended the DL-based formalism OWL[7] [HPSvH03] for this purpose. The KR formalism OWL is the result of the union of the European research project OIL [FvHH+01] and the US research project DAML [Shi01]; see [HPSvH03] for an overview.

In fact, OWL does not only provide one but three KR formalisms of increasing expressive power, OWL-Lite, OWL-DL, and OWL-Full. Without going into detail, let it suffice to say that OWL-Lite corresponds to the very expressive DL $\mathcal{SHIF}(D)$ with general TBoxes and ABoxes. OWL-DL corresponds to the more expressive DL $\mathcal{SHOIN}(D)$ and supports general TBoxes and ABoxes likewise. Finally, OWL-Full extends the expressive power of OWL-DL by admitting constructs from RDF[8] and RDFS[9]. Since the semantics of RDF and RDFS cannot be expressed properly in first-order logics, OWL-Full cannot be viewed as a DL any more.

A peculiarity of all OWL dialects is their syntax. In contrast to the usual DL syntax sketched previously, OWL ontologies are represented in RDF-syntax derived from XML. The following example shows a concept definition in OWL.

Example 1.3.2 Recall our simple definition of the concept *Parent* from Example 1.1.1. The corresponding definition in an OWL ontology in standard syntax would be as follows.

```
<owl:Class rdf:ID="Parent">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Person" />
    <owl:Restriction>
      <owl:onProperty rdf:resource="#has_child" />
      <owl:hasValue rdf:resource="#Person" />
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

As described in the OWL Web Ontology Language Guide[10], OWL concepts are called 'classes', roles are called 'properties', and conjunction is called 'intersection'. An existential restriction is denoted by a 'restriction' of the type 'hasValue'—as opposed to 'allValuesFrom' for value restrictions. □

The fact that a crucial component of the Semantic Web is supposed to be based on DL systems is currently a strong motivation for DL research and points to a potentially major

[7] Ontology Web Language, see also http://www.w3.org/2004/OWL/.
[8] Resource Description Framework, see also http://www.w3.org/RDF/.
[9] RDF Schema, see also http://www.w3.org/TR/rdf-schema/.
[10] See http://www.w3.org/TR/owl-guide/.

field of future applications. Given the amount of data available even on the Web of today, it seems clear that the Semantic Web poses a major challenge to the efficiency of reasoning w.r.t. very large knowledge bases. It should, however, be noted that the most common case of using the Semantic Web, i.e., searching for information, means looking for assertional information stored in the ABox-part of a DL knowledge base. It does not necessitate actively classifying ontologies, but only passively using their, probably pre-computed, subsumption hierarchy. Consequently, this kind of interaction with the Semantic Web might be handled by a technology not that far remote from the highly optimized databases behind the Web of today.

Another consequence of the Semantic Web vision for the research area DL-based KR is a demand for reasoners and knowledge editors capable of handling ontologies, or TBoxes, of considerable size. It is also to be expected that ontologies for the Semantic Web will be designed by a very large number of domain experts and will evolve over time. Just as pointed out in the context of the application domain Life Sciences in Section 1.2, this motivates the question of a standard methodology for the maintenance of knowledge bases and appropriate tools for this purpose. In the following section, the tasks related to knowledge maintenance will be discussed in further detail.

Note that the results of the present work are relevant to the Semantic Web in that they examine how tractable reasoning and expressive DL formalisms can be reconciled, and in that they discuss non-standard inference services by which maintenance of DL knowledge bases can be supported in a formally well-defined way. On the other hand, we would like to stress that our work is not specifically tailored to Semantic Web applications and does not further examine the specific requirements of this emerging application domain.

1.4 Knowledge maintenance

In our overview of DL-based KR, we have so far taken for granted that knowledge bases for a given domain of discourse are readily available. The task of creating a knowledge base and maintaining it over time, however, poses challenges that need to be discussed in further detail. In particular, extending a DL terminology in an unstructured way puts at risk the quality criteria from Section 1.1. In order to support users in this task, we present an approach to knowledge maintenance that goes beyond mere tools embedded in the user interface of knowledge editors.

One distinguishing property that has led to DLs being preferred over other knowledge representation formalisms is their well-defined model theoretic semantics. For every concept in the terminology we can exactly and unambiguously state what this concept means as such. It is only coherent to preserve this quality in the design of reliable tools aimed to support users extending or maintaining DL terminologies. Hence, one starts by finding appropriate well-defined logical inference services whose effect can be stated exactly, unambiguously, and independently of the algorithm implementing it. Then, one embeds an implementation of these inference services in a DL system and provides a user-interface for the relevant ontology editor, thereby providing the user with the desired maintenance tools.

1.4.1 Constructing knowledge bases

We consider three typical scenarios in which a user, i.e., a domain expert, wants to extend an existing DL-TBox. For each of these, we introduce so-called *non-standard* inference services for DL-TBoxes and show how these can be used to support the user. The attribute

'non-standard' stems from the fact that these inference services are not supported by conventional DL systems and cannot even be simulated by standard inference services, such as satisfiability or subsumption. The relevant scenarios for the extension of DL-TBoxes are *top-down* extension, *bottom-up* extension and extension by *import*.

Top-down extension

In the most straightforward case, the user simply is faced with a TBox in which he finds only concepts too general to represent a particular new notion. In this case, the user locates a suitable structurally similar concept in the terminology and produces a specialization of it by re-using the definition of the general concept. To this end, concept or role names are modified and additional constraints included. This approach is sometimes referred to as 'top-down' [BT01] approach.

The key question for this approach is: how to locate a structurally similar concept in the TBox from whose definition the new concept can be derived? Note that the newly defined concept is inserted into the subsumption hierarchy of the TBox automatically by the DL system and not manually by the user. Therefore, our query for an appropriate concept is driven only by the desired structure of the new definition and not by the expected position in the subsumption hierarchy.

In order to find concepts of a certain syntactic structure in DL-TBoxes, we propose the non-standard inference service *matching*. A *matching problem (modulo equivalence)* consists of a concept description C and a *concept pattern*, i.e., a concept description in which, additionally, *variables* may occur in places of atomic concepts. A *matcher* of a given matching problem is a substitution assigning concept descriptions to the variables in the concept pattern in such a way that equivalence to C holds. A matching problem can be viewed either as a decision problem, where the task is just to decide whether or not a matcher exists, or as a computation problem, where all (interesting) matchers are to be computed. Matching problems can also be defined *modulo subsumption*. In this case, a matcher only has to make the input concept pattern *subsume* the input concept C. Note, however, that matching modulo subsumption can be expressed by matching modulo equivalence because a subsumption $C_1 \sqsubseteq C_2$ holds if and only if $C_1 \equiv C_1 \sqcap C_2$.

Matching can be utilized as a retrieval mechanism over DL-TBoxes in a straightforward way. The user specifies a concept pattern with the syntactic structure he has in mind. The DL system then retrieves all concepts in the TBox for which a matcher w.r.t. the given pattern exists. From the resulting set, the user selects the one most suitable for his purpose and modifies it, producing a new definition to be added to the TBox.

The fact that variables in concept patterns are named, in contrast to, e.g., wildcards ('$*$') known from standard database queries, allows to search the TBox for concepts with specific structural properties. As an example from engineering, consider a TBox containing concept definitions representing technical devices. Matching against the pattern $\mathsf{Device} \sqcap \exists \mathsf{has\_unit}.X \sqcap \exists \mathsf{has\_backup\_unit}.X$ returns only those concepts which represent devices containing a sub-unit and a backup unit *of the same type*. We shall discuss later on how to rule out too general solutions in such cases, e.g., matchers that replace X by the top-concept.

For an overview of origin and existing results on DL-matching algorithms, see Section 1.5.2. Our new contributions to matching are summarized in Section 1.6.2.

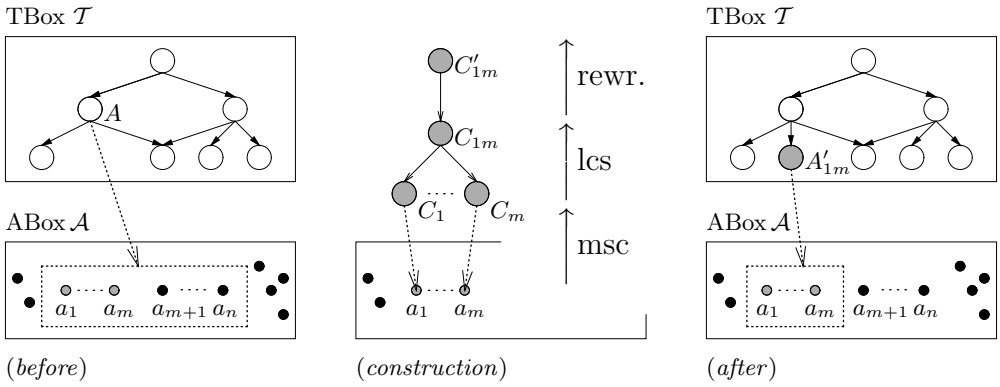


Figure 1.4.1: Bottom-up extension of knowledge bases

Bottom-up extension

In order to motivate bottom-up extensions, consider a DL knowledge base consisting of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, the TBox containing concept definitions and the ABox assertions for individuals. In our case, the domain expert detects the following scenario for one concept A defined in $\mathcal{T}$: in the TBox, A is the most specific concept representing the individuals $a_1, \ldots, a_m, a_{m+1}, \ldots, a_n$. However, the domain expert feels that the subset $\{a_1, \ldots, a_m\}$ of individuals should be more adequately represented by a concept more specific than A. In other words, he wants to extend the TBox by a new concept which is more specifically tailored to represent the properties of the individuals $a_1, \ldots, a_m$.

This situation is illustrated schematically in Figure 1.4.1 (*before*). The upper box shows the subsumption hierarchy of the TBox $\mathcal{T}$, where every circle represents one concept defined in $\mathcal{T}$ and every edge a direct subsumption relation. Every dot in the lower box represents an individual name occurring in the ABox $\mathcal{A}$. Relations between individuals are not shown. Dotted lines show concept-instance relationships, i.e., all instances of A are contained in the dotted box inside the ABox $\mathcal{A}$.

This extension task can be accomplished by a combination of the three non-standard inference services, *most-specific concept (msc)*, *least-common subsumer (lcs)*, and *rewriting*. Intuitively, the msc is used to construct a concept description C_i for every single individual a_i. The lcs then extracts the commonalities of all C_i, producing a new concept description C_{1m}. This is simplified to C'_{1m} by rewriting. Describing this strategy in more detail, we begin by defining the msc.

Given an ABox-individual a described w.r.t. a TBox $\mathcal{T}$, the msc of a is defined as the most specific concept description a is an instance of that can be constructed w.r.t. $\mathcal{T}$. Hence, the msc does not retrieve a concept from $\mathcal{T}$, but rather *constructs* a new concept description representing a as specific as possible. In the case of the above example, the DL system would, in the first step, construct the concept descriptions $C_1 := \mathsf{msc}(a_1)$, ..., $C_m := \mathsf{msc}(a_m)$.

In the second step, the lcs inference is used. The lcs of a set of concept descriptions $C_1, \ldots, C_m$ w.r.t. a TBox $\mathcal{T}$ is a concept description C_{1m} that is more general than every C_i, and is the most specific one with this property w.r.t. $\mathcal{T}$. Note that, again, the lcs is not merely obtained from the TBox by retrieval but actually constructed. In the above example, the DL system computes C_{1m} as $\mathsf{lcs}\{\mathsf{msc}(a_1), \ldots, \mathsf{msc}(a_m)\}$, a concept description representing the common properties of the individuals $a_1, \ldots, a_m$.

The last computation step of our bottom-up strategy, *rewriting*, is aimed at simplifying

the concept description C_{1m} obtained from the lcs computation in the second step. To this end, the DL system tries to find sub-descriptions of C_{1m} for which a name has been defined in the TBox $\mathcal{T}$. In such a case, the sub-description is replaced by the corresponding defined name from $\mathcal{T}$, producing the concept description C'_{1m}. The construction process is illustrated in Figure 1.4.1 (*construction*). Again, dotted lines represent instance relations and solid edges subsumption relations. Note that the direction of construction is upwards while subsumption and instance relations point downwards.

Finally, the simplified concept description C'_{1m} is presented to the domain expert for inspection. Probably after applying some manual modifications, the domain expert provides a new name A'_{1m} and extends the TBox by the new definition $A'_{1m} \equiv C'_{1m}$. As a result, the most specific concept representing the ABox individuals $a_1, \ldots, a_m$ is now A'_{1m}. The resulting knowledge base is illustrated in Figure 1.4.1 (*after*). This approach is usually called 'bottom-up' approach [BKM99, BT01].

A slightly simpler instance of the bottom-up extension strategy refers only to the TBox. In this case, the domains experts finds a concept A in the TBox with a large number of concepts $A_1, \ldots, A_m, A_{m+1}, \ldots, A_n$ directly subsumed by A. However, the domain expert feels that the concepts, $A_1, \ldots, A_m$ deserve a super-concept more specific than A. By means of the lcs inference and rewriting, the DL system could similarly present the lcs of $A_1, \ldots, A_m$ as a candidate for the super-concept in question. The domain expert could then again inspect the result and add the corresponding definition to the TBox, thus moving from a wide and shallow subsumption hierarchy to a deeper and narrower one. One advantage of the latter is that browsing through the subsumption hierarchy of a TBox is simplified.

Although three non-standard inferences play a role in the bottom-up approach described above, the present work is, among these, only concerned with the lcs. For an overview of existing results on the lcs, see Section 1.5.2. Our new contributions are summarized in Section 1.6.2.

Extension by import

In the last case to discuss, the domain expert wants to extend a terminology by a concept definition he has found in another terminology defined over a different DL language. In this case, there is a mechanism needed to import, or, more precisely, to translate the concept into the language of the destination terminology.

This task can be facilitated by the non-standard inference *approximation*. Given a concept description C defined over a source DL language $\mathcal{L}$, and given a destination DL language $\mathcal{L}'$, the (upper) $\mathcal{L}'$-approximation of C is defined as the most specific concept description C' expressible in $\mathcal{L}'$ that subsumes C. Intuitively, C' is the translation of C into $\mathcal{L}'$ with a minimized loss of information. In particular, if C is expressible in $\mathcal{L}'$ then C and C' are equivalent.

A natural generalization of the above scenario is the translation of an entire DL knowledge base into another DL language. One motive behind this might simply be to merge the translated knowledge base with another one represented in the destination language. Alternatively, the translation might serve as a starting point to build up a new knowledge base, possibly using language constructs or TBox constructs unavailable in the source DL or TBox formalism. For instance, in comparison to the source knowledge base, the destination language might be lacking disjunction but providing GCIs in the underlying TBox formalism.

Existing results for approximation are discussed in Section 1.5.2. Our new contributions to this non-standard inference are summarized in Section 1.6.2.

1.4.2 Other maintenance tasks

The above mentioned non-standard inference services, i.e., matching, least-common subsumer and approximation, have several more applications related to knowledge maintenance tasks or support of domain experts using DL systems. To provide further motivation for their study, we briefly describe other application scenarios for each non-standard inference in question.

Detecting redundancies

As the examples in Section 1.2 suggest, real-world knowledge bases are usually large, containing thousands of concepts, and are often maintained by a large number of domain experts over a period of several years. In this situation, different knowledge engineers might extend their terminology by similarly defined, yet different, concepts. This would compromise clarity, one of our quality criteria for terminologies. Obviously, standard reasoning services can be used to find such cases in DL knowledge bases to some extent: concepts coincide in the subsumption hierarchy if they are equivalent.

Apart from perfect equivalence, however, other more general sources of redundancy apply to our scenario. Consider a case where two domain experts independently intend to introduce the same concept but define it with reference to different concept names. For instance, recall our trivial TBox $\mathcal{T}$ from Example 1.1.1, in which *Parent* is defined. Another knowledge engineer might have defined a similar concept, e.g., *FatherOrMother* by $\mathsf{Human} \sqcap \exists \mathsf{has\_child}.\mathsf{Human}$, which due to the difference between the atomic concepts Person and Human is not equivalent to *Parent*. Clearly, both definitions attempt to represent the same notion.

By means of matching, such redundancies can sometimes be detected. For instance, after extension by the definition of *FatherOrMother*, the TBox $\mathcal{T}$ could be queried for concepts matching the concept pattern $X \sqcap \exists \mathsf{has\_child}.X$. The resulting set, i.e., {*Parent*, *FatherOrMother*}, could then be used to resolve the problem. A similar case is described in [Küs01]. Note, however, that concepts are not generally redundant if they can be matched against the same pattern. Nonetheless, in search of redundancies matching can provide valuable clues.

Integrating knowledge bases

As in the case of the Gene Ontology mentioned above, knowledge bases can be a by-product of the merger of organizations, thus creating a need for a coherent representation of their combined domain knowledge. Assuming that the relevant knowledge sources are already represented by, or could be translated into, DL-TBoxes, the question is how to integrate them into one coherent TBox.

It has been shown in [BK00b] that, to some extent, the answer can be matching (under side conditions, see Section 1.5.2). More precisely, matching is used to find so-called conflict free mappings between the TBoxes to merge, where conflicts are defined only taking into account the structure of a concept definition, disregarding the actual names of atomic concepts and roles. For related work on the topic of integrating knowledge bases, see Section 1.7.2.

Supporting inexperienced users

From a KR perspective, the flexibility of DL knowledge bases, admitting arbitrarily complex concept descriptions in definitions or even in GCIs, clearly is an asset. Nevertheless,

users of DL systems are typically domain experts with limited initial knowledge representation expertise. Such users may have difficulties to understand and make use of the full expressive power of the DL language underlying the DL system.

In order to provide a simplified view on the knowledge base, some DL systems have been equipped with a simplified frame-based user interface built on top of a more powerful knowledge editor. Examples of systems with this dual architecture are the TAMBIS system [BBB+98] and the knowledge editors OilEd [BHGS01] and Protégé [GMF+03]. In particular, observe that the sub-window labeled 'Class Editor' in Figure 1.1.1 shows such a simplified interface. On several occasions, these knowledge editors present concept descriptions to the user for editing, inspection, or as a solution of inference problems. Operating in the simplified frame-based mode, these concept descriptions need not always fit into the restricted slots of the frames representing the concepts in question.

In such cases, approximation might serve as a means to translate the original concept description into a restricted language compatible with the frame-based representation, thus adapting to the users' level of expertise. Furthermore, the aspects not captured by the frame-based representation could be computed for further inspection by means of a difference operator for concept descriptions [BKT02b].

Non-standard inferences for expressive DLs

In the context of the bottom-up extension of DL knowledge bases described in Section 1.4.1, we have presented the non-standard inference least-common subsumer as a means to extract the commonalities of concept descriptions. Given concept descriptions $C_1, \ldots, C_n$, their lcs is defined as the most specific (w.r.t. subsumption) concept description subsuming every C_i. In DLs providing a disjunction operator ($\sqcup$), this definition can be satisfied trivially by the disjunction of all C_i. The disjunction, however, does not explicitly show the commonalities of the input concepts, so that no further insight is obtained by a domain expert inspecting the disjunction.

This can be overcome to some extent by approximation. Instead of computing the lcs of $C_1, \ldots, C_n$ at once, the commonalities of the relevant concepts can be made explicit by first approximating every C_i in a DL language *without* disjunction and then computing the lcs. Note, however, that information might be lost in this process due to approximation.

A similar strategy can be used to solve matching problems in DLs too expressive for currently available matching algorithms. For several reasons, matching is, like the lcs, unavailable in DLs providing disjunction or general negation, see Section 1.5.2 for details.

1.5 Relevant existing results

In order to specify precisely the contribution of the present work, we begin by relating currently existing results on the topics directly relevant to us, especially reasoning w.r.t. general TBoxes and non-standard inferences. Related work motivated by similar application scenarios is discussed in Section 1.7. For a broader overview over the field of DL research, see, e.g., [NB03, BN03].

In the remainder of this chapter, it will be necessary to refer to some notions from DL research, especially specific DLs and TBox formalisms, that have not been introduced formally in our overview so far. Readers not familiar with the common vocabulary of the DL domain might refer to Section 2.1 for a formal definition of syntax and semantics of all relevant DLs, TBox formalisms and semantics mentioned in the remainder of this chapter.

1.5.1 Standard inferences

Initially, DL research was motivated by the goal to provide a formal semantics for semantic networks and frame systems, see Section 1.1. In this context, property arcs in semantic networks and slots in frames were interpreted as universal restrictions on the relevant relations. For that reason, the first complexity analyses were carried out for DLs providing value restrictions.

As soon as terminologies were taken into account, however, reasoning even w.r.t. very small DLs proved to be intractable. Even with the simplest form of acyclic TBoxes, subsumption in the small DL $\mathcal{FL}_0$, providing only conjunction and value restriction, is co-NP-hard [Neb90]. The hardness result carries over to every extension of $\mathcal{FL}_0$. Admitting cycles in TBoxes even increases the complexity to PSPACE-completeness, both for fixed point semantics [Baa96] and descriptive semantics [KN03].

Due to the above mentioned intractability results for very basic DLs even w.r.t. acyclic TBoxes, the tractability question for more complex TBox formalisms has hardly been considered. On the contrary, research on general TBoxes has been mainly focused on very expressive DLs, reaching as far as, e.g., $\mathcal{ALCNR}$ [BDS93] and $\mathcal{SHIQ}$ [HST99], where deciding subsumption of concepts w.r.t. general TBoxes is EXPTIME-hard. More recently, a subsumption algorithm for $\mathcal{SHOIQ}$ has been presented [HS05] which can be used to decide subsumption in the slightly weaker OWL-DL, one of the DL languages recommended as a standard for the Semantic Web. In $\mathcal{SHOIQ}$, reasoning is known to be even NEXPTIME-complete [Tob00].

Research in the direction of less expressive DLs suggested that, for reasoning w.r.t. general TBoxes, the EXPTIME lower bound would not give way easily. In [GMWK02], the problem has been shown to remain EXPTIME-complete for $\mathcal{FLE}$, a DL providing only top concept, conjunction, value restriction and existential restriction. The same holds for the small DL $\mathcal{AL}$ which provides conjunction, value and unqualified existential restriction, and primitive negation [Don03]. Even restricting general TBoxes to primitive ones, where only concept names are admitted on left-hand sides of GCIs, hardly improves the complexity of reasoning. As shown in [Cal96], reasoning w.r.t. primitive general TBoxes remains EXPTIME-complete for the DLs $\mathcal{ALC}$ and $\mathcal{ALU}$, and is PSPACE-complete for $\mathcal{ALE}$.

In the light of the above intractability results, it came as a surprise that subsumption w.r.t. cyclic $\mathcal{EL}$-TBoxes is tractable [Baa03b], i.e., decidable in polynomial time. In particular, tractability of subsumption has not only been shown for (standard) descriptive semantics, but also both for greatest-fixedpoint and least-fixedpoint semantics.

1.5.2 Non-standard inferences

While most standard inference problems are decision problems used to reason over an already existing DL knowledge base, most *non-standard* inference problems are computation problems tailored to the purpose of supporting build-up and maintenance of knowledge bases. For an overview over the topic of non-standard inferences in DL, see [Küs01]. For the scope of the present work, the non-standard inferences matching, least-common subsumer, and approximation will be of particular interest.

Matching

Matching in DL has first been utilized as a pruning mechanism in the context of the DL system CLASSIC developed at AT&T [BBMR89, BMPS+91]. In industrial applications, the concept definitions contained in CLASSIC knowledge bases turned out to be much too large

to be presented to human users [McG96]. In order to reduce the amount of information, users could specify 'interesting' aspects of a concept by means of a concept pattern, causing the system to display only those parts of concept definitions that could be matched against the pattern [BM96]. For instance, specifying the pattern $\exists \mathsf{has\_unit}.X$ causes every concept definition to be pruned to existential restrictions referring to the role has_unit.

The matching algorithm implemented for the CLASSIC system could not handle arbitrary patterns and was incomplete even on restricted ones [Küs01]. The first complete, truly general DL-matching algorithm has been presented in [BN98] for the small DL $\mathcal{FL}_0$. Subsequently, matching algorithms for several other DLs have been developed: for $\mathcal{ALN}$ and sublanguages [BKBM99], for $\mathcal{EL}$ and $\mathcal{ALE}$ [BK00a]. It has been shown that deciding the solvability of $\mathcal{EL}$-matching problems modulo equivalence is NP-complete, while the decision problem is tractable for $\mathcal{EL}$-matching problems modulo subsumption. Both decision problems are NP-complete in $\mathcal{ALE}$. Furthermore, sound and complete matching algorithms for $\mathcal{ALN}^*$ and $\mathcal{ALNS}$ have been presented in [Küs01]. Despite their strong practical motivation, none of the above algorithms have initially been implemented. The first implementations are part of the present work, see Section 1.6.2.

Usually, even in relatively inexpressive DLs, matching problems can have many solutions. As an example, consider the simple matching problem $P \sqcap Q \equiv^? X \sqcap Y$ with atomic concepts P, Q and variables X, Y. This matching problem has $2^3 - 1$ solutions, two of them being $\{X \mapsto P, Y \mapsto Q\}$ and $\{X \mapsto P \sqcap Q, Y \mapsto P \sqcap Q\}$. In order to further specify desired solutions of matching problems by imposing additional restrictions on admissible values for variables, *side conditions* have been proposed [MRI95, McG96] and have been examined in depth for the DL $\mathcal{ALN}$ and sublanguages in [BKBM99].

A side condition is an expression of the form $X \sqsubseteq D$, called *subsumption condition*, or $X \sqsubset D$, called *strict* subsumption condition, where X is a variable and D a concept *pattern*. A matching problem *under side conditions* is a matching problem together with at most one side condition for every variable occurring in the matching problem. A matcher σ satisfies a subsumption condition $X \sqsubseteq D$ if and only if $\sigma(X) \sqsubseteq \sigma(D)$ and analogous for strict subsumption conditions. Hence, side conditions correspond to equation systems imposing further constraints between the variables in the matching problem. In the above example, the strict subsumption condition $X \sqsubset P \sqcap Y$ reduces the number of admissible solutions to one, namely $\{X \mapsto P \sqcap Q, Y \mapsto P\}$.

It has been shown in [BKBM99] that deciding the solvability of matching problems under strict subsumption conditions is NP-hard in the DL $\mathcal{ALN}$ and its sublanguages $\mathcal{FL}_\bot$ and $\mathcal{FL}_\neg$. Moreover, matching in $\mathcal{ALN}$ under subsumption conditions is polynomial, even if the relevant system of subsumption conditions contains cycles [Bra00, BBK01].

With respect to further extensions of the underlying DL language, matching proved to be a hard problem. Especially, in a DL that provides full negation, such as $\mathcal{ALC}$, matching becomes equally hard as unification, where variables are admitted on both sides of the equation. A unification problem $D_1 \equiv^? D_2$ with concept patterns D_1, D_2 is solved by a substitution σ if and only if σ solves the matching problem $\bot \equiv^? (\neg D_1 \sqcap D_2) \sqcup (D_1 \sqcap \neg D_2)$, where $\bot$ denotes the unsatisfiable concept. It is open how to solve unification problems in $\mathcal{ALC}$ or in the corresponding multi-modal logic $\mathbf{K}_m$. For more particulars on unification, see Section 1.7.2.

Least-common subsumer (and most-specific concept)

The least-common subsumer has first been introduced for the DLs $\mathcal{ALN}$ and $\mathcal{LS}$ in [CBH92]. The results for $\mathcal{LS}$ have been extended by introducing value restrictions in [CH94a, CH94b], thus reaching CORE CLASSIC, a subset of the DL language underlying the CLASSIC sys-

tem. Moreover, an lcs algorithm for full CLASSIC has been presented in [FP96]. It has been observed in [Küs01], however, that the algorithm in [FP96] is flawed because inconsistencies are not handled correctly. More importantly, the lcs algorithms cited above do not distinguish between total and partial functions when interpreting attributes. As shown in [Küs01], this difference must be strictly observed when computing the lcs in the languages mentioned above. In particular, the lcs of two $\mathcal{ALNS}$-concept descriptions with partial attributes always exists and can be computed in polynomial time; while in the case of total attributes the lcs does not always exist and, if it exists, can be of exponential size in the size of the input $\mathcal{ALNS}$-concept descriptions.

For the DLs $\mathcal{EL}$ and $\mathcal{ALE}$, a sound and complete lcs algorithm has been presented in [BKM99]. In [Küs01], an lcs algorithm for the DLs $\mathcal{ALNS}$ and $\mathcal{ALN}^*$ is proposed which deals correctly with partial and total attributes. An lcs operator for the full CLASSIC language has been presented in [KB01]. In [KM01b], an lcs algorithm for $\mathcal{ALEN}$ is presented.

The above mentioned contributions to the lcs have in common that only acyclic TBoxes are supported. In fact, most lcs algorithms are defined without underlying TBox. In order to compute the lcs of concepts defined in a TBox, the concepts must be expanded beforehand, i.e., every occurring defined concept is be replaced by its definition until only atomic concepts remain.

More recently, an lcs algorithm for possibly cyclic $\mathcal{EL}$-TBoxes has been proposed in [Baa03a]. In particular, it has been shown that the lcs need not exist when interpreting the TBox with the standard semantics, i.e., descriptive semantics. For greatest-fixedpoint semantics, however, it has been shown that the lcs always exists, that the binary lcs can be computed in polynomial time, and an optimal computation algorithm has been presented.

The computational complexity of the above lcs algorithms has been studied in depth. The binary lcs, i.e., the lcs applied to a set of only two concept descriptions, can be computed in polynomial time in the DL $\mathcal{EL}$, but can grow exponentially large in the size of the input concepts in $\mathcal{ALE}$ [BKM99]. Note that therefore any lcs algorithm in $\mathcal{ALE}$ is necessarily worst-case exponential. Without the restriction to two input concepts, the lcs can grow exponentially large in the size of the input concepts both in $\mathcal{EL}$ and $\mathcal{ALE}$ [BKM99]. In $\mathcal{ALN}^*$, the binary lcs can be computed in exponential time [Küs01] but it is open whether it can grow exponentially large in the size of the input concepts and whether a PSPACE-algorithm exists to compute it. The general lcs in $\mathcal{ALN}^*$ can grow exponentially large in the size of the input concepts and there exists an EXPTIME-algorithm to compute it. The binary lcs w.r.t. cyclic $\mathcal{EL}$-TBoxes interpreted with greatest-fixedpoint semantics can be computed in polynomial time [Baa03a], implying that it is of at most polynomial size in the size of the input concepts.

The non-standard inference most-specific concept has initially been suggested in the context of the early DL system KL-ONE as a means to reduce the instance problem to subsumption [SL83] and has subsequently also been employed withing the DL system CLASSIC. In contrast to the lcs, the msc does not always exist for many standard DLs. This has been shown in [BK98] for individuals defined w.r.t. acyclic $\mathcal{ALN}$-TBoxes and in [Mol00, KM01a] for acyclic $\mathcal{ALE}$-TBoxes. In order to overcome this problem, it has been shown in [BK98] that the msc w.r.t. cyclic $\mathcal{ALE}$-TBoxes does always exist. Hence, the msc of an individual defined w.r.t. an $\mathcal{ALN}$-TBox can always be expressed by introducing (possibly) cyclic definitions. In [KM01a], it has been shown for the case of $\mathcal{ALE}$ how an approximate msc can be computed whose maximum role depth is limited to some constant k. For the purpose of the present work, the only relevant result on the msc, see [Baa03a], shows how the msc of an individual defined w.r.t. an ABox and a cyclic $\mathcal{EL}$-TBox with fixedpoint semantics can be computed in polynomial time.

Approximation

Approximation has first been suggested as a DL inference service in [BKM00]. The first in-depth examination of approximation, however, has only been done in the context of the present work. See Section 1.6.2 for details.

1.6 New results

In our overview so far, we have highlighted two main directions of DL research; firstly, DL formalisms that provide expressive TBox formalisms, such as GCIs, but still allow TBoxes to be classified in polynomial time; and secondly, non-standard inference services for the support of knowledge maintenance tasks.

The present work provides new results in both of these directions. In the following section, we summarize all new results on standard inferences. Apart from subsumption as the main inference in terminological reasoning, the instance problem, i.e., assertional reasoning, will also be of interest. On the topic of non-standard inferences, we mainly present new results on matching and approximation. Our findings, however, will have some implications on the least-common subsumer and the most-specific concept.

In general, our work is not limited to examinations of the computational complexity of new instances of the above reasoning problems, but also provides practical decision and computation algorithms, several prototypical implementations, and systematic performance evaluations.

1.6.1 Standard inferences

Concerning the standard inference problem subsumption, the present work aims to find a DL language together with a TBox formalism with the following two properties.

- It should be as expressive as possible to be useful in practice, in particular for KR in the Life Sciences and to some extent the Semantic Web. As discussed in Sections 1.2 and 1.3, a TBox formalism supporting GCIs is highly desirable.
- Given the considerable size of knowledge bases in particular in the Life Sciences, reasoning should be tractable.

Note that an additional third property, the ability to support non-standard inference services, is discussed in more detail in Section 1.6.2.

In the light of the negative results for many well-known DLs shown in Section 1.5.1, clearly, tractable reasoning w.r.t. general TBoxes appears to be hard to achieve. Nevertheless, encouraged by the tractability result for cyclic $\mathcal{EL}$-TBoxes [Baa03b], the present work takes $\mathcal{EL}$ as a starting point of a systematic search for a DL with the above properties.

Intractability results

As every DL is characterized by the set of language constructors it provides, we begin by presenting intractability results obtained from extending $\mathcal{EL}$ by single additional constructors. Depending on the constructor in question, the subsumption problem becomes intractable already w.r.t. the empty TBox or acyclic TBoxes, implying intractability for general TBoxes. All new contributions to the complexity of subsumption problems are summarized in Table 1.6.1.

TBox	Extension of $\mathcal{EL}$ by	Deciding subsumption is	Proof in Section
empty	Disjunction	co-NP-complete	3.1.1, Th. 3.1.4
	Number restrictions	co-NP-complete	3.1.1, Th. 3.1.8
acyclic	Allsome, but no exist. restr.	co-NP-complete	3.1.2, Th. 3.1.20
	Allsome	co-NP-hard	3.1.2, Th. 3.1.20
general	Value restr., but no exist. restr.	EXPTIME-complete	3.2.4, Th. 3.2.13
	Atomic negation	EXPTIME-complete	3.1.3, Th. 3.1.23
	Disjunction	EXPTIME-complete	3.1.3, Th. 3.1.24
	Inverse roles	PSPACE-hard	3.1.3, Th. 3.1.25
	Functional roles	EXPTIME-complete	3.1.3, Th. 3.1.28
	At-most[11]restriction	EXPTIME-complete	3.1.3, Th. 3.1.29
	At-least[12]restriction	EXPTIME-complete	3.1.3, Th. 3.1.30
	Non-p-admissible concr. dom.	EXPTIME-complete	3.1.3, Th. 3.1.32
	Role negation	EXPTIME-complete	3.1.3, Th. 3.1.34
	Role union	EXPTIME-complete	3.1.3, Th. 3.1.34
	Refl.-trans. closure on roles	EXPTIME-complete	3.1.3, Th. 3.1.34

Table 1.6.1: New contributions to the complexity of subsumption

For $\mathcal{EL}$ extended by the the constructors disjunction ($\sqcup$) or number restrictions ($(\leqslant n\ r), (\geqslant n\ r)$), co-NP-completeness of subsumption is shown w.r.t. the empty TBox, see Section 3.1.1, Theorems 3.1.4 and 3.1.8. Adding the constructor allsome ($\forall\exists$) to $\mathcal{EL}$ makes subsumption co-NP-hard already w.r.t. acyclic TBoxes, see Section 3.1.2, Theorem 3.1.20. It is open whether this lower bound is tight, i.e., whether a non-deterministic polynomial time algorithm deciding non-subsumption exists. Without existential restrictions, however, i.e., in a DL providing only top concept, conjunction, and allsome, tight complexity bounds are obtained: we show that subsumption w.r.t. acyclic TBoxes is co-NP-complete. The relevant results are published in [Bra04b] and summarized in the first four rows of Table 1.6.1.

As mentioned in Section 1.5.1, extending $\mathcal{EL}$ by value restrictions is already known to make subsumption EXPTIME-complete w.r.t. general TBoxes [GMWK02]. In Section 3.2.4, Theorem 3.2.13, we improve upon this result by showing that EXPTIME-completeness of the subsumption problem holds even without existential restrictions, i.e., w.r.t. general $\mathcal{FL}_0$-TBoxes. Our result thereby also improves the known PSPACE-hardness of subsumption w.r.t. general $\mathcal{FL}_0$-TBoxes with descriptive semantics [KN03].

We prove that the same computational complexity, i.e., EXPTIME-completeness, is obtained when extending $\mathcal{EL}$ by any of the concept constructors atomic negation ($\neg P$), see Theorem 3.1.23, disjunction ($\sqcup$), see Theorem 3.1.24, at-most restrictions ($\leqslant n\ r$), see Theorem 3.1.29, and at-least restrictions ($\geqslant n\ r$), see Theorem 3.1.30. Moreover, EXPTIME-completeness of subsumption w.r.t. general TBoxes is also obtained when extending $\mathcal{EL}$ by functional roles, see Theorem 3.1.28, or by any of the role constructors negation ($\neg r$), union ($r \cup s$), or reflexive transitive closure (r^*), see Theorem 3.1.34. For the inverse role constructor (r^-), we could show that subsumption w.r.t. the corresponding extension of $\mathcal{EL}$ becomes PSPACE-hard. It is open whether this bound is tight, i.e., whether an appropriate polynomial space decision algorithm exists. Finally, reasoning

[11] Holds even if only ($\leqslant 1\ r$)-number restrictions are admitted.
[12] Holds even if only ($\geqslant 2\ r$)-number restrictions are admitted.

	TBox	Language	Proof in Section
$\mathcal{EL}^{++}$	general TBox + restr. role value maps	$\mathcal{EL}$ + bottom concept + nominals + p-admissible concr. dom.	3.2, Th. 3.2.7
$\mathcal{EL}$Hy	hybrid TBox (general + cyclic TBox)	$\mathcal{EL}$	3.3.2, Cor. 3.3.19

Table 1.6.2: New contributions to tractable subsumption algorithms

w.r.t. general TBoxes becomes EXPTIME-hard when extending $\mathcal{EL}$ by a concrete domain that is not p-admissible, see Theorem 3.1.32.

Note that Theorem 3.1.8 also shows NP-hardness of the *satisfiability* problem w.r.t. the empty TBox in $\mathcal{EL}$ extended by number restriction. Moreover, Theorems 3.1.30 and 3.1.29 show that EXPTIME-hardness for number restrictions holds even when restricting at-most restrictions ($\leqslant n\ r$) to the case $n = 1$ or at-least restrictions ($\geqslant n\ r$) to the case $n = 2$.

The relevant results are published in [BBL05]. Note that, trivially, all intractability results carry over to further extensions of $\mathcal{EL}$. For instance, subsumption w.r.t. general TBoxes in $\mathcal{EL}$ extended by number restrictions is also EXPTIME-complete.

Tractability results

Despite the abundance of intractability results shown above, our search for an appropriate DL in which subsumption w.r.t. general TBoxes is tractable has not only produced negative results. In a first positive step, we could answer the hitherto open question for $\mathcal{EL}$: it is shown in [Bra04b] that subsumption w.r.t. general TBoxes in $\mathcal{EL}$ is not only tractable, but remains so even when extending the underlying TBox formalism by role hierarchies. The result is further improved in [Bra04a], where the instance problem w.r.t. general $\mathcal{EL}$-TBoxes with role hierarchies is also shown to be tractable. In both cases, tractability is shown by devising a sound and complete decision algorithm that can be used in practice.

Using similar techniques for an extended subsumption procedure, we show in Section 3.2 that the subsumption problem remains tractable when, firstly, extending $\mathcal{EL}$ by the bottom concept ($\bot$), nominals, and p-admissible concrete domains, and secondly, extending general TBoxes by restricted[13] role value maps (RVMs). The resulting formalism is called $\mathcal{EL}^{++}$-CBoxes, where $\mathcal{EL}^{++}$ symbolizes the language extension of $\mathcal{EL}$, and CBox stands for *Constraint*-Box, emphasizing that GCIs *and* restricted RVMs are admitted in the TBox formalism. In Section 3.2.2, Theorem 3.2.7, we prove that not only subsumption w.r.t. $\mathcal{EL}^{++}$-CBoxes is tractable, but also satisfiability, ABox consistency, and the instance problem, if an $\mathcal{EL}^{++}$-ABox is additionally taken into account.

Hence, all relevant DL standard reasoning tasks, terminological as well as assertional, are tractable. Taking into account the above intractability results, it might be claimed that $\mathcal{EL}^{++}$-CBoxes are optimal in that every additional standard DL constructor, or every more powerful concrete domain, makes reasoning intractable. The features of $\mathcal{EL}^{++}$-CBoxes are summarized in the first row of Table 1.6.2. All relevant results are published in [BBL05]. We also show in Theorem 3.2.8 that $\mathcal{EL}^{++}$-CBoxes exhibit a small-model property, i.e., every satisfiable $\mathcal{EL}^{++}$-CBox has a model of linear size in the size of the CBox, every

[13] It has been shown in [Baa03b] that subsumption becomes undecidable even w.r.t. cyclic $\mathcal{EL}$-TBoxes when arbitrary RVMs are admitted.

non-subsumption w.r.t. an $\mathcal{EL}^{++}$-CBox has a countermodel of linear size, and analogously for assertional reasoning problems.

The claim that $\mathcal{EL}^{++}$-CBoxes are expressive enough for practical applications is substantiated further in Section 3.2. Without going into detail, let us sum up that $\mathcal{EL}^{++}$-CBoxes provide the means to express GCIs, role hierarchies, transitive roles, right identities, disjointness constraints, the unique name assumption, and domain restrictions on roles. Furthermore, examples of practically relevant p-admissible concrete domains are discussed in Section 3.2.3.

Having identified $\mathcal{EL}^{++}$-CBoxes as a solution of maximum expressivity in our search for tractable DL formalisms supporting GCIs, our further investigation is motivated by an additional goal: to support knowledge maintenance tasks by non-standard inference services in the manner described in Section 1.4. This leads to a novel TBox formalism, so-called *hybrid* $\mathcal{EL}$-TBoxes, where both GCIs and non-standard inferences, such as the lcs or matching, are supported. Tractability of subsumption w.r.t. hybrid $\mathcal{EL}$-TBoxes is shown in Section 3.3.2, Corollary 3.3.19. The proof is by a polynomial reduction to cyclic $\mathcal{EL}$-TBoxes for which a tractable subsumption algorithm exists [Baa03a], see also Table 1.6.2. The relevant results on subsumption w.r.t. hybrid $\mathcal{EL}$-TBoxes are published in [BM05]. As their main underlying motivation are non-standard inferences, our results on hybrid TBoxes are presented in more detail in Section 1.6.2.

Publications on standard inferences

The following publications document our contributions appertaining to standard inference services. As mentioned above, intractability results w.r.t. empty and acyclic TBoxes as well as the initial polynomial subsumption algorithm for general $\mathcal{EL}$-TBoxes extended by role hierarchies are contained in [Bra04b], while a polynomial algorithm for the instance problem is presented in [Bra04a]. The improvement of intractability results to general TBoxes and the extension of tractability results to $\mathcal{EL}^{++}$-CBoxes [BBL05] has been achieved in collaboration with coauthors. Our results on subsumption w.r.t. hybrid $\mathcal{EL}$-TBoxes [BM05] have been obtained in collaboration with a coauthor.

- [BBL05]
  F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, Edinburgh, UK, 2005. Morgan-Kaufmann Publishers.

- [BM05]
  S. Brandt and J. Model. Subsumption in $\mathcal{EL}$ w.r.t. hybrid TBoxes. In *Proceedings of the 28th Annual German Conference on Artificial Intelligence, (KI 2005)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2005.

- [Bra04a]
  S. Brandt. On subsumption and instance problem in $\mathcal{ELH}$ w.r.t. general TBoxes. In *Proceedings of the 2004 International Workshop on Description Logics (DL2004)*, CEUR-WS, 2004.

- [Bra04b]
  S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In R. López de Mantáras and L. Saitta, editors, *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-2004)*, pages 298–302. IOS Press, 2004.

Matching modulo equiv.	TBox	Language	Computation problem is	Proof in Section
under side conditions	empty	$\mathcal{FL}_\bot$	n.p., tight	4.3.3, Th. 4.3.16
		$\mathcal{FL}_\neg$	n.p., tight	4.3.3, Th. 4.3.16
		$\mathcal{ALN}$	n.p., tight	4.3.3, Th. 4.3.16
no side conditions	cyclic	$\mathcal{EL}$	exptime, tight	4.4.4, Cor. 4.4.32
	hybrid	$\mathcal{EL}$	exptime, tight	4.4.4, Cor. 4.4.32

Table 1.6.3: New contributions to the matching problem in DL

1.6.2 Non-standard inferences

The present work extends theoretical results on the non-standard inference matching and investigates in depth the novel non-standard inference approximation. Additionally, we present and evaluate the, to the best of our knowledge, first implementations of general purpose matching algorithms for DLs.

Matching

Our contributions to matching are threefold. Firstly, we present non-deterministic polynomial-time algorithms to solve matching problems under acyclic side conditions in $\mathcal{ALN}$ and two sublanguages, thereby also proving NP-completeness of the relevant decision problems. Secondly, we show how to solve matching problems w.r.t. cyclic $\mathcal{EL}$-TBoxes and in the presence of GCIs; and thirdly, we describe our implementation and empirical evaluation of matching algorithms for two common DLs, namely $\mathcal{ALE}$ and $\mathcal{ALN}$.

In Section 4.3.1, we devise non-deterministic polynomial-time algorithms to solve matching problems under side conditions in the DLs $\mathcal{ALN}$ and its sublanguages $\mathcal{FL}_\bot$ and $\mathcal{FL}_\neg$. As deciding the solvability of matching problems under side conditions is known to be NP-hard in the relevant languages [BKBM99], our algorithms are optimal and prove NP-completeness of the decision problems. Moreover, all algorithms compute the *least* matcher, i.e., the solution that contains as much information as possible. Note that, in contrast to subsumption, complexity upper bounds for matching do not automatically transfer to sublanguages and complexity lower bounds not automatically to superlanguages. The relevant results are published in [BBK01] and summarized in the first three rows of Table 1.6.3, where 'n.p., tight' denotes that a non-deterministic polynomial time computation algorithm has been devised and that this is optimal because the corresponding decision problem is already in NP.

In Section 4.4.4 we devise a deterministic exponential-time algorithm to solve matching problems w.r.t. cyclic $\mathcal{EL}$-TBoxes interpreted with greatest-fixedpoint semantics. The algorithm is sound and complete in the sense that all minimal[14] matchers are computed. Regarding the computational complexity, our algorithm is optimal because, as shown in Theorem 4.4.20, minimal matchers of matching problems w.r.t. cyclic $\mathcal{EL}$-TBoxes with greatest-fixedpoint semantics can be of exponential size in the input matching problem. Our matching algorithm generalizes matching in $\mathcal{EL}$ w.r.t. the empty TBox, as we show in Lemma 4.4.19. Together with the hardness result from [BK99], this implies that deciding the solvability of matching problems modulo equivalence w.r.t. cyclic $\mathcal{EL}$-TBoxes is NP-hard. It is open whether this bound is tight, i.e., whether a non-deterministic polynomial

[14] Even w.r.t. the empty TBox the least matcher to an $\mathcal{EL}$-matching problem does not always exist.

time decision algorithm exists. For matching problems modulo subsumption w.r.t. cyclic $\mathcal{EL}$-TBoxes, the decision problem is tractable.

The practical utility of GCIs and non-standard inferences motivates the question for a DL formalism in which both can be provided. The underlying problem here is one of appropriate choice of semantics. General TBoxes have to be interpreted w.r.t. descriptive semantics. As mentioned in Section 1.5.2, for this kind of semantics the lcs (and msc) need not always exist, even w.r.t. cyclic $\mathcal{EL}$-TBoxes [Baa03b]. This result carries over to general TBoxes and many extensions of the DL $\mathcal{EL}$. As the lcs can be reduced to matching, the same problem also occurs for matching problems.

In order to provide lcs and msc without sacrificing the convenience of GCIs, we propose *hybrid* TBoxes. A hybrid $\mathcal{EL}$-TBox is a pair $(\mathcal{F}, \mathcal{T})$ of a general TBox $\mathcal{F}$ ('foundation') and a possibly cyclic TBox $\mathcal{T}$ ('terminology') defined over the same set of atomic concepts and roles. $\mathcal{F}$ serves as a foundation of $\mathcal{T}$ in that the GCIs in $\mathcal{F}$ define relationships between concepts used as atomic concept names in the definitions in $\mathcal{T}$. Hence, $\mathcal{F}$ lays a foundation of general implications constraining $\mathcal{T}$. Hybrid $\mathcal{EL}$-TBoxes cannot be reduced to ordinary general $\mathcal{EL}$-TBoxes because of a difference in semantics: the foundation $\mathcal{F}$ is interpreted by descriptive semantics while the terminology $\mathcal{T}$ is interpreted by greatest-fixedpoint semantics, see Section 2.4 for details.

In Section 4.4, we show how minimal[14] matchers for matching problems w.r.t. hybrid $\mathcal{EL}$-TBoxes can be computed. Due to the reduction of hybrid $\mathcal{EL}$-TBoxes to cyclic $\mathcal{EL}$-TBoxes shown in Section 3.3.2, the above mentioned matching algorithm for the latter TBox formalism can be re-used. In this way, the goal of providing non-standard inferences in the presence of GCIs is met. As shown in Corollary 4.4.32, minimal matchers of matching problems w.r.t. hybrid $\mathcal{EL}$-TBoxes can grow exponentially large in the size of the input matching problem. The results are summarized in the last two rows of Table 1.6.3, where 'exptime, tight' denotes that a deterministic exponential time computation algorithm has been devised and that the solution of the relevant computation problem is of exponential size in the size of the input in the worst case, implying that any computation algorithm is worst-case exponential.

In addition to the above theoretical results, the present work also provides implementations of matching algorithms, namely matching in $\mathcal{ALE}$ [BK00a] and sublanguages, and matching in $\mathcal{ALN}$ [BKBM99] and sublanguages. To the best of our knowledge, these are the first implementations of independent, general matching algorithms for DLs. The implemented algorithms have been tested on randomly generated matching problems, see Sections 4.5.1 and 4.5.2, respectively, for results of the performance tests. Without going into detail here, one might state that both algorithms performed well even on relatively large matching problems.

A prototype implementation of our matching algorithms for cyclic and hybrid $\mathcal{EL}$-TBoxes have also been finished [Liu05] but have not yet been evaluated thoroughly.

Publications on matching

The following publications contain our new contributions to matching. The relevant results on matching under side conditions [BBK01] mainly stem from our own work. Apart from the entire implementation of matching in $\mathcal{ALE}$ [Bra03], our part in the implementation of matching in $\mathcal{ALN}$ [BL04] has been the algorithm's key architecture, data structures, and conception and implementation of a testbed for performance tests on randomized data.

Knowledge maintenance as one application of matching, see Section 1.4.1, is described in detail in [BT01]. In collaboration with a coauthor, the paper presents an application from the domain of chemical process engineering in which knowledge maintenance tasks are

facilitated by means of non-standard inference services. The results presented in [BT01] are not related to matching alone but also describe the approach of bottom-up extension, see Section 1.4.1, and thus can also be seen as contributing to the use of the lcs inference.

- [BBK01]
  F. Baader, S. Brandt, and R. Küsters. Matching under side conditions in description logics. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI-01*, pages 213–218, Seattle, Washington, 2001. Morgan Kaufmann.
- [BL04]
  S. Brandt and H. Liu. Implementing matching in $\mathcal{ALN}$. In *Proceedings of the KI-2004 Workshop on Applications of Description Logics (KI-ADL'04)*, CEUR-WS, Ulm, Germany, September 2004.
- [Bra03]
  S. Brandt. Implementing matching in $\mathcal{ALE}$—first results. In *Proceedings of the 2003 International Workshop on Description Logics (DL2003)*, CEUR-WS, 2003.
- [BT01]
  S. Brandt and A.-Y. Turhan. Using non-standard inferences in description logics — what does it buy me? In *Proceedings of the KI-2001 Workshop on Applications of Description Logics (KIDLWS'01)*, number 44 in CEUR-WS, Vienna, Austria, September 2001. RWTH Aachen.

Least-common subsumer (and most-specific concept)

Although the non-standard inferences lcs and msc are not in the focus of the present work, some results on matching have implications on them. Due to our polynomial reduction from hybrid $\mathcal{EL}$-TBoxes to cyclic $\mathcal{EL}$-TBoxes with greatest-fixedpoint semantics, see Section 3.3.2, all non-standard inferences for the latter TBox formalism are made available for hybrid TBoxes. Hence, building on the results from [Baa03a], we show in Section 4.4.6 how to compute the lcs and msc w.r.t. hybrid $\mathcal{EL}$-TBoxes. Moreover, we can show that the binary lcs w.r.t. hybrid $\mathcal{EL}$-TBoxes always exists and can be computed in polynomial time while the lcs of arbitrary arity is EXPTIME-complete, see Corollaries 4.4.24 and 4.4.26. Given an additional ABox, the most-specific concept w.r.t. this ABox and a hybrid TBox always exists and can be computed in deterministic polynomial time in the size of the input.

As shown in Section 4, every lcs computation can be expressed as a least solution to a matching problem modulo subsumption. Hence, our implementation of matching in $\mathcal{ALN}$ can be used to compute the least-common subsumer w.r.t. $\mathcal{ALN}$ and its sublanguages. To the best of our knowledge, this is the first implementation of an lcs algorithm for $\mathcal{ALN}$ and its sublanguages.

Apart from these derived findings, we have proposed a sound and complete algorithm to compute the lcs of $\mathcal{FLE}^+$-concept descriptions, i.e., in a DL providing top concept, conjunction, existential and value restriction, and transitive roles [BTK03]. As supporting general TBoxes seemed the more important goal from an application point of view, this result will, however, not be presented in detail.

Publications on the least-common subsumer

The following two papers document our results on the least-common subsumer in $\mathcal{FLE}$ extended by transitive roles, which, however, are beyond the scope of the present work.

Both results have been established in collaboration with coauthors.

- [BT03]
  S. Brandt and A.-Y. Turhan. Computing least common subsumers for $\mathcal{FLE}^+$. In *Proceedings of the 2003 International Workshop on Description Logics*, CEUR-WS, 2003.

- [BTK03]
  S. Brandt, A.-Y. Turhan, and R. Küsters. Extensions of non-standard inferences to description logics with transitive roles. In M. Vardi and A. Voronkov, editors, *Proceedings of the 10th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2003)*, Lecture Notes in Computer Science. Springer, 2003.

Approximation

As mentioned in Section 1.5.2, concept approximation is a novel non-standard inference examined in depth for the first time in the present work. Our main result is a sound and complete algorithm computing upper $\mathcal{ALE}$-approximations of $\mathcal{ALC}$-concept descriptions, see Section 5.1.2. While $\mathcal{ALC}$ provides the constructors top concept, conjunction, disjunction, negation, and existential and value restrictions, $\mathcal{ALE}$ is lacking general negation and disjunction but provides atomic negation. Hence, the key task is to translate concept descriptions with disjunctions into such without with minimal loss of information.

The computational complexity of approximation is discussed in Section 5.1.2. We show that any algorithm computing upper $\mathcal{ALE}$-approximations of $\mathcal{ALC}$-concept descriptions is worst-case exponential, see Corollary 5.1.13, and that our algorithm is a deterministic double exponential time algorithm, see Lemma 5.1.14. It is an open problem whether the upper bound is tight, i.e., whether any approximation algorithm in the above sense is worst-case double exponential. Our results on $\mathcal{ALC}$-$\mathcal{ALE}$-approximation are published in [BKT02b].

The high computational complexity of approximating arbitrary $\mathcal{ALC}$-concept descriptions naturally gives rise to the question whether certain classes of $\mathcal{ALC}$-concept descriptions can be found where approximations can be computed more easily. It is shown in Section 5.1.3 that so-called *nice* $\mathcal{ALC}$-concepts can be found for which a much simplified approximation algorithm produces correct results. The relevant result is published in [BT02b].

We also show in [BKT02b] how to provide clues about information lost during approximation. To this end, the accuracy of $\mathcal{ALC}$-$\mathcal{ALE}$-approximations is measured by means of a *difference* operator. Originally proposed to compute the difference of two $\mathcal{ALE}$-concept descriptions in [Küs01], we have generalized the difference operator to $\mathcal{ALC}$-$\mathcal{ALE}$-differences.

Moreover, the above mentioned algorithm for upper $\mathcal{ALE}$-approximations of $\mathcal{ALC}$-concept descriptions is extended by number restrictions in [BKT02a], thus computing upper $\mathcal{ALEN}$-approximations of $\mathcal{ALCN}$-concept descriptions. Both approximation algorithms, i.e., upper $\mathcal{ALE}$-approximation of $\mathcal{ALC}$-concepts and upper $\mathcal{ALEN}$-approximation of $\mathcal{ALCN}$-concepts, have been implemented prototypically, see [BKT02b] and [BKT02a].

The difference operator, $\mathcal{ALCN}$-$\mathcal{ALEN}$-approximation, and all results on implementations of the relevant algorithms, however, are beyond the scope of this work. The relevant contributions will be part of the forthcoming Ph.D. thesis by Turhan.

Publications on approximation

Our publications related to the non-standard inference approximation are listed below. The $\mathcal{ALC}$-$\mathcal{ALE}$-approximation algorithm is published in [BKT02b], together with results on an appropriate difference operator. The extended $\mathcal{ALCN}$-$\mathcal{ALEN}$-approximation algorithm is presented in [BKT02a]. Our optimization technique for $\mathcal{ALC}$-$\mathcal{ALE}$-approximation is presented in [BT02b]. All results have been obtained in collaboration with coauthors.

The results on difference, $\mathcal{ALCN}$-$\mathcal{ALEN}$-approximation, and implementations of all relevant algorithms will be part of the forthcoming Ph.D. thesis by Turhan.

- [BKT02a]
  S. Brandt, R. Küsters, and A.-Y. Turhan. Approximating $\mathcal{ALCN}$-concept descriptions. In *Proceedings of the 2002 International Workshop on Description Logics*, 2002.

- [BKT02b]
  S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. In D. Fensel, F. Giunchiglia, D. McGuiness, and M.-A. Williams, editors, *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*, pages 203–214, San Francisco, CA, 2002. Morgan Kaufman.

- [BT02]
  S. Brandt and A.-Y. Turhan. An approach for optimized approximation. In *Proceedings of the KI-2002 Workshop on Applications of Description Logics (KIDLWS'01)*, CEUR-WS, Aachen, Germany, September 2002. RWTH Aachen.

1.7 Related work

In order to complete our overview of the present work, we summarize some interesting results by other researchers in which similar techniques have been applied or which have been motivated by similar goals. A more detailed account of some of the related work mentioned below will be provided later on in the context of our technical results.

Analogous to the previous sections, we divide the presentation of related work into those appertaining to the field of standard inferences and to non-standard inferences.

1.7.1 Standard inferences

The primary motivation of our investigation of standard reasoning problems has been to devise a DL formalism that provides GCIs, i.e., general TBoxes, and w.r.t. which DL knowledge bases can be classified in polynomial time. In the following, we mention one implementation of our tractable DL formalism $\mathcal{EL}^{++}$-CBoxes, and also describe another approach to tractable reasoning w.r.t. DL knowledge bases.

The CEL reasoner

For the DL formalism $\mathcal{EL}^{++}$-CBoxes, an appropriate reasoner is currently being implemented. In its current state, the reasoner CEL [BLS05] supports $\mathcal{EL}^{+}$-CBoxes, where $\mathcal{EL}^{+}$ denotes the fragment of $\mathcal{EL}^{++}$ without the bottom concept, nominals, and concrete domains. Both GCIs and RIs are fully supported in the TBox formalism. The performance of CEL has been evaluated using three biomedical terminologies mentioned above, namely

SNOMED, GALEN, and GO. In the case of GALEN, an $\mathcal{EL}^+$-fragment of the original terminology has been used.

In contrast to current releases of the well-known, highly optimized reasoners FaCT and Racer, Cel was the only reasoner that could successfully classify all benchmark terminologies, and performed fastest on all but one benchmark. Currently, Cel is being further optimized and extended to support full $\mathcal{EL}^{++}$-CBoxes. For more details on Cel, see Section 3.4.

Tractable reasoning with DL-Lite

Apart from the DL formalism $\mathcal{EL}^{++}$-CBoxes proposed in the present work, other DL formalisms are motivated by the goal to provide tractable reasoning. Probably the most notable example is DL-Lite [CDGL$^+$05a]. The usual distinction between DL language and TBox formalism is not appropriate for DL-Lite-TBoxes because some constructors may occur only in certain places in the TBox. A DL-Lite-TBox is a set of inclusion axioms of the form $B \sqsubseteq C$, where B and C are defined by means of the grammar

$$B ::= P \mid \exists r \mid \exists r^-$$
$$C ::= B \mid \neg B \mid C \sqcap C,$$

where P stands for an atomic concept. Furthermore, DL-Lite-TBoxes may contain functionality assertions of the form (funct r) or (funct r^-) which enforce that the role r, or r^-, must be interpreted by a functional relation.

Obviously, DL-Lite does not support full $\mathcal{EL}$ and does not support arbitrary GCIs, and thus, no general TBoxes. On the other hand, DL-Lite is well suited to represent conceptual data models, such as entity relationship diagrams, and object-oriented formalisms, such as UML class diagrams. The true strengths of DL-Lite, however, lie in ABox reasoning and specifically in answering conjunctive queries over ABoxes. Like for our $\mathcal{EL}^{++}$ formalism, ABox reasoning in DL-Lite is tractable. But unlike the standard approach to ABox reasoning, DL-Lite ABoxes are translated into relational databases, so that reasoning over DL-Lite ABoxes can be performed by standard Database Management Systems, of which highly optimized implementations exist. As a consequence, ABoxes even with millions of instances can be handled in practice. In Section 3.4, some more details of DL-Lite are discussed. Moreover, in Section 1.7.2 we briefly show how DL-Lite queries are defined.

Other related work

One of our intractability results mentioned above, EXPTIME-hardness of the subsumption problem in $\mathcal{FL}_0$ w.r.t. general TBoxes, has been proven independently by reduction of the existence of winning strategies in pushdown games [Hof05]. Moreover, after publication of [BBL05], tractability of subsumption w.r.t. $\mathcal{EL}^{++}$-CBoxes has been re-proven in [Kaz05] by means of resolution calculi.

1.7.2 Non-standard inferences

On the subject of non-standard inferences, we begin by presenting theoretical results on unification that are related to matching. Regarding the application of matching as a query mechanism for DL knowledge bases, we describe other well-known DL-based query languages and point out important differences to our approach in the underlying motivation. Although the present work has only minor implications on the lcs inference, some results related to this non-standard inference are discussed. Furthermore, we describe a

DL system that makes several non-standard inference services available to modern knowledge editors. Finally, another important inference tasks appertaining to the domain of non-standard inferences is mentioned, namely explanation. Although explanation has no immediate relation to the other non-standard inferences presented here, we would like to mention it in order to complete the picture of non-standard inferences as tools supporting knowledge engineering.

Unification

As an extension of the non-standard inference service matching, we should mention *unification*. As described above, a matching problem (modulo equivalence) consists of a concept description C and a concept pattern D containing variables. The task is to assign concept descriptions to the variables in D in such a way that equivalence to C holds. Unification problems extend matching problems in that variables may occur on both sides of the equation, i.e., both C and D may be concept patterns. A substitution solves a unification problem if and only if equivalence of C and D holds after application of the substitution to both patterns. It has been observed in [BN01] that unification can be used to detect redundancies in DL knowledge bases, and is even better suited for this purpose than the inference service matching.

Unification has been studied in depth in [BN01], where a sound and complete deterministic exponential time unification algorithm for the DL $\mathcal{FL}_0$ has been presented. It has also been shown that deciding the solvability of $\mathcal{FL}_0$-unification problems is EXPTIME-hard. For the DL $\mathcal{FL}_{\text{reg}}$, the extension of $\mathcal{FL}_0$ by the role constructors union, composition, and transitive closure, a sound and complete deterministic exponential time unification algorithm has been devised [BK01]. It has also been shown that deciding the solvability of $\mathcal{FL}_{\text{reg}}$-unification problems is EXPTIME-complete and that $\mathcal{FL}_{\text{reg}}$-unification problems always have a least unifier. It has been shown in [BK02] that all properties observed for $\mathcal{FL}_{\text{reg}}$ carry over to an extension of $\mathcal{FL}_{\text{reg}}$ by the bottom concept. As matching is a special case of unification, the relevant results show how to solve matching problems in $\mathcal{FL}_{\text{reg}}$ extended by the bottom concept in deterministic exponential time, show that a least matcher to such matching problems always exists (and how it can be computed), and show that the decision problem is in EXPTIME.

Unfortunately, unification problems proved to be particularly hard to solve for DLs extending the above languages. As a consequence, there are no unification algorithms available for the DL formalisms of main interest in the present work.

Querying DL knowledge bases

One motivation mentioned above for the non-standard inference matching is querying DL terminologies for concepts of a certain structure specified by a concept pattern. To the best of our knowledge, there exist no other results on the topic of TBox querying. Nevertheless, the question of how to query the ABox of a DL knowledge base has been studied in depth. In order to clarify the difference between both kinds of querying, we discuss some of the relevant results.

For the DL formalism DL-Lite mentioned above, a query language has been proposed in [CDGL$^+$05a, ACDG$^+$05]. The relevant approach supports so-called *conjunctive queries* of the form $q(\vec{x}) \leftarrow \exists \vec{y}.conj(\vec{x}, \vec{y})$, where $\vec{x}$ denotes a vector of *distinguished* variables, $\vec{y}$ a vector of *non-distinguished* variables, and $conj(\vec{x}, \vec{y})$ a conjunction of atoms of the form $B(z)$ or $r(z_1, z_2)$ with the following properties. B denotes a basic concept (see above), r a role name, and z, z_1, z_2 stand either for individual names defined in the underlying

ABox or for variables occurring in the vectors $\vec{x}$ or $\vec{y}$. To answer a query of the above form means to substitute the variables in $\vec{x}$ and $\vec{y}$ by names of ABox individuals such that the expression $conj(\vec{x}, \vec{y})$ is true w.r.t. every model of the underlying DL-Lite knowledge base. Query answering of this form w.r.t. DL-Lite knowledge bases is supported by the system QuOnto [ACDG+05]. The system can also decide query containment, i.e., the question whether one query always returns a subset of the results of another query, and satisfiability of a given DL-Lite knowledge base.

It should be clarified that the above query formalism aims at ABox querying w.r.t. an underlying TBox and not at TBox querying. Hence, the purpose is to retrieve ABox individuals with certain properties and not TBox concepts, as in our case. The topic of query formalisms for ABoxes is also considered important in the context of reasoning over the Semantic Web. For instance, significant effort has been invested into defining standard languages for querying OWL knowledge bases, see, e.g., [FHH04, HT02]. Another successful approach to defining query languages for the Semantic Web has been presented in [WM05, GKM05]. Again, we mention these results in order to distinguish our notion of querying, i.e., querying TBoxes for concept descriptions, from the more common notion of querying used DL research, i.e., querying for ABox individuals.

Integrating DL knowledge bases

Apart from matching under side conditions, other successful techniques have been proposed as a means to integrate DL terminologies. In [CDGL02, CDGL+01, CDGL+98], approaches are presented that are influenced especially by research in the area of relational databases, where the problem of integrating database schemata by means of *interschema assertions* is well-investigated. Moreover, distributed DLs proposed in [ST04, BS03] allow to integrate several knowledge bases using so-called *bridge rules* between concepts and *correspondences* between (groups of) individuals. It has been observed in [KLWZ04] that distributed DLs can be seen as a special case of *$\mathcal{E}$-connections* between *abstract description systems*, a generalization of DL, modal logics, epistemic logics, and some logics of time and space. In [GP04] an approach to integrating OWL-Lite terminologies based on $\mathcal{E}$-connections has been presented—together with an implementation based on the DL reasoner Pellet.

Extensions of the lcs

Concerning the non-standard inference least-common subsumer, there are approaches to make use of TBox information differently from just expanding all definitions. For instance, in [BST04], an algorithm is proposed to compute least-common subsumers of $\mathcal{ALE}$ concept descriptions containing concept names defined over a much more expressive $\mathcal{ALC}$-TBox. The relevant algorithm uses methods from formal concept analysis [Gan99, GW99] to extract information from the underlying TBox and approximates the lcs by a so-called *good common subsumer (gcs)*.

More precisely, formal concept analysis is used to compute the smallest conjunction of concepts (and their negations) defined in the $\mathcal{ALC}$-TBox that subsume concept names occurring in the concept descriptions of which a gcs is required. This conjunction is then used in the gcs construction. The gcs is preferred over the lcs when computing commonalities w.r.t. a background terminology mainly due to efficiency reasons. It has also been shown in [BST04] that the lcs of a finite set of $\mathcal{ALE}$-concept descriptions defined over an acyclic $\mathcal{ALC}$-TBox always exists and can be effectively computed. Hence, it might be possible in the future to devise a practical algorithm computing the lcs in this scenario.

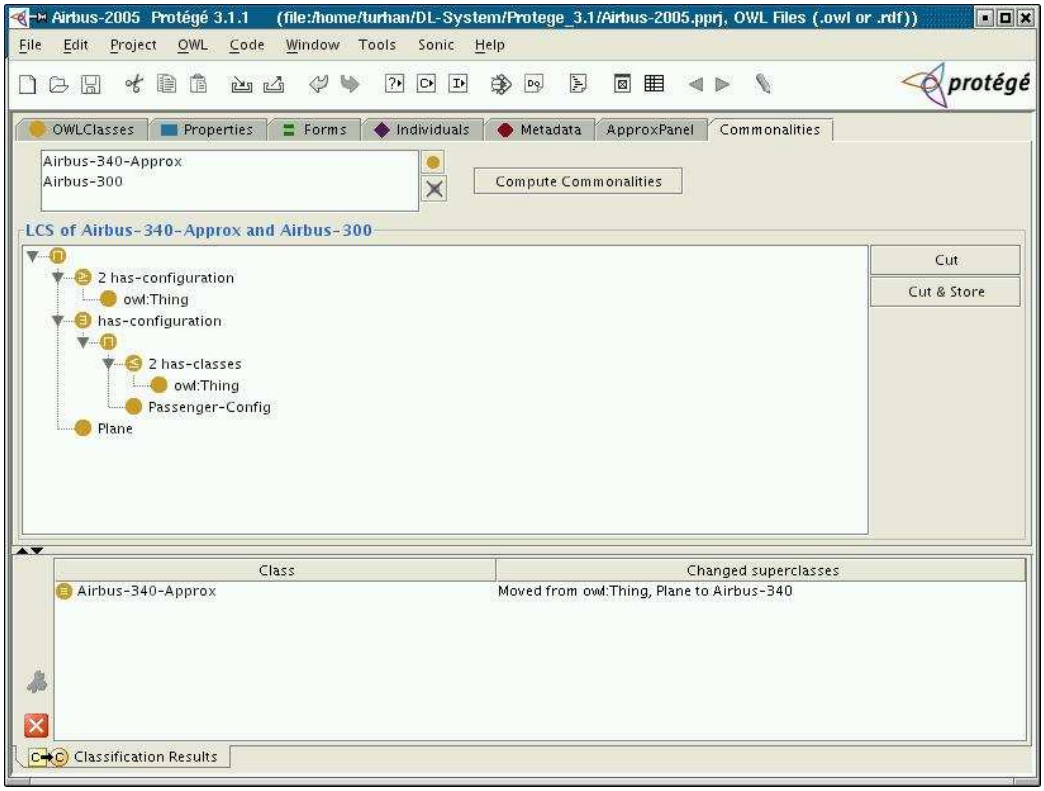


Figure 1.7.1: The non-standard inference component SONIC

The SONIC system

From a practical perspective, the arguably most notable contribution to the area of non-standard inferences is the system SONIC[15] [Tur05, TK04]. In contrast to stand-alone DL systems consisting of reasoner and knowledge editor, SONIC comprises (i) a reasoning component that uses a DIG-compliant DL reasoner, e.g., FACT or RACER, and (ii) a user-interface component that can be plugged into the knowledge editors OILED [BHGS01] and PROTÉGÉ [GMF+03]. The reasoning component implements the non-standard inferences least-common subsumer, good common subsumer, approximation, and rewriting. More precisely, SONIC provides the lcs for $\mathcal{ALEN}$-concept descriptions defined over acyclic $\mathcal{ALEN}$-TBoxes, the good common subsumer of $\mathcal{ALE}$-concept descriptions defined over acyclic $\mathcal{ALC}$-TBoxes, $\mathcal{ALE}$-approximations of $\mathcal{ALC}$-concept descriptions defined over acyclic $\mathcal{ALC}$-TBoxes, $\mathcal{ALEN}$-approximations of $\mathcal{ALCN}$-concept descriptions defined over acyclic $\mathcal{ALCN}$-TBoxes, and rewriting of $\mathcal{ALE}$-concept descriptions w.r.t. acyclic $\mathcal{ALC}$-TBoxes. In particular, for $\mathcal{ALC}$-$\mathcal{ALE}$-approximation, an implementation of our approximation algorithm presented in Section 5.1.2 is used.

The user-interface component of SONIC for the knowledge editor PROTÉGÉ is shown in Figure 1.7.1. Two new sub-windows are provided, one for the computation of approximations ('ApproxPanel'), and another for lcs or gcs ('Commonalities', shown above), depending on the type of the underlying TBox. The lcs is preferred for concepts defined over acyclic $\mathcal{ALEN}$-TBoxes, while for $\mathcal{ALCN}$-TBoxes the gcs inference is used. Figure 1.7.1 shows an example computation of the lcs of the concepts *Airbus-340-Approx* and *Airbus-300*. Both

[15] Simple Ontology Non-standard Inference Component

concepts have been taken from an example TBox hidden under the sub-window 'OWL Classes' and added to a list in the upper left of the sub-window shown above. The button 'Compute commonalities' actuates the lcs computation, the result of which is presented to the user in a box in the middle of the sub-window, labeled 'LCS of *Airbus-340-Approx* and *Airbus-300*' in our example. It should be stressed that the SONIC component is fully integrated into the knowledge editor in the sense that input to and output from non-standard inference computations can be imported from and exported to the TBoxes managed by the knowledge editor. In the example shown in Figure 1.7.1, it suffices to press the button 'Cut&Store' to insert the newly computed concept into the current TBox.

The components of SONIC communicate with the knowledge editor and with the underlying DL reasoner via the standardized DIG interface. Hence, it is relatively easy to exchange both reasoner and knowledge editor for which to provide additional functionality. In the future, SONIC might be extended to other non-standard inferences, such as the most-specific concept, thereby supporting the full bottom-up approach to extending DL knowledge bases described in Section 1.4.1, and matching.

Explanation

As pointed out in Section 1.5.2, the main motivation underlying non-standard inference services is to support users of DL systems in the task of knowledge engineering, i.e., extending and maintaining DL knowledge bases in a controlled and well-structured way. The non-standard inferences introduced so far mainly serve to retrieve concepts from a TBox and to construct new concepts by means of tools that help to avoid introducing redundancies and inconsistencies into a TBox.

Given the enormous size of real-world knowledge bases, and the number of persons maintaining them simultaneously, it seems unlikely that redundancies and inconsistencies can be avoided altogether. As mentioned above, the consistency of a DL knowledge base can easily be checked by modern DL systems. But even if the DL system detects that a new concept constructed by a domain expert is inconsistent with the TBox, and thus cannot be added to it, the domain expert might not understand the reason of this inconsistency, for instance because the overall size of the concept makes it difficult to perceive all its implications. Situations of this kind can be resolved by an inference service that is able to *explain* which parts of an inconsistent concept are actually responsible for the inconsistency.

An basic explanation facility for subsumption and non-subsumption based on deduction rules was already part of the DL system CLASSIC, see [MB95, McG96]. Although this approach relied on manual annotations to some extent, it produced relatively concise explanations even on large TBoxes due to a pruning mechanism based on matching. An approach to explaining subsumption between $\mathcal{ALC}$-concept descriptions has been presented in [BFH00, BFH$^+$99], where a modified sequent calculus is used to explain subsumptions computed by a tableaux algorithm.

The two approaches mentioned above have in common that subsumption or inconsistency is explained 'semantically': they are independent of the actual syntactical form of the underlying DL knowledge base because the explanation refers to the *reasoner*, or to a corresponding calculus. More recently, a weaker 'syntactical' approach to explanation has been discussed, explaining inconsistencies by merely highlighting inconsistent parts of a given DL knowledge base. This approach has already been studied in depth in [BH95], where an algorithm is presented to compute minimal inconsistent (and maximal consistent) subsets of a given $\mathcal{ALCF}$-ABox.

The same strategy has been utilized in [SC03] to devise an explanation facility for acyclic $\mathcal{ALC}$-TBoxes containing inconsistent concept definitions. The main idea is to identify

inconsistencies in a TBox $\mathcal{T}$ by (i) removing all concept definitions from $\mathcal{T}$ which are not responsible for the inconsistency, and (ii) pruning the remaining definitions as long as the resulting TBox is still inconsistent. More precisely, the approach does not remove definitions and subconcepts arbitrarily, but rather employs some optimization techniques to find those parts of the TBox 'most responsible' for the inconsistency.

Recently, an extended version of the explanation facility from [SC03] has been incorporated into the knowledge editor SWOOP [PSK05]. As a result, the editor can support users maintaining OWL knowledge bases.

It should be noted, however, that explanation differs from the other non-standard inference services mentioned above in that the usual perspective of a sound and complete algorithm solving computation problems does not really apply. The reason is that a meaningful definition of optimality is missing, i.e., the question of which subdescriptions in which concept definition are truly responsible for an observed inconsistency defies a well-defined answer. In this sense, explanation might rather be regarded as a tool enhancing the functionality of knowledge editors—albeit, a useful and important one.

Other related work

For the new non-standard inference approximation, we could show that $\mathcal{ALE}$-approximations of $\mathcal{ALC}$-concept descriptions can be computed in double exponential time, leaving open the question whether or not there exists an exponential time algorithm. In [LDLT02] an attempt has been made to answer this question in the affirmative by proposing an algorithm to compute $\mathcal{ALC}$-$\mathcal{ALE}$-approximations in deterministic exponential time. This result, however, has been refuted by the same authors by claiming in [LDLT03] that, even w.r.t. a certain compact representation of concept descriptions, no deterministic exponential time algorithm for $\mathcal{ALC}$-$\mathcal{ALE}$-approximations exists. To the best of our knowledge, the relevant question still has to be considered open.

1.8 The structure of the thesis

The remainder of the present work is organized as follows. Chapter 2 lays the formal foundation necessary for the subsequent presentation of our technical results. In Section 2.1, syntax and semantics of DLs are defined formally. In Section 2.2, we compare the different semantics available for cyclic $\mathcal{EL}$-TBoxes, i.e., descriptive semantics and greatest-fixedpoint semantics. Section 2.3 introduces concrete domains, while Section 2.4 provides a formal definition of hybrid TBoxes, a novel TBox formalisms proposed in the present work.

Chapters 3 to 5 contain the core of our technical results. In Chapter 3, we examine the standard reasoning problem subsumption for the DL $\mathcal{EL}$ and extensions thereof. All intractability results, i.e., extensions of $\mathcal{EL}$ for which subsumption is intractable, are presented in Section 3.1. As our main tractability result, we show in Section 3.2 that subsumption w.r.t. $\mathcal{EL}^{++}$-CBoxes is tractable, and incidentally all other terminological and assertional standard inferences. Our second tractability results is presented in Section 3.3, where tractability of subsumption is shown for the novel DL formalism hybrid TBoxes.

Our results on the topic of non-standard inferences are presented in Chapters 4 and 5. Chapter 4 deals with the non-standard inference matching, introducing in Sections 4.1 and 4.2 two known matching algorithms for the DLs $\mathcal{ALE}$ and $\mathcal{ALN}$, respectively. We show in Section 4.3 how matching problems under side conditions can be solved in the DLs $\mathcal{ALN}$ and its sublanguages $\mathcal{FL}_\bot$ and $\mathcal{FL}_\neg$. Section 4.4 introduces an appropriate notion of matching problems for cyclic and hybrid $\mathcal{EL}$-TBoxes and presents matching algorithms for

both DL formalisms. The main implementation results of the present work are described in Section 4.5, where implementations of the matching algorithms from Sections 4.1 and 4.2 are presented.

In contrast to the well-investigated non-standard inferences examined in Chapter 4, we introduce a novel non-standard inference in Chapter 5, namely approximation. In particular, we define $\mathcal{ALE}$-approximations of $\mathcal{ALC}$-concept descriptions in Section 5.1 and present an appropriate approximation algorithm. We have extended our results on approximation to $\mathcal{ALEN}$-approximations of $\mathcal{ALCN}$-concept descriptions and have also devised a difference operator by which an $\mathcal{ALC}$-concept description can be compared to its approximation. These extensions are briefly described in Section 5.2.

Finally, the results achieved in the present thesis will be summarized and discussed in Chapter 6.

CHAPTER 2

FORMAL PRELIMINARIES

The present chapter formally introduces syntax and semantics of the DL formalisms of interest in the present work. In particular, two different kinds of semantics will be relevant, namely descriptive semantics, introduced in Section 2.1.1, and greatest-fixedpoint semantics, introduced in Section 2.1.2. The difference between both semantics is discussed in Section 2.2. A more advanced topic from DL research is introduced in Section 2.3, namely concrete domains. Finally, hybrid TBoxes are introduced in Section 2.4 as a novel TBox formalism.

Throughout this work, the word 'iff' is used as an equivalent of 'if and only if'. It should also be noted that we include 0 in the set of natural numbers, i.e., $\mathbb{N}$ is defined as $\mathbb{N} := \{0, 1, \dots\}$. For every set S, we denote the cardinality of S by $|S|$ or $\#S$, and the power set of S by $\wp(S)$, i.e., $\wp(S) := \{T \mid T \subseteq S\}$. Note also that notions newly introduced in a definition are set im *italic* type and that the end of the body of every definition, every example, and every proof is indicated by a box ($\square$).

2.1 Description logic basics

Concept descriptions are inductively defined by means of a set of concept *constructors*, starting with a set N_{con} of *concept names* and a set N_{role} of *role names*. In this work, we mainly consider the constructors bottom concept ($\bot$), top concept ($\top$), conjunction ($\sqcap$), disjunction ($\sqcup$), negation ($\neg$), existential restriction ($\exists$), value restriction ($\forall$), allsome ($\forall\exists$), and number restrictions ($\leqslant n$, $\geqslant n$). Using an additional set N_{nom} of *individual names*, we will also consider the *nominal* constructor ($\{\cdot\}$).

Definition 2.1.1 (Concept descriptions)
Let $\mathsf{N}_{\text{con}}, \mathsf{N}_{\text{role}}, \mathsf{N}_{\text{nom}}$ be pairwise disjoint finite sets. Then the set of *concept descriptions over* N_{con}, N_{role}, *and* N_{nom} is inductively defined as follows.

1. $\top$, $\bot$, and every $P \in \mathsf{N}_{\text{con}}$ is a concept description;
2. if C, D are concept descriptions then so are $(C \sqcap D)$, $(C \sqcup D)$, $\neg C$, $\exists r.C$, $\forall r.C$, and $\forall\exists r.C$;
3. for every $r \in \mathsf{N}_{\text{role}}$ and $n \in \mathbb{N}$, $(\leqslant n\ r)$ and $(\geqslant n\ r)$ are concept descriptions; and
4. for every $a \in \mathsf{N}_{\text{nom}}$, $\{a\}$ is a concept description. $\square$

	$\top, \sqcap$	$\bot$	$\sqcup$	$\neg P$	$\neg$	$\exists$	$\forall$	$\forall\exists$	$\leq, \geq$	$\{a\}$
$\mathcal{FL}_0$	•						•			
$\mathcal{FL}_\bot$	•	•					•			
$\mathcal{FL}_\neg$	•	•		•			•			
$\mathcal{ALN}$	•	•		•			•		•	
$\mathcal{ALE}$	•	•		•		•	•			
$\mathcal{ALC}$	•	•	•	•	•	•	•			
$\mathcal{EL}^{++}$	•	•				•				•
$\mathcal{ELN}$	•					•			•	
$\mathcal{ELU}$	•		•			•				
$\mathcal{EL}_{\forall\exists}$	•					•		•		
$\mathcal{EL}_\forall$	•					•	•			
$\mathcal{EL}$	•					•				

Table 2.1.1: Description logics under consideration

In order to refer to the computational complexity of reasoning tasks later on, we need a measure for the size of concept descriptions. Intuitively, we define the size of a concept description as the space necessary to write it down.

Definition 2.1.2 The size of a concept description over N_{con}, N_{role}, and N_{nom} is inductively defined as follows.

- $|\top| := |\bot| := |A| := 1$ for every $A \in \mathsf{N}_{\text{con}}$, $|\{a\}| := 1$ for every $a \in \mathsf{N}_{\text{nom}}$;
- $|C \sqcap D| := |C \sqcup D| := 1 + |C| + |D|$, $|\neg C| := 1 + |C|$, $|\exists r.C| := |\forall r.C| := |\forall\exists r.C| := 1 + |C|$ for all concept descriptions C, D and every role $r \in \mathsf{N}_{\text{role}}$; and
- $|(\leqslant n\ r)| = |(\geqslant n\ r)| = \lceil \lg(max\{2, n\}) \rceil$ for every $n \in \mathbb{N}$ and every role $r \in \mathsf{N}_{\text{role}}$.

Note that, usually, binary encoding is assumed for number restrictions. Every actual *description logic* (DL) is characterized by the set of constructors it offers. In this work, several different DLs are discussed, especially $\mathcal{FL}_0$, $\mathcal{FL}_\bot$, $\mathcal{FL}_\neg$, $\mathcal{ALN}$, $\mathcal{ALE}$, $\mathcal{ALC}$, $\mathcal{EL}$, $\mathcal{EL}_\forall$[1], $\mathcal{EL}_{\forall\exists}$, $\mathcal{ELU}$, $\mathcal{ELN}$, and $\mathcal{EL}^{++}$. Table 2.1.1 shows which constructors are available in which DL. For instance, $\mathcal{ELN}$ provides top concept, conjunction, existential restriction, and number restrictions. For every DL $\mathcal{L}$, those concept descriptions restricted to the set of constructors provided by $\mathcal{L}$ are called *$\mathcal{L}$-concept descriptions*.

Concept descriptions are interpreted w.r.t. a model-theoretic *semantics*, where a given interpretation assigns to every concept description a subset of the relevant interpretation domain.

Definition 2.1.3 (Semantics)
Let $\Delta^\mathcal{I}$ be a nonempty set and let $\cdot^\mathcal{I}$ be an *interpretation function* with $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ for every $A \in \mathsf{N}_{\text{con}}$, $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ for every $r \in \mathsf{N}_{\text{role}}$, and $a^\mathcal{I} \in \Delta^\mathcal{I}$ for every $a \in \mathsf{N}_{\text{nom}}$. Then

[1] $\mathcal{EL}_\forall$ is also called $\mathcal{FLE}$, e.g. in [Küs01]. In our context, however, this DL is only of minor interest as an extension of $\mathcal{EL}$, and is therefore called $\mathcal{EL}_\forall$.

the *interpretation* $\mathcal{I} := (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is inductively extended to concept descriptions as follows.

top concept	$\top^{\mathcal{I}} := \Delta^{\mathcal{I}}$
bottom concept	$\bot^{\mathcal{I}} := \emptyset$
conjunction	$(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$(C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$
negation	$(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
existential restriction	$(\exists r.C)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid \exists y\colon (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
value restriction	$(\forall r.C)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid \exists y\colon (x,y) \in r^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$
allsome	$(\forall\exists r.C)^{\mathcal{I}} := (\forall r.C \sqcap \exists r.\top)^{\mathcal{I}}$
$\leq$-number restriction	$(\leqslant n\ r)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid \#\{y \mid (x,y) \in r^{\mathcal{I}}\} \leq n\}$
$\geq$-number restriction	$(\geqslant n\ r)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid \#\{y \mid (x,y) \in r^{\mathcal{I}}\} \geq n\}$
nominal	$\{a\}^{\mathcal{I}} := \{a^{\mathcal{I}}\}$.

As intersection and union are associative, parentheses can be omitted in aggregate conjunctions and disjunctions, e.g., $P_1 \sqcap P_2 \sqcap P_3$ instead of $(P_1 \sqcap (P_2 \sqcap P_3))$. If more than one type of constructor occurs, the precedence order $\{\neg\} > \{\exists, \forall, \forall\exists\} > \{\sqcap, \sqcup\}$ applies. E.g., $\exists r.\neg P_1 \sqcap P_2$ stands for $((\exists r.(\neg P_1)) \sqcap P_2)$. Moreover, for every finite set S of concept descriptions, $\bigsqcap_{C \in S} C$ denotes the conjunction over all elements of S and $\bigsqcup_{C \in S} C$ the respective disjunction. Define $\bigsqcap_{C \in \emptyset} C := \top$ and $\bigsqcup_{C \in \emptyset} C := \bot$.

As mentioned in Section 1.1, the main purpose of DLs is to be used as underlying representation language for knowledge bases. The following definition introduces the components of DL-based knowledge bases formally.

Definition 2.1.4 (TBox, CBox, ABox)
Let $\mathsf{N}_{\text{def}}, \mathsf{N}_{\text{prim}}$ be a partition of N_{con}. For every $A \in \mathsf{N}_{\text{def}}$ and every concept description C over N_{con}, N_{role}, and N_{nom}, $A \equiv C$ is a *definition of* A. Every finite set of definitions is a *terminology (TBox)* over N_{def}, N_{prim}, N_{role}, and N_{nom} iff it contains at most one definition of A for every $A \in \mathsf{N}_{\text{def}}$.

A TBox $\mathcal{T}$ is *acyclic* iff $\mathcal{T}$ is of the form $\{A_i \equiv C_i \mid 1 \leq i \leq n\}$ such that for every $i \in \{1, \ldots, n\}$, only defined names from $\{A_1, \ldots, A_{i-1}\}$ occur in C_i.

For $\mathcal{L}$-concept descriptions C, D over N_{con}, N_{role}, and N_{nom}, $C \sqsubseteq D$ is a *general concept inclusion (GCI) axiom*. For every $n \in \mathbb{N}_+$ and $r_1, \ldots, r_n, r \in \mathsf{N}_{\text{role}}$, $r_1 \circ \cdots \circ r_n \sqsubseteq r$ is a *role inclusion (RI) axiom*. In case of $n = 1$, an RI is called *simple RI (SRI)*. Every finite set of GCIs is a *general TBox* and every finite set of GCIs and RIs is a *constraint Box (CBox)* over N_{con}, N_{role}, and N_{nom}.

For every $a, b \in \mathsf{N}_{\text{nom}}$, $C \in \mathsf{N}_{\text{con}}$, and $r \in \mathsf{N}_{\text{role}}$, $C(a)$ is a *concept assertion* and $r(a,b)$ is a *role assertion*. Every finite set of concept assertions and role assertions is an *assertional Box (ABox)*. □

Throughout the present work, we consider arbitrary but fixed sets $\mathsf{N}_{\text{con}} = \mathsf{N}_{\text{def}} \uplus \mathsf{N}_{\text{prim}}$, N_{role}, and N_{nom}. Moreover, we consider only $\leqslant$- and $\geqslant$-number restrictions from arbitrary but fixed finite sets $\mathsf{N}_{\leqslant}$ and $\mathsf{N}_{\geqslant}$. Unless otherwise specified, all concept descriptions, interpretations, TBoxes, CBoxes, and ABoxes are defined over these sets. For every TBox $\mathcal{T}$, denote by $\mathsf{N}_{\text{con}}^{\mathcal{T}}$ ($\mathsf{N}_{\text{def}}^{\mathcal{T}}$, $\mathsf{N}_{\text{prim}}^{\mathcal{T}}$), $\mathsf{N}_{\text{role}}^{\mathcal{T}}$, and $\mathsf{N}_{\text{nom}}^{\mathcal{T}}$ the sets of concept names (defined, primitive concept names), role names and individual names, respectively, occurring in $\mathcal{T}$. The same notation is introduced for CBoxes and ABoxes.

Note that, throughout this work, concept names are usually denoted by uppercase letters, e.g., $A, B, \ldots$ for defined concept names and $P, Q, \ldots$ for atomic concept names, and lowercase letters for roles, e.g., $r, s, \ldots$. In examples with longer concept and role names, defined concept names are denoted in italics, e.g., *Parent*, while atomic concepts and roles are denoted in upright shape, e.g., Person.

For a given DL $\mathcal{L}$, we speak of an $\mathcal{L}$-TBox $\mathcal{T}$, an $\mathcal{L}$-CBox $\mathcal{C}$, or an $\mathcal{L}$-ABox $\mathcal{A}$ iff only $\mathcal{L}$-concept descriptions occur in the definitions of $\mathcal{T}$, in the GCIs of $\mathcal{C}$, and in the assertions of $\mathcal{A}$. Note that usually complex concept descriptions do not occur in ABoxes. Nevertheless, these may use concept names defined in a TBox or CBox.

In order to refer to the contents of TBoxes more easily, we introduce some auxiliary notation used throughout this work.

Definition 2.1.5 (deft, def)
Let $\mathcal{T}$ be a TBox and let $A \equiv C \in \mathcal{T}$ for some $A \in \mathsf{N}_{\text{def}}$ and some concept description C. Then define $\mathsf{deft}_{\mathcal{T}}(A) := C$. Moreover, denote by $\mathsf{def}_{\mathcal{T}}(A)$ the set of all conjuncts occurring on the toplevel of C.

The following example illustrates our notation.

Example 2.1.6 For $A \in \mathsf{N}_{\text{def}}$ and $P, Q \in \mathsf{N}_{\text{prim}}$, consider the TBox $\mathcal{T} := \{A \equiv P \sqcap Q \sqcap \exists r.(A \sqcap P)\}$. Then $\mathsf{deft}_{\mathcal{T}}(A) = P \sqcap Q \sqcap \exists r.(A \sqcap P)$ and $\mathsf{def}_{\mathcal{T}}(A) = \{P, Q, \exists r.(A \sqcap P)\}$.

In the next section, we introduce *descriptive semantics*, the standard semantics for TBoxes, CBoxes, and ABoxes. For TBoxes, we additionally introduce *greatest-fixedpoint semantics* in Section 2.1.2. Note that both semantics coincide on acyclic TBoxes. A comparison between both semantics w.r.t. cyclic TBoxes is presented in Section 2.2. We do not consider least-fixedpoint semantics here because our DL of main interest for fixedpoint semantics is $\mathcal{EL}$, for which it has been shown in [Baa03b] that least-fixedpoint semantics is not appropriate.

2.1.1 Descriptive semantics

Definition 2.1.7 (Descriptive semantics)
Every interpretation $\mathcal{I}$ is a *model of a TBox* $\mathcal{T}$ *(*$\mathcal{I} \models \mathcal{T}$*)* iff $A^{\mathcal{I}} = C^{\mathcal{I}}$ for every definition $A \equiv C \in \mathcal{T}$. $\mathcal{I}$ is a *model of a CBox* $\mathcal{C}$ *(*$\mathcal{I} \models \mathcal{C}$*)* iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every GCI $C \sqsubseteq D \in \mathcal{C}$ and $r_1^{\mathcal{I}} \circ \cdots \circ r_n^{\mathcal{I}} = r^{\mathcal{I}}$ for every $r_1 \circ \cdots \circ r_n \sqsubseteq r \in \mathcal{C}$, where '$\circ$' is interpreted as an associative binary operator with $r_1^{\mathcal{I}} \circ r_2^{\mathcal{I}} := \{(x, z) \mid (x, y) \in r_1^{\mathcal{I}} \wedge (y, z) \in r_2^{\mathcal{I}}\}$ for every $r_1, r_2 \in \mathsf{N}_{\text{role}}$. $\mathcal{I}$ is a *model of a general TBox* $\mathcal{T}$ iff it is a model of the CBox $\mathcal{T}$.

$\mathcal{I}$ is a model of an ABox $\mathcal{A}$ together with a CBox $\mathcal{C}$ iff $\mathcal{I} \models \mathcal{C}$, $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for every concept assertion $C(a) \in \mathcal{A}$, and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ for every role assertion $r(a, b) \in \mathcal{A}$. Furthermore, the *unique name assumption* applies, i.e., for every pair of individuals $a, b \in \mathsf{N}_{\text{nom}}$, $a \neq B$ implies $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. □

This semantics has been called *descriptive semantics* by Nebel [Neb91]. In case of an acyclic TBox $\mathcal{T}$, any interpretation of atomic concept names in $\mathcal{T}$ can be uniquely extended to an interpretation of all defined names in $\mathcal{T}$. This does not hold anymore for cyclic TBoxes interpreted with descriptive semantics. As an alternative semantics for cyclic TBoxes, we introduce greatest-fixedpoint semantics in the following section.

Note that every TBox can be interpreted as a general TBox (or CBox) since every definition $A \equiv C$ is equivalent to the pair of GCIs $A \sqsubseteq C$, $C \sqsubseteq A$.

One of the most basic inference services provided by DL systems is computing the subsumption hierarchy. The following definition introduces (descriptive) subsumption and most other standard inference problems formally.

Definition 2.1.8 (Inference problems)
For every CBox $\mathcal{C}$ and arbitrary concept descriptions C, D, *C is subsumed by D w.r.t. $\mathcal{C}$ ($C \sqsubseteq_\mathcal{C} D$)* iff $C^\mathcal{I} \subseteq D^\mathcal{I}$ for every model of $\mathcal{C}$. *C is equivalent to D w.r.t. $\mathcal{C}$ ($C \equiv_\mathcal{C} D$)* iff $C \sqsubseteq_\mathcal{C} D$ and $D \sqsubseteq_\mathcal{C} C$. Moreover, *$C$ strictly subsumes D w.r.t. $\mathcal{C}$ ($\mathcal{C} \sqsubset_\mathcal{C} D$)* iff $C \sqsubseteq_\mathcal{C} D$ and $D \not\sqsubseteq_\mathcal{C} C$.

A concept description C is *satisfiable w.r.t. a CBox $\mathcal{C}$ and an ABox $\mathcal{A}$* iff $\mathcal{A}$ and $\mathcal{C}$ have a common model $\mathcal{I}$ such that $C^\mathcal{I} \neq \emptyset$. Otherwise, C is *unsatisfiable w.r.t. $\mathcal{C}$ and $\mathcal{A}$*. An ABox $\mathcal{A}$ is *consistent together with a CBox $\mathcal{C}$* iff $\mathcal{A}$ and $\mathcal{C}$ have a common model.

An individual $a \in \mathsf{N}_{\text{nom}}$ is an *instance of a concept description C w.r.t. an ABox $\mathcal{A}$ together with a CBox $\mathcal{C}$* iff $a^\mathcal{I} \in C^\mathcal{I}$ for every common model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{C}$. □

By the above definition, all inference problems are implicitly defined for (general) TBoxes as well since every (general) TBox can be viewed as a CBox. Explicit reference to the empty TBox and the empty ABox may be omitted: if $\mathcal{T} = \emptyset$, write $C \sqsubseteq D$ instead of $C \sqsubseteq_\mathcal{T} D$, and analogously for strict subsumption and equivalence. Similarly, we say that, e.g., C is unsatisfiable iff it is unsatisfiable w.r.t. the empty TBox and the empty ABox.

Reasoning w.r.t. acyclic TBoxes can be reduced to reasoning w.r.t. the empty TBox by expanding[2] all definitions, i.e., by iteratively replacing all defined concept names occurring on right-hand sides by their respective definitions.

Depending on the underlying DL, inference problems can often be reduced to one another. The following list shows how all other inference problems can be reduced to subsumption and vice versa.

- *Satisfiability to (non-)subsumption*
  A concept C is satisfiable w.r.t. a CBox $\mathcal{C}$ iff $C \not\sqsubseteq_\mathcal{C} \bot$.
- *Subsumption to (un-)satisfiability*
  $C \sqsubseteq_\mathcal{C} D$ iff $C \sqcap \{a\}$ is unsatisfiable w.r.t. the CBox $\mathcal{C} \cup \{D \sqcap \{a\} \sqsubseteq \bot\}$, where a is a fresh individual name. Moreover, $C \sqsubseteq_\mathcal{C} D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{C}$.
- *Consistency to (non-)subsumption*
  $\mathcal{A}$ is consistent w.r.t. $\mathcal{C}$ iff $C_\mathcal{A} \not\sqsubseteq_\mathcal{C} \bot$, where $C_\mathcal{A}$ is defined as follows, using a fresh role name u:
  $$C_\mathcal{A} := \bigsqcap_{C(a) \in \mathcal{A}} \exists u.(\{a\} \sqcap C) \;\sqcap\; \bigsqcap_{r(a,b) \in \mathcal{A}} \exists u.(\{a\} \sqcap \exists r.\{b\}).$$
- *Subsumption to (in-)consistency*
  $C \sqsubseteq_\mathcal{C} D$ iff the ABox $\{C(a)\}$ is inconsistent w.r.t. the CBox $\mathcal{C} \cup \{D \sqcap \{a\} \sqsubseteq \bot\}$.
- *Instance problem to subsumption*
  An individual a is an instance of a concept C w.r.t. an ABox $\mathcal{A}$ and a CBox $\mathcal{C}$ iff $\{a\} \sqcap C_\mathcal{A} \sqsubseteq_\mathcal{C} C$, where $C_\mathcal{A}$ is defined as above.
- *Subsumption to the instance problem*
  $C \sqsubseteq_\mathcal{C} D$ iff a is an instance of D w.r.t. the ABox $\{C(a)\}$ together with $\mathcal{C}$.

Note that concept satisfiablility is trivial in a DL that cannot express inconsistencies.

[2] Note that here 'expanding' is used in the sense of [Neb90, BN03]. In the literature, this is sometimes also called 'unfolding', e.g. in [Küs01]. The latter term, however, has a different meaning in [Neb90].

2.1.2 Greatest-fixedpoint semantics

Throughout the present work, greatest-fixedpoint semantics is only relevant for the DL $\mathcal{EL}$. For this reason, we introduce this semantics only for $\mathcal{EL}$. A gfp-model for a given $\mathcal{EL}$-TBox $\mathcal{T}$ is obtained in two steps. At first, only primitive concepts and roles occurring in $\mathcal{T}$ are interpreted. This primitive interpretation is then extended to defined concept names by means of a fixedpoint iteration.

Definition 2.1.9 (Primitive interpretation)
Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox. A *primitive interpretation* $(\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ of $\mathcal{T}$ interprets all primitive concepts $P \in \mathsf{N}_{\text{prim}}$ by subsets of $\Delta^{\mathcal{J}}$ and all roles $r \in \mathsf{N}_{\text{role}}$ by binary relations on $\Delta^{\mathcal{J}}$. An Interpretation $\mathcal{I} := (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is *based on* $\mathcal{J}$ iff $\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}}$ and $\cdot^{\mathcal{J}}$ and $\cdot^{\mathcal{I}}$ coincide on N_{role} and N_{prim}. The set of all interpretations based on $\mathcal{J}$ is denoted by

$$\text{Int}(\mathcal{J}) := \{\mathcal{I} \mid \mathcal{I} \text{ is an interpretation based on } \mathcal{J}\}.$$

On $\text{Int}(\mathcal{J})$, a binary relation $\preceq_{\mathcal{J}}$ is defined for all $\mathcal{I}_1, \mathcal{I}_2 \in \text{Int}(\mathcal{J})$ by

$$\mathcal{I}_1 \preceq_{\mathcal{J}} \mathcal{I}_2 \quad \text{iff} \quad A^{\mathcal{I}_1} \subseteq A^{\mathcal{I}_2} \text{ for all } A \in \mathsf{N}_{\text{def}}^{\mathcal{T}}. \qquad \square$$

The pair $(\text{Int}(\mathcal{J}), \preceq_{\mathcal{J}})$ is a complete lattice, so that every subset of $\text{Int}(\mathcal{J})$ has a least upper bound (lub) and a greatest lower bound (glb) w.r.t. $\preceq_{\mathcal{J}}$. Hence, by Tarski's fixedpoint theorem [Tar55], every monotonic function on $\text{Int}(\mathcal{J})$ has a fixedpoint. In particular, this applies to the function $O_{\mathcal{T},\mathcal{J}}$ defined as follows.

Definition 2.1.10 Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox and $\mathcal{J}$ a primitive interpretation of N_{prim} and N_{role}. Then $O_{\mathcal{T},\mathcal{J}}$ is defined as follows.

$$O_{\mathcal{T},\mathcal{J}} \colon \text{Int}(\mathcal{J}) \to \text{Int}(\mathcal{J})$$
$$\mathcal{I}_1 \mapsto \mathcal{I}_2 \text{ iff } A^{\mathcal{I}_2} = C^{\mathcal{I}_1} \text{ for all } A \equiv C \in \mathcal{T}. \qquad \square$$

As shown in [Baa03b], $O_{\mathcal{T},\mathcal{J}}$ is in fact a fixedpoint operator on $\text{Int}(\mathcal{J})$. As a result, the following proposition holds.

Proposition 2.1.11 *Let $\mathcal{I}$ be an interpretation based on the primitive interpretation $\mathcal{J}$. Then $\mathcal{I}$ is a fixedpoint of $O_{\mathcal{T},\mathcal{J}}$ iff $\mathcal{I}$ is a model of $\mathcal{T}$.*

The general notion of fixedpoint models for $\mathcal{EL}$-TBoxes is defined as follows.

Definition 2.1.12 (Gfp-semantics)
Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox. The model $\mathcal{I}$ of $\mathcal{T}$ is called *gfp-model* iff there is a primitive interpretation $\mathcal{J}$ such that $\mathcal{I} \in \text{Int}(\mathcal{J})$ is the greatest fixedpoint of $O_{\mathcal{T},\mathcal{J}}$. $\square$

As $(\text{Int}(\mathcal{J}), \preceq_{\mathcal{J}})$ is a complete lattice, the gfp-model is uniquely determined for a given TBox $\mathcal{T}$ and a primitive interpretation $\mathcal{J}$. We may thus refer to *the* gfp-model $\text{gfp}(\mathcal{T}, \mathcal{J})$ for any given $\mathcal{T}$ and $\mathcal{J}$. With this preparation, we define gfp-subsumption as follows.

Definition 2.1.13 (Gfp-subsumption)
Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox and let $A, B \in \mathsf{N}_{\text{def}}^{\mathcal{T}}$. Then, *$A$ is subsumed by B w.r.t. gfp-semantics* ($A \sqsubseteq_{\text{gfp},\mathcal{T}} B$) iff $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ holds for all gfp-models $\mathcal{I}$ of $\mathcal{T}$. $\square$

Note that descriptive semantics considers a superset of the set of gfp-models, implying that descriptive subsumption entails gfp-subsumption. Hence, all subsumption relations w.r.t. $\sqsubseteq_{\mathcal{T}}$ also hold w.r.t. $\sqsubseteq_{\text{gfp},\mathcal{T}}$. Moreover, as mentioned above, both semantics coincide on acyclic TBoxes. For cyclic TBoxes, there is a notable difference between descriptive and gfp-semantics which will be discussed in further detail in the next section.

2.2 Cyclic TBoxes and their semantics

In an acyclic TBox $\mathcal{T}$, a concept definition $A \equiv C$ can be seen as merely the assignment of a new name A to the complex construct C. This changes in case of cyclic TBoxes interpreted with descriptive semantics, as the following example from [Baa03b] illustrates.

Example 2.2.1 Let $\mathsf{Node} \in \mathsf{N}_{\text{prim}}$ be an atomic concept and $\mathsf{edge} \in \mathsf{N}_{\text{role}}$ be a role. Consider the simple $\mathcal{EL}$-TBox $\mathcal{T} := \{\mathit{Inode} \equiv \mathsf{Node} \sqcap \exists \mathsf{edge}.\mathit{Inode}\}$ in which the concept Inode is defined to represent nodes on infinite paths in directed graphs.

Consider a primitive interpretation $\mathcal{J}$ defined by

$$\begin{aligned} \Delta^{\mathcal{J}} &:= \{x_n \mid n \in \mathbb{N}\} \cup \{y\} \\ \mathsf{Node}^{\mathcal{J}} &:= \Delta^{\mathcal{J}} \\ \mathsf{edge}^{\mathcal{J}} &:= \{(x_n, x_{n+1}) \mid n \in \mathbb{N}\} \cup \{(y, y)\}. \end{aligned}$$

Hence, $\mathcal{J}$ defines an infinite path over elements x_n with $n \in \mathbb{N}$ and additionally a loop over y. In order to extend $\mathcal{J}$ to an interpretation of $\mathcal{T}$, it remains to interpret Inode. To this end, we define four different extensions of $\mathcal{J}, \mathcal{I}_1, \ldots, \mathcal{I}_4$ by

$$\begin{aligned} \mathit{Inode}^{\mathcal{I}_1} &:= \emptyset \\ \mathit{Inode}^{\mathcal{I}_2} &:= \{x_n \mid n \in \mathbb{N}\} \\ \mathit{Inode}^{\mathcal{I}_3} &:= \{y\} \\ \mathit{Inode}^{\mathcal{I}_4} &:= \Delta^{\mathcal{J}}. \end{aligned}$$

It is easy to verify that all of the above interpretations are models of $\mathcal{T}$ interpreted with descriptive semantics. In contrast, only $\mathcal{I}_4$ is a gfp-model of $\mathcal{T}$. Clearly, the intuition behind the definition of Inode is captured only by $\mathcal{I}_4$. □

The restriction of gfp-models to interpret all defined names as large as possible has the consequence that the names of defined concepts as such are not as important in gfp-semantics as in descriptive semantics, as the following example from [Baa03b] shows.

Example 2.2.2 Consider a TBox $\mathcal{T}$ defined over the set $\mathsf{N}_{\text{prim}} := \{\mathsf{Mammal}\}$ of primitive concepts and the set $\mathsf{N}_{\text{role}} := \{\mathsf{has\_parent}\}$ of roles. $\mathcal{T}$ defines two concepts, Lion and Tiger, as follows.

$$\begin{aligned} \mathit{Lion} &\equiv \mathsf{Mammal} \sqcap \exists \mathsf{has\_parent}.\mathit{Lion} \\ \mathit{Tiger} &\equiv \mathsf{Mammal} \sqcap \exists \mathsf{has\_parent}.\mathit{Tiger} \end{aligned}$$

Interpreting $\mathcal{T}$ with gfp-semantics makes Lion and Tiger equivalent while they are not if $\mathcal{T}$ is interpreted with descriptive semantics. □

We sum up our comparison by recalling some of the arguments presented in [Neb91], where the advantages and disadvantages of descriptive versus fixed-point semantics have been discussed in depth.

- *Descriptive semantics*: is conceptually the most straightforward generalization of the standard semantics for acyclic TBoxes and can be applied to arbitrary DLs. On the other hand, deciding subsumption w.r.t. descriptive semantics can be more complicated than in the case of gfp-semantics.

- *Gfp-semantics*: naturally extends the property of acyclic TBoxes that the interpretation of defined concepts is determined by that of atomic concepts and roles. Moreover, fixedpoint semantics seems appropriate when concepts are viewed under a more structural perspective. On the negative side, gfp-semantics cannot always be extended to expressive DLs.

It should be added that gfp-semantics cannot be applied to general TBoxes and thus also not to CBoxes. Moreover, all modern DL systems, such as FaCT or Racer, rely on descriptive semantics only. Nevertheless, in Section 2.4 we will discuss a TBox formalism that in some sense combines the advantages of both semantics.

It has been pointed out in [Neb91] that, independently of the choice of semantics, there are appropriate and inappropriate ways of using cycles in concept definitions, depending on whether roles are involved in the cycles or not. A concept A in a TBox $\mathcal{T}$ is called *component-circular* iff the definition of A refers directly or indirectly to itself *without* using existential or value restrictions to establish that cycle. Consider the following component-circular concepts from [Neb91].

Example 2.2.3 Let $\mathcal{T}$ be a TBox defined over the set of atomic concepts $\mathsf{N}_{\text{prim}} := \{\mathsf{Human}\}$ and an empty set of roles. In $\mathcal{T}$, the two concepts *Man* and *MaleHuman* are defined as follows.

$$\begin{aligned} \mathit{Man} &\equiv \mathsf{Human} \sqcap \mathit{MaleHuman} \\ \mathit{MaleHuman} &\equiv \mathsf{Human} \sqcap \mathit{Man} \end{aligned}$$

Obviously, the concept *Man* is defined as a specialization of *MaleHuman* and vice versa, which not only in this case but in general does not seem to make sense. □

All concepts with a cyclic definition that is not component-circular are called *restriction-circular*. Hence, in this case the cyclic definition is established using existential or value restrictions. Consider the following example, again adapted from [Neb91].

Example 2.2.4 Using the atomic concept RootNode and the role branch, the concept of binary trees can be defined by means of a restriction-circular concept *BinaryTree* by

$$\mathit{BinaryTree} \equiv \mathsf{RootNode} \sqcap (\leqslant 2\ \mathsf{branch}) \sqcap \forall \mathsf{branch}.\mathit{BinaryTree}.$$

According to the definition, binary trees comprise a root node with at most two branches again pointing to binary trees. □

In [Neb91], a 'third kind of terminological cycles' is discussed, where two restriction-circular concepts refer to each other, as the following example illustrates.

Example 2.2.5 Using the atomic concepts Vehicle and Engine, and roles engine_part and is_engine_of, we define the relationship between cars and their engines by

$$\begin{aligned} \mathit{Car} &\equiv \mathsf{Vehicle} \sqcap \exists \mathsf{engine\_part}.\mathit{CarEngine} \\ \mathit{CarEngine} &\equiv \mathsf{Engine} \sqcap \exists \mathsf{is\_engine\_of}.\mathit{Car}, \end{aligned}$$

thus specifying a cyclic relationship between *Car* and *CarEngine*. □

It has been pointed in [Neb91] that concept definitions of the above kind are not 'well-founded' in the sense that they admit counterintuitive models. In the following example, we show how one such model could look like.

Example 2.2.6 For instance, the above example definitions for cars and their engines are satisfied by an interpretation $\mathcal{I}$ defined by

$$\begin{aligned}
\Delta^{\mathcal{I}} &:= \{c_n \mid n \in \mathbb{N}\} \sqcap \{e_n \mid n \in \mathbb{N}\} \\
\mathit{Car}^{\mathcal{I}} := \mathsf{Vehicle}^{\mathcal{I}} &:= \{c_n \mid n \in \mathbb{N}\} \\
\mathit{CarEngine}^{\mathcal{I}} := \mathsf{Engine}^{\mathcal{I}} &:= \{e_n \mid n \in \mathbb{N}\} \\
\mathsf{engine_part}^{\mathcal{I}} &:= \{(c_n, e_n) \mid n \in \mathbb{N}\} \\
\mathsf{is_engine_of}^{\mathcal{I}} &:= \{(e_n, c_{n+1}) \mid n \in \mathbb{N}\}.
\end{aligned}$$

Hence, we have infinitely many instances c_n of *Car* and infinitely many instances e_n of *CarEngine*. As the interpretation of engine_part shows, e_n is the engine of c_n for every $n \in \mathbb{N}$. Unfortunately, by the interpretation of is_engine_of, e_n is the engine of the *next* car, i.e., c_{n+1} and not c_n. It is easy to see that $\mathcal{I}$ is a model of $\mathcal{T}$. □

It should be clarified, however, that similar effects to the ones described above can also occur for simpler restriction-circular concepts. For instance, the definition of the concept *BinaryTree* from Example 2.2.4 can be satisfied by a circular model $\mathcal{I}$ defined by, e.g., $\Delta^{\mathcal{I}} := \mathit{BinaryTree}^{\mathcal{I}} := \{x\}$ and $\mathsf{branch}^{\mathcal{I}} := \{(x, x)\}$. In the case of cars and their engines in Example 2.2.5, inverse roles must be expressible in order to avoid acyclic models, such as the one from Example 2.2.6.

Note also that unintended models of a TBox $\mathcal{T}$ might do little harm as long as the main interest is in the subsumption relations between concepts in $\mathcal{T}$, because the definition of subsumption is quantified over *all* models of the TBox.

In the present work, several DL formalisms are discussed in which cyclic definitions for concepts can be expressed. This holds for $\mathcal{EL}^{++}$-CBoxes examined in depth in Section 3.2, for hybrid $\mathcal{EL}$-TBoxes introduced in Section 2.4 and examined further in Section 3.3, and clearly also for cyclic $\mathcal{EL}$-TBoxes studied in further detail in Section 3.3.1.

2.3 Concrete domains

As defined so far, DL knowledge bases are well suited to define *structural* properties of notions relevant in some application domain; especially interrelations between notions, which can be expressed by means of roles. In real-world applications, however, not only qualitative but also quantitative properties play an important role, e.g., weights, sizes, time intervals, spacial relations, etc. For instance, in a healthcare system, it might be necessary to state that a patient has a systolic blood pressure of 150 mmHg, and the underlying KR system might be expected to infer from this information that the patient's blood pressure is elevated. Information of this kind cannot be represented adequately by the DL formalisms introduced above.

In order to refer to domain such as real numbers, rational numbers, integers, or strings in DL knowledge bases, *concrete domains* have been devised [BH91, Lut03]. Intuitively, a concrete domain consists of an arbitrary domain and a fixed set of predicates over this domain. The predicates can be used in concept descriptions by means of dedicated roles. Therefore, we require an additional set N_{fe} of *feature names* pairwise disjoint to $\mathsf{N}_{\mathrm{con}}$, $\mathsf{N}_{\mathrm{role}}$, and $\mathsf{N}_{\mathrm{nom}}$. Every feature $f \in \mathsf{N}_{\mathrm{fe}}$ must be interpreted by a (partial) function. We are now ready to introduce concrete domains formally.

Definition 2.3.1 (Concrete domain)
Let $\Delta^{\mathcal{D}}$ be a nonempty set and let $\mathcal{P}^{\mathcal{D}}$ be a set of *predicate names* such that every $p \in \mathcal{P}^{\mathcal{D}}$

has an arity $n \in \mathbb{N} \setminus \{0\}$ and an extension $p^{\mathcal{D}} \subseteq (\Delta^{\mathcal{D}})^n$. Then $\mathcal{D} := (\Delta^{\mathcal{D}}, \mathcal{P}^{\mathcal{D}})$ is a *concrete domain*. For every $p \in \mathcal{P}^{\mathcal{D}}$ with arity n and features $f_1, \ldots, f_n \in \mathsf{N}_{\text{fe}}$, $p(f_1, \ldots, f_n)$ is a *concrete domain restriction*. Given an interpretation $\mathcal{I}$, $p(f_1, \ldots, f_n)$ is interpreted as follows.

$$p(f_1, \ldots, f_n)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid \exists y_1, \ldots, y_n \in \Delta^{\mathcal{D}} \colon f_i^{\mathcal{I}}(x) = y_i \text{ for all } 1 \leq i \leq n \\ \wedge (y_1, \ldots y_n) \in p^{\mathcal{D}}\}.$$ □

Note that DLs may be equipped with finitely many concrete domains $\mathcal{D}_1, \ldots, \mathcal{D}_m$ as long as their respective domains $\Delta^{\mathcal{D}_1}, \ldots, \Delta^{\mathcal{D}_m}$ are pairwise disjoint. The extension of a DL $\mathcal{L}$ by concrete domains $\mathcal{D}_1, \ldots, \mathcal{D}_m$ is usually denoted by $\mathcal{L}(\mathcal{D}_1, \ldots, \mathcal{D}_m)$. The following simple example shows how concrete domains can be used to incorporate quantitative information in the definition of concepts of GCIs.

Example 2.3.2 Consider the concrete domain $\mathcal{Q} = (\mathbb{Q}^{\mathcal{Q}}, \mathcal{P}^{\mathcal{Q}})$ defined over the set of rational numbers with a predicate set $\mathcal{P}^{\mathcal{Q}}$ containing the following predicates:

- a unary predicate $\geq_{18}$ interpreted by $\geq_{18}^{\mathcal{Q}} := \{x \in \mathbb{Q} \mid x \geq 18\}$; and
- a binary predicate $<$ interpreted by $<^{\mathcal{Q}} := \{(x, y) \in \mathbb{Q}^2 \mid x < y\}$.

Furthermore, let $\mathsf{N}_{\text{fe}} := \{\mathsf{has\_age}, \mathsf{has\_diastolic\_bp\_mmHg}, \mathsf{has\_systolic\_bp\_mmHg}\}$. As an example of an $\mathcal{EL}(\mathcal{Q})$-concept description, we might define the concept of a grown-up person as a person at least 18 years old, i.e.,

$$\mathit{GrownUpPerson} \equiv \mathsf{Person} \sqcap \geq_{18}(\mathsf{has\_age}),$$

thus using the concrete domain restriction $\geq_{18}(\mathsf{has\_age})$. Moreover, in order to state that the diastolic blood pressure of every person is less than or equal than his systolic blood pressure, we can add the following GCI to our TBox.

$$<(\mathsf{has\_systolic\_bp\_mmHg}, \mathsf{has\_diastolic\_bp\_mmHg}) \sqsubseteq \bot$$ □

Much more sophisticated concrete domains have been studied in depth. For instance, temporal extensions of DLs can be defined by a concrete domain representing time intervals. Relationships between time intervals can be expressed by means of the Allen-relations [All83] which can be represented by concrete domain predicates. Moreover, there exist concrete domains to represent qualitative spacial relations in topological spaces based on the RCC-8 relations [RCC92]. For an overview, see [Lut03].

In the context of $\mathcal{EL}^{++}$-TBoxes, we are concerned with a special class of concrete domains, so-called *p-admissible* concrete domains. These are defined as follows.

Definition 2.3.3 A concrete domain $\mathcal{D}$ is *p-admissible* iff

1. satisfiability and implication in $\mathcal{D}$ are decidable in polynomial time; and
2. $\mathcal{D}$ is *convex*, i.e., if a conjunction of atoms of the form $p(f_1, \ldots, f_k)$ implies a disjunction of such atoms, then it also implies one of its disjuncts. □

We investigate the property of p-admissibility in more detail in Section 3.2.3, where we also exhibit some useful concrete domains that actually are p-admissible.

Apart from $\mathcal{EL}^{++}$-CBoxes, another novel DL formalism is discussed in the present work, namely hybrid $\mathcal{EL}$-TBoxes. These are introduced formally in the following section, which finishes our chapter of formal preliminaries.

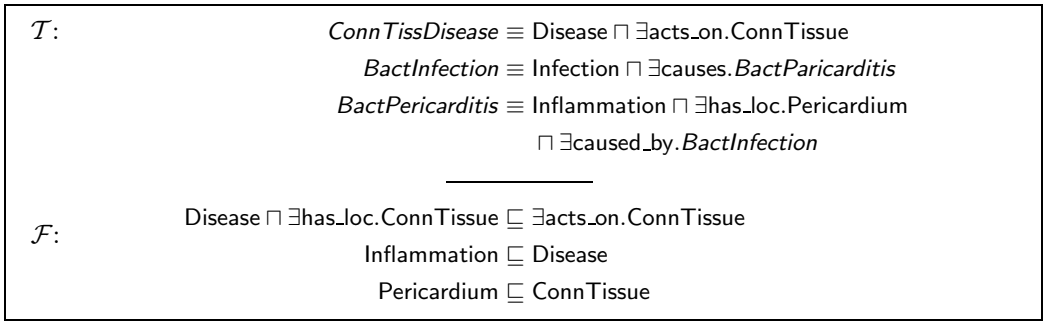


Figure 2.4.1: Example hybrid $\mathcal{EL}$-TBox

2.4 Hybrid TBoxes

In Section 3.3, a novel type of terminologies is examined in depth, so-called hybrid TBoxes. In the present section, their syntax and semantics is defined formally.

Definition 2.4.1 (Hybrid TBox)
For every general $\mathcal{L}$-TBox $\mathcal{F}$ over N_{prim} and N_{role}, and every $\mathcal{L}$-TBox $\mathcal{T}$ over N_{def}, N_{prim}, and N_{role}, the pair $(\mathcal{F}, \mathcal{T})$ is called a *hybrid $\mathcal{L}$-TBox*. □

The semantics of hybrid TBoxes are defined as follows.

Definition 2.4.2 (Semantics)
Let $(\mathcal{F}, \mathcal{T})$ be a hybrid TBox over N_{prim}, N_{role}, and N_{def}. A primitive interpretation $\mathcal{J}$ is a *model of $\mathcal{F}$ ($\mathcal{J} \models \mathcal{F}$)* iff $C^{\mathcal{J}} \subseteq D^{\mathcal{J}}$ for every GCI $C \sqsubseteq D$ in $\mathcal{F}$. A model $\mathcal{I} \in \text{Int}(\mathcal{J})$ is a *gfp-model* of $(\mathcal{F}, \mathcal{T})$ iff $\mathcal{J} \models \mathcal{F}$ and $\mathcal{I}$ a gfp-model of $\mathcal{T}$. □

Note that $\mathcal{F}$ ("foundation") is interpreted w.r.t. descriptive semantics while $\mathcal{T}$ ("terminology") is interpreted w.r.t. gfp-semantics. Note also that every gfp-model of $(\mathcal{F}, \mathcal{T})$ can be expressed as the greatest fixedpoint $\text{gfp}(\mathcal{T}, \mathcal{J})$ for some primitive interpretation $\mathcal{J}$ with $\mathcal{J} \models \mathcal{F}$.
In order to complete the semantics of hybrid TBoxes, we still have to introduce an appropriate notion of subsumption.

Definition 2.4.3 (Subsumption w.r.t. hybrid TBoxes)
Let $(\mathcal{F}, \mathcal{T})$ be a hybrid $\mathcal{L}$-TBox over N_{prim}, N_{role}, and N_{def}. Let A, B be defined concepts in $\mathcal{T}$. Then *A is subsumed by B w.r.t. $(\mathcal{F}, \mathcal{T})$ ($A \sqsubseteq_{\text{gfp},\mathcal{F},\mathcal{T}} B$)* iff $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ for all gfp-models $\mathcal{I}$ of $(\mathcal{F}, \mathcal{T})$. □

The following example is supposed to give an impression of how an actual hybrid TBox might look like.

Example 2.4.4 Consider Figure 2.4.1, showing a simplified part of a medical terminology[3] represented by a hybrid TBox $(\mathcal{F}, \mathcal{T})$. $\mathcal{T}$ is supposed to define the concepts 'disease of the connective tissue', 'bacterial infection' and 'bacterial pericarditis'. For instance, bacterial Pericarditis is defined as an inflammation located in the Pericardium caused by a bacterial infection. Note that $\mathcal{T}$ is cyclic. For the primitive concepts in $\mathcal{T}$, the foundation $\mathcal{F}$ states, e.g., that a disease located in connective tissue acts on connective tissue. □

[3]Our example is only supposed to show the features of hybrid $\mathcal{EL}$-TBoxes and in no way claims to be adequate from a Medical KR perspective.

Hybrid TBoxes generalize cyclic TBoxes with gfp-semantics in the sense that every cyclic $\mathcal{L}$-TBox $\mathcal{T}$ can be viewed as a hybrid TBox with an empty foundation. Thus, gfp-subsumption w.r.t. $\mathcal{T}$ coincides with subsumption w.r.t. the hybrid TBox $(\emptyset, \mathcal{T})$. Also note that, every general TBox $\mathcal{T}'$ can be seen as a hybrid TBox $(\mathcal{T}', \emptyset)$. In this case, a descriptive subsumption $P \sqsubseteq_{\mathcal{T}'} Q$ holds iff A_P is subsumed by A_Q holds w.r.t. the normalized instance of $(\mathcal{T}', \emptyset)$.

Throughout the present work, we will be concerned only with hybrid $\mathcal{EL}$-TBoxes. In Section 3.3, we show how to classify hybrid $\mathcal{EL}$-TBoxes in polynomial time. Moreover, Section 4.4.6 presents a matching algorithm that can be used on hybrid $\mathcal{EL}$-TBoxes. In this section, we shall also see how to compute the lcs and msc w.r.t. hybrid $\mathcal{EL}$-TBoxes.

CHAPTER 3

SUBSUMPTION

In the present chapter, we discuss several extensions of the small DL $\mathcal{EL}$ and study the complexity of the relevant subsumption problems. By 'extensions' both additional language constructors and more expressive TBox formalisms are implied. As mentioned previously, our main motive is to find 'useful' extensions of $\mathcal{EL}$ in which terminological reasoning is still tractable.

In Section 3.1, we begin by showing that tractability of reasoning in extensions of $\mathcal{EL}$ is a property easily lost. For several common DL-constructors, adding one of these to $\mathcal{EL}$ makes the subsumption problem, and thus classification, intractable. Contrasting this, the main positive result of this chapter is presented in Section 3.2, where the subsumption problem in a substantial extension of $\mathcal{EL}$ is shown to be tractable even w.r.t. CBoxes, an extension of general TBoxes.

In Section 3.3, a new type of terminologies is discussed, so-called *hybrid* terminologies. We show that reasoning in $\mathcal{EL}$ w.r.t. hybrid TBoxes is tractable and argue that hybrid terminologies combine advantages of general TBoxes on the one side and ordinary cyclic TBoxes on the other.

3.1 Intractable extensions of $\mathcal{EL}$

For several DL-constructors introduced in Definitions 2.1.1 and 2.1.3, we show that extending $\mathcal{EL}$ by one of these constructors renders the subsumption problem intractable. More precisely, we distinguish between three variants of the subsumption problem, namely subsumption without TBoxes (Section 3.1.1), w.r.t. acyclic TBoxes (Section 3.1.2), or w.r.t. general TBoxes (Section 3.1.3). Note that all hardness results automatically carry over to the more general TBox formalism, i.e., from the empty TBox to acyclic TBoxes to general TBoxes.

3.1.1 Subsumption w.r.t. the empty TBox

We show that adding one of the constructors value restriction ($\forall$), disjunction ($\sqcup$), or number restrictions ($\leqslant n, \geqslant n$) makes the subsumption problem w.r.t. the empty TBox intractable. In the case of value restrictions, we show the relevant NP-completeness result from the literature, while the other cases yield co-NP-hardness by reduction to well-known intractability results.

Value restriction

Let $\mathcal{EL}_\forall$ be the extension of $\mathcal{EL}$ by the constructor $\forall$ introduced in Definitions 2.1.1 and 2.1.3. In [DLN+92], the following theorem has been shown.

Theorem 3.1.1 *Deciding subsumption of $\mathcal{EL}_\forall$-concept descriptions is NP-complete.*

Note that in [DLN+92], the DL $\mathcal{EL}_\forall$ is called $\mathcal{FLE}^-$. W.r.t. general $\mathcal{EL}_\forall$-TBoxes, deciding subsumption is even EXPTIME-complete, as we show in Section 3.1.3.

Disjunction

We show that subsumption in $\mathcal{ELU}$ is co-NP-complete. The upper bound holds due to a co-NP-result for a complementary DL called $\mathcal{ALU}$ in [SSS91]. For arbitrary $\mathcal{ELU}$-concepts C, D, $C \sqsubseteq D$ iff $\neg D \sqsubseteq \neg C$. The negation normal form of $\neg C$ and $\neg D$ can be expressed using only bottom concept, atomic negation, conjunction, disjunction, and value restriction. Hence, we obtain concept descriptions in the DL $\mathcal{ALU}$, where all relevant constructors (plus restricted existential quantification $\exists r.\top$) are provided. Subsumption in $\mathcal{ELU}$ can thus be reduced to subsumption in $\mathcal{ALU}$ shown to be co-NP-complete in [SSS91].

The lower bound is by reducing MONOTONE 3SAT to non-subsumption of $\mathcal{ELU}$-concept descriptions. The monotone problem differs from 3SAT only in that every clause contains either only negated or only unnegated literals.

Definition 3.1.2 (MONOTONE 3SAT)
Let U be a set of variables and S^+, S^- be two sets of clauses over U such that every $s \in S^+$ contains exactly 3 un-negated variables and every $s \in S^-$ exactly 3 negated ones. Then, $P := (U, S^+, S^-)$ is called a *Monotone 3Sat* problem. A *solution* to P is a truth assignment $t\colon U \to \{0,1\}$ satisfying $S^+ \cup S^-$.

MONOTONE 3SAT is an NP-complete problem [GJ79, p. 259]. We can immediately represent the clauses in S^+ and S^- in $\mathcal{ELU}_\neg$, an extension of $\mathcal{ELU}$ by atomic negation. The conjunction over all clauses is then split into $C \sqcap D$, C containing all positive clauses and D all negative ones. Satisfiability of $C \sqcap D$ is reduced to $\mathcal{ELU}$-non-subsumption by deciding $C \not\sqsubseteq \mathrm{nnf}(\neg D)$, where nnf denotes the negation normal form of $\neg D$. The following lemma provides the formal proof.

Lemma 3.1.3 *Let $P = (U, S^+, S^-)$ be a Monotone 3Sat problem. Then there exist $\mathcal{ELU}$-concept descriptions C, D such that P has a solution iff $C \not\sqsubseteq D$.*

PROOF. Let $\mathsf{N}_{\mathrm{prim}} := U$ and $\mathsf{N}_{\mathrm{role}} := \emptyset$. We can immediately translate $S^+ \cup S^-$ into an $\mathcal{ELU}_\neg$-concept description C^P of the following form:

$$C^P := \bigsqcap_{s \in S^+} \bigsqcup_{u \in s} u \sqcap \bigsqcap_{s \in S^-} \bigsqcup_{\neg u \in s} \neg u.$$

Clearly, P has a solution iff C^P is satisfiable. The satisfiability of C^P is equivalent to the non-subsumption

$$C := \bigsqcap_{s \in S^+} \bigsqcup_{u \in s} u \not\sqsubseteq \neg \bigsqcap_{s \in S^-} \bigsqcup_{\neg u \in s} \neg u \equiv \bigsqcup_{s \in S^-} \bigsqcap_{\neg u \in s} u =: D.$$

Observe that both C and D are concept descriptions in $\mathcal{ELU}$. □

Theorem 3.1.4 *Deciding subsumption of $\mathcal{ELU}$-concept descriptions w.r.t. the empty TBox is co-NP-complete.*

The above reduction implies co-NP-completeness of the subsumption problem even for the very small description logic providing only conjunction and disjunction.

Number restrictions

We show that subsumption in $\mathcal{ELN}$ is co-NP-complete. Membership in co-NP is shown similarly to the previous case. For $\mathcal{ELN}$-concepts C, D, $C \sqsubseteq D$ iff $\neg D \sqsubseteq \neg C$. For every $n \in \mathbb{N}$ and every role $r \in \mathsf{N}_{\text{role}}$, the following equivalences hold.

$$\begin{aligned} \neg(\leqslant n\ r) &\equiv (\geqslant n+1\ r) \\ \neg(\geqslant 0\ r) &\equiv \bot \\ \neg(\geqslant n+1\ r) &\equiv (\leqslant n\ r) \end{aligned}$$

Hence, $\neg C$ and $\neg D$ can be expressed by the constructors bottom concept, atomic negation, conjunction, disjunction, value restriction, and number restriction. These are provided in the DL $\mathcal{ALUN}$ (as well as unqualified existential restriction $\exists r.\top$) examined in [DLNN95], where co-NP-completeness (in the strong sense) of the subsumption problem and NP-completeness (in the strong sense) of concept satisfiability have been shown.

The lower bound is by reduction of BIN-PACKING to consistency of $\mathcal{ELN}$ concepts. Since $\mathcal{ELN}$ can express inconsistency as $(\leqslant 0\ r) \sqcap (\geqslant 1\ r)$, inconsistency can be reduced to non-subsumption of $\mathcal{ELN}$ concepts, yielding the desired reduction.

Definition 3.1.5 (BIN-PACKING)
Let U be a nonempty finite set. Let $s\colon U \to \mathbb{N}^+$ and let $b, k \in \mathbb{N}^+$. Then, $P := (U, s, b, k)$ is a *Bin-packing* problem. A *solution* to P is a partition of U into k pairwise disjoint sets $U_1, \ldots, U_k$ such that for all $i \in \{1, \ldots, k\}$ it holds that $\Sigma_{u \in U_i} s(u) \leq b$.

BIN-PACKING is an NP-complete problem in the strong sense [GJ79, p. 226], implying that we may assume unary encoding for the numbers in P. Given P, we construct a concept C_P which is satisfiable iff P has a solution.

The intuition behind C_P is to use a concept description of fixed depth 2 and, (i) express on toplevel that at most k bins, i.e., k pairwise disjoint sets $U_1, \ldots, U_k$, exist, (ii) express on the first role level that every bin weighs at most b, and (iii) use the second role level to represent the weights $s(u)$ of the objects $u \in U$. The following definition formalizes this notion.

Definition 3.1.6 (Bin-packing concept)
Let $P = (U, s, b, k)$ be a Bin-packing problem. Let $\ell := \lceil \lg(\Sigma_{u \in U} s(u)) \rceil$. Define $\mathsf{N}^P_{\text{prim}} := \emptyset$ and $\mathsf{N}^P_{\text{role}} := \{r\} \cup \{r_1, \ldots, r_\ell\}$. Let

$$\mathcal{C}^P := \left\{ \prod_{i=1}^{\ell} C_i \;\middle|\; C_i \in \{(\leqslant 0\ r_i), (\geqslant 1\ r_i)\} \right\}$$

Let $f^P\colon \{(u,i) \mid u \in U, 1 \leq i \leq s(u)\} \to \mathcal{C}^P$ be an injective mapping. The $\mathcal{ELN}$-concept description C^P is defined as follows:

$$C^P := (\leqslant k\ r) \sqcap \bigsqcap_{u \in U} \exists r. \left((\leqslant b\ r) \sqcap \prod_{i=1}^{s(u)} \exists r. f^P(u,i) \right)$$

Note that $\Sigma_{u \in U} s(u) \leq |\mathcal{C}^P| < 2 \cdot \Sigma_{u \in U} s(u)$ so that f^P in fact exists and can be computed easily in polynomial time in the size of P with unary number encoding. The above definition is well-defined only w.r.t. the mapping f^P of which in general many different ones exist. Nevertheless, for our purpose an arbitrary but fixed instance f^P suffices.

The motivation behind the function $f(u,i)$ is to provide a simple method to count binarily from 0 to $\Sigma_{u\in U}s(u)$, the sum of the weights of all elements in U. The concept descriptions $f(u,i)$ on the second role level, i.e., in the leaves of C_P, enforce that no two leaves can be represented by the same element in a model of C_P. This is guaranteed by the fact that two arbitrary but different leaves in C_P differ in negating or not negating at least one number restriction for a role r_i. The following lemma proves formally that the reduction is correct.

Lemma 3.1.7 *Let $P = (U, s, b, k)$ be a Bin-Packing problem and C^P the corresponding concept description over $\mathsf{N}^P_{\text{prim}}$ and $\mathsf{N}^P_{\text{role}}$. Then,*

1. *For every $u, v \in U$, $i \in \{1, \ldots, s(u)\}$, and $j \in \{1, \ldots, s(v)\}$ it holds that $f^P(u,i) \sqcap f^P(v,j) \equiv \bot$ iff $u \neq v$ or $i \neq j$.*
2. *P has a solution iff C^P is satisfiable.*

Proof. 1. ($\Leftarrow$) If $u = v$ and $i = j$ then $f^P(u,i) \sqcap f^P(v,j) \equiv f^P(u,i) \in \mathcal{C}^P$. Every concept description in $\mathcal{C}^P$ is consistent because each of its conjuncts imposes a number restriction on a different role.

($\Rightarrow$) Then the injectivity of f^P implies that $f^P(u,i)$ and $f^P(v,j)$ are two distinct concepts in $\mathcal{C}^P$. Hence, there exists an index t such that $f^P(u,i)$ contains the conjunct $(\leqslant 0\ r_t)$ and $f^P(u,i)$ contains the conjunct $(\geqslant 1\ r_t)$ or vice versa. Hence, the conjunction $f^P(u,i) \sqcap f^P(v,j)$ is subsumed by $(\leqslant 0\ r_t) \sqcap (\geqslant 1\ r_t) \equiv \bot$.

2. ($\Rightarrow$) Denote by $U_1, \ldots, U_k$ a solution to P. Define a model $\mathcal{I}$ of C^P as follows:

$$\Delta^{\mathcal{I}} := \{w, z\} \cup \{x_i \mid 1 \leq i \leq k\} \cup \{y_{uj} \mid u \in U,\ 1 \leq j \leq s(u)\}.$$

Let

$$r^{\mathcal{I}} := \bigcup_{i=1}^{k} \big(\{(w, x_i)\} \cup \{(x_i, y_{uj}) \mid u \in U_i,\ 1 \leq j \leq s(u)\}\big)$$

and for every $t \in \{1, \ldots, \ell\}$, let

$$r_t^{\mathcal{I}} := \{(y_{uj}, z) \mid u \in U,\ 1 \leq j \leq s(u),\ f^P(u,j) \sqsubseteq (\geqslant 1\ r_t)\}.$$

We show that $w \in C^{P\mathcal{I}}$, i.e., w is a witness of C^P. The definition of $r^{\mathcal{I}}$ shows that w has exactly k successors w.r.t. the role r, namely $x_1, \ldots, x_k$. Hence, the number restriction on the toplevel of C^P is satisfied. For the rest of C^P, consider an arbitrary $u \in U$ and select $i \in \{1, \ldots, k\}$ such that $u \in U_i$. It suffices to show that $x_i \in (\leqslant b\ r)^{\mathcal{I}}$ and that $x_i \in (\exists r.f^P(u,j))^{\mathcal{I}}$ for all $1 \leq j \leq s(u)$.

Due to the definition of $r^{\mathcal{I}}$, x_i has exactly one successor y_{ui} for every element $u \in U_i$ and for every $1 \leq j \leq s(u)$. Hence, the total number of successors of x_i equals $\Sigma_{u\in U_i}s(u)$ which does not exceed b, the size limit for every U_i.

Consider an arbitrary $j \in \{1, \ldots, s(u)\}$. Since $(x_i, y_{uj}) \in r^{\mathcal{I}}$ it suffices to show that $y_{uj} \in f^P(u,j)^{\mathcal{I}}$. By definition, $f(u,j) = \bigsqcap_{t=1}^{\ell} C_t$ with $C_t = (\leqslant 0\ r_t)$ or $C_t = (\geqslant 1\ r_t)$ for every $t \in \{1, \ldots, \ell\}$. For every t, the pair (y_{uj}, z) occurs in $r_t^{\mathcal{I}}$ iff $C_t = (\geqslant 1\ r_t)$. Hence y_{uj} has no successor w.r.t. every role r_t occurring in a number restriction $(\leqslant 0\ r_t)$ and has t as successor w.r.t. every role r_t occurring in a number restriction $(\geqslant 1\ r_t)$. Thus, y_{uj} is a witness of $f^P(u,j)$.

($\Leftarrow$) To simplify notation, for all $u \in U$ let

$$C^u := (\leqslant b\ r) \sqcap \bigsqcap_{j=1}^{s(u)} \exists r.f^P(u,j).$$

Hence, C^P can be written as

$$C^P = (\leqslant k\ r) \sqcap \bigsqcap_{u \in U} \exists r.C^u.$$

Denote by $\mathcal{I}$ a model of C^P and denote by w a witness $w \in C^{P\mathcal{I}}$. Due to the number restriction on the toplevel of C^P, w has at most k successors w.r.t. r. The $|U|$ existential restrictions on the other hand guarantee that at least one successor exists. Denote by $X := \{x_1, \ldots, x_{k'}\}$ the set of r-successors of w. If $k' < k$ then w.l.o.g., $k - k'$ isolated vertices $x_{k'+1}, \ldots, x_k$ may be added to $\Delta^{\mathcal{I}}$.

Define the partition of U as follows: starting from 1, for $i = 1, \ldots, k$ let

$$U_i := \{u \in U \mid x_i \in C^{u\mathcal{I}},\ \forall j < i\colon u \notin U_j\}.$$

Note that the above definition is well-defined only w.r.t. an order on $\{1, \ldots, k\}$ by which to compute the U_i. We have to show that $U_1, \ldots, U_k$ in fact is a partition of U and that for every $1 \leq i \leq k$ the overall size $\Sigma_{u \in U_i} s(u)$ does not exceed b.

As $w \in C^{P\mathcal{I}}$, every C^u must have a witness in the set X. Thus, the union over all subsets U_i yields U. The restriction $u \notin U_j$ in the definition of every U_i ensures that for every $u \in U$ at most one index i exists with $u \in U_i$. Hence, $U_1, \ldots, U_k$ is a partition of U.

Let $i \in \{1, \ldots, k\}$. By definition, U_i contains a subset of all $u \in U$ of which x_i is a witness. If U_i is nonempty, then two facts are implied. Firstly, x_i has at most b successors w.r.t. r because of the number restriction in one C_u. Secondly, x_i has at least $\Sigma_{u \in U_i} s(u)$ successors w.r.t. r. This holds due to the existential restrictions of the form $\exists r.f^P(u, j)$ in every C^u with $u \in U_i$: for every $u \in U_i$ and for every $j \in \{1, \ldots, s(u)\}$, denote by y_{uj} the r-successor of x_i implied by $\exists r.f^P(u, j)$. Assume that $y_{uj} = y_{u'j'}$ for some $u, v \in U_i$, $j \in \{1, \ldots, s(u)\}$, and $j' \in \{1, \ldots, s(u)\}$. Then, y_{uj} is a witness of $f^P(u, j) \sqcap f^P(u', j')$, in contradiction to Claim (1) of the proof. □

As satisfiability of $\mathcal{ELN}$ concepts can be reduced to subsumption, i.e., a concept description C is satisfiable if and only if $C \not\sqsubseteq \bot \equiv (\leqslant 0\ r) \sqcap (\geqslant 1\ r)$, we immediately obtain the following completeness results:

Theorem 3.1.8 *Deciding satisfiability in $\mathcal{ELN}$ w.r.t. the empty TBox is NP-complete in the strong sense. Deciding subsumption in $\mathcal{ELN}$ w.r.t. the empty TBox is co-NP-complete in the strong sense.*

3.1.2 Subsumption w.r.t. acyclic TBoxes

We show co-NP-hardness of subsumption w.r.t. acyclic $\mathcal{EL}_{\forall\exists}$-TBoxes, i.e., of acyclic TBoxes over $\mathcal{EL}$ extended by the constructor allsome, see Definitions 2.1.1 and 2.1.3. The proof is by reduction of the subsumption problem in $\mathcal{FL}_0$ w.r.t. acyclic TBoxes.

Denote by $\mathcal{L}_{\forall\exists}$ the sublanguage of $\mathcal{EL}_{\forall\exists}$ in which the usual existential restriction ($\exists r.C$) is missing. In order to establish the lower bound, we translate acyclic $\mathcal{FL}_0$-TBoxes into subsumption-preserving equivalent ones over $\mathcal{L}_{\forall\exists}$, thereby reducing the subsumption problem. To this end, we introduce a normal form for $\mathcal{FL}_0$-TBoxes that simplifies the translation.

Definition 3.1.9 (Translation function)
Let $\mathcal{T}$ be an arbitrary $\mathcal{FL}_0$-TBox over N_{con} and N_{role}. $\mathcal{T}$ is called *reduced* iff none of the

following transformation rules can be applied to any concept description C with $A \equiv C \in \mathcal{T}$ or any of its subdescriptions:

$$\begin{aligned} \forall r.\top &\longrightarrow \top \\ E &\longrightarrow \top \quad \text{iff } E \equiv \top \in \mathcal{T} \\ F \sqcap \top &\longrightarrow F, \end{aligned}$$

where $r \in \mathsf{N}_{\text{role}}^{\mathcal{T}}$, E is an arbitrary defined concept, and F an arbitrary concept description over $\mathsf{N}_{\text{con}}^{\mathcal{T}}$ or $\top$. For a reduced TBox $\mathcal{T}$, the translated TBox $\mathsf{trans}(\mathcal{T})$ is defined by syntactically replacing all $\forall$-quantors by $\forall\exists$-quantors: $\mathsf{trans}(\mathcal{T}) := \mathcal{T}\{\forall/\forall\exists\}$. □

The above definition is meant to be correct only in the sense that all subsumption relations are preserved. While a model of $\mathsf{trans}(\mathcal{T})$ can always be shown to be a model of $\mathcal{T}$, the reverse direction need *not* hold.

To prove correctness of the translation, we first devise a formal-language characterization of subsumption for $\mathcal{L}_{\forall\exists}$-concept descriptions. We restrict ourselves to subsumption w.r.t. the empty TBox since acyclic TBoxes can be expanded until no defined concepts occur on right-hand sides of concept definitions. In $\mathcal{FL}_0$, the equivalence $\forall r.(C \sqcap D) \equiv \forall r.C \sqcap \forall r.D$ gives rise to a particularly simple representation of concept descriptions, called *unfolding* in [Neb90] or *concept centered normal form* in [BN98]. Given a concept description C, the idea is to exploit the above equivalence from left to right until conjunction in C occurs only on toplevel, implying that all value restrictions are of the form $\forall r_1.\forall r_2.\cdots\forall r_n.A$ with $A \in \mathsf{N}_{\text{prim}}$. The word $r_1 r_2 \ldots r_n$ can then be used to represent the corresponding restriction C imposes w.r.t. A.

The same principle holds for $\mathcal{L}_{\forall\exists}$: a concept description $\forall\exists r.(C \sqcap D)$ by definition equals $\forall r.(C \sqcap D) \sqcap \exists r.(C \sqcap D)$. Because of the propagation from value to existential restrictions, replacing $\exists r.(C \sqcap D)$ by $\exists r.\top$ preserves equivalence. Duplicating $\exists r.\top$, the propagation argument in the reverse direction yields $\forall\exists r.C \sqcap \forall\exists r.D$.

We will use the above normalization to define so-called *role languages* for atomic concepts occurring in concept descriptions. We start by extending the constructors $\forall$ and $\forall\exists$ to words over N_{role}. In the remainder of this section, we may w.l.o.g. assume that N_{con} and N_{role} are not only finite but limited by the concept names and role names occurring in the concept descriptions C, D for which we want to decide $C \sqsubseteq D$.

Definition 3.1.10 (Word restrictions)
For all $A \in \mathsf{N}_{\text{prim}}$, $r \in \mathsf{N}_{\text{role}}$, $w \in \mathsf{N}_{\text{role}}^*$, and for $Q \in \{\forall, \forall\exists\}$, the concept description $Qw.A$ is inductively defined by:

$$\begin{aligned} Q\varepsilon.A &:= A \\ Qrw.A &:= Qr.Qw.A \end{aligned}$$ □

As we need to refer to the (already existing) role-language characterization for $\mathcal{FL}_0$, we simultaneously introduce role languages for $\mathcal{FL}_0$-concept descriptions and $\mathcal{L}_{\forall\exists}$-concept descriptions. Obviously, while $\top$ can be ignored for $\mathcal{FL}_0$, it must be treated as an ordinary concept name in $\mathcal{L}_{\forall\exists}$.

Definition 3.1.11 (Role languages)
Let C be an $\mathcal{FL}_0$-concept description. Then, for $Q = \forall$ and for arbitrary $A, B \in \mathsf{N}_{\text{prim}}$ the

formal language $L_A(C)$ is inductively defined by:

$$\begin{aligned} L_A(\top) &:= \emptyset \\ L_A(B) &:= \{\varepsilon \mid A = B\} \\ L_A(\textstyle\bigsqcap_i C_i) &:= \bigcup_i L_A(C_i) \\ L_A(Qr.C) &:= \{r\} \cdot L_A(C) \end{aligned}$$

For $\mathcal{L}_{\forall\exists}$-concept descriptions ($Q = \forall\exists$) the top-concept $\top$ is treated like a primitive concept. Hence, the inductive definition is extended to arbitrary $A, B \in \mathsf{N}_{\text{prim}} \cup \{\top\}$ and the definition $L_A(\top) := \emptyset$ is removed. To simplify notation, denote $L_A(C)$ by $C|_A$. □

The language $C|_A$ contains all words $r_1 \dots r_n$ over N_{role} with $C \sqsubseteq Qr_1.\cdots Qr_n.A$, where $Q = \forall$ in case of $\mathcal{FL}_0$ and $Q = \forall\exists$ in case of $\mathcal{L}_{\forall\exists}$. In [Neb90] it has been shown that the set of all role languages of a given $\mathcal{FL}_0$-concept description in fact *characterizes* the concept up to equivalence. The following lemma holds:

Lemma 3.1.12 *Let C be an $\mathcal{FL}_0$-concept description over N_{prim} and N_{role}. Then, $C \equiv \bigsqcap_{A \in \mathsf{N}_{\text{prim}}} \bigsqcap_{w \in C|_A} \forall w.A$*

In [BKBM99], subsumption of $\mathcal{FL}_\bot$ concept descriptions $C \sqsubseteq D$ has been characterized by the role languages of C and D. For the sublanguage $\mathcal{FL}_0$, we can immediately derive the following characterization of subsumption of $\mathcal{FL}_0$-concept descriptions.

Lemma 3.1.13 *Let C, D be $\mathcal{FL}_0$-concept descriptions. Then, $C \sqsubseteq D$ iff $C|_A \supseteq D|_A$ for all $A \in \mathsf{N}_{\text{con}}$.*

We aim at a similar characterization of subsumption for $\mathcal{L}_{\forall\exists}$. For arbitrary $\mathcal{L}_{\forall\exists}$-concept descriptions C, D and every $r \in \mathsf{N}_{\text{role}}$ it holds that $\forall\exists r.(C \sqcap D) \equiv \forall r.(C \sqcap D) \sqcap \exists r.\top$, which in turn is equivalent to $\forall r.C \sqcap \forall r.D \sqcap \exists r.\top \sqcap \exists r.\top$, which simplifies to $\forall\exists r.C \sqcap \forall\exists r.D$. This immediately yields the extension of Lemma 3.1.12 to $\mathcal{L}_{\forall\exists}$-concept descriptions.

Lemma 3.1.14 *Let C be an $\mathcal{L}_{\forall\exists}$-concept description. Then, $C \equiv \bigsqcap_{A \in \mathsf{N}_{\text{prim}}} \bigsqcap_{w \in C|_A} \forall\exists w.A \sqcap \bigsqcap_{w \in C|_\top} \forall\exists w.\top$.*

Note that $\forall\exists r.\top \not\equiv \top$ as in the case of $\mathcal{FL}_0$. The following lemma gives a role-language characterization of subsumption of $\mathcal{L}_{\forall\exists}$-concept descriptions w.r.t. the empty TBox. The interesting part is the treatment of the $\top$-concept for which an additional equation is introduced.

Lemma 3.1.15 *Let C, D be $\mathcal{L}_{\forall\exists}$-concept descriptions. Then, $C \sqsubseteq D$ iff*

1. *$C|_P \supseteq D|_P$ for all $P \in \mathsf{N}_{\text{prim}}$; and*
2. *$C|_\top \cup \bigcup_{P \in \mathsf{N}_{\text{prim}}} C|_P \cup \{\varepsilon\} \supseteq D|_\top$.*

PROOF. ($\Rightarrow$) Proof by contraposition. If the first condition is violated then there exists an atomic concept $P \in \mathsf{N}_{\text{prim}}$ with $C|_P \not\supseteq D|_P$, implying $w \in D|_P \setminus C|_P$ for some word w. We construct a model $\mathcal{I}$ of C that is no model of D. Let $\Delta^\mathcal{I} := \{x_0, \dots, x_{|w|+1}\}$. Let $P^\mathcal{I} := \Delta^\mathcal{I} \setminus \{x_{|w|}\}$. For all $Q \in \mathsf{N}_{\text{prim}} \setminus \{P\}$, let $Q^\mathcal{I} := \Delta^\mathcal{I}$. For all $r \in \mathsf{N}_{\text{role}}$, define

$$r^\mathcal{I} := \{(x_i, x_{i+1}) \mid 0 \le i \le |w|\} \cup \{(x_{|w|+1}, x_{|w|+1})\}.$$

Thus, every vertex x_i has a successor w.r.t. every role $r \in \mathsf{N}_{\text{role}}$. Every vertex except $x_{|w|}$ is a witness of all atomic concepts $Q \in \mathsf{N}_{\text{prim}}$. Only $x_{|w|}$ is a witness of all atomic concepts except P. Since $w \notin C|_P$, x_0 is a witness of C but no witness of D, where a w-chain of successors must lead to a witness of P. Hence, $\mathcal{I}$ contradicts $C \sqsubseteq D$.

If the second condition is violated then there is a word $w \in D|_\top \setminus \{\varepsilon\}$ with $w \notin C|_\top$ and $w \notin C|_P$ for every $P \in \mathsf{N}_{\text{prim}}$. Since $|w| \geq 1$, denote w as vs with $v \in \mathsf{N}_{\text{role}}^*$ and $s \in \mathsf{N}_{\text{role}}$. Let $\Delta^{\mathcal{J}} := \{x_0, \ldots, x_{|w|}\}$. For all $P \in \mathsf{N}_{\text{prim}}$, let $P^{\mathcal{J}} := \Delta^{\mathcal{J}}$, i.e., all atomic concepts hold in every vertex of $\mathcal{J}$. For all $r \in \mathsf{N}_{\text{role}}$, define

$$r^{\mathcal{J}} := \{(x_i, x_{i+1}) \mid 0 \leq i \leq |v| - 1\} \cup \{(x_{|v|}, x_{|v|+1}) \mid r \neq s\} \cup \{(x_{|v|+1}, x_{|v|+1})\}.$$

In $\mathcal{J}$, every vertex x_i except $x_{|v|}$ has a successor w.r.t. every role $r \in \mathsf{N}_{\text{role}}$ while $x_{|v|}$ has a successor w.r.t. every role except s. Hence, x_0 is a witness of C but none of D where an s-successor must be present after traveling a v-path. This contradicts $C \sqsubseteq D$.

($\Leftarrow$) Then C is equivalent to

$$\bigsqcap_{P \in \mathsf{N}_{\text{prim}}} \bigsqcap_{w \in C|_P} \forall\exists w.P \sqcap \bigsqcap_{w \in C|_\top} \forall\exists w.\top$$

and analogously for D. Using the subset relations from Condition 1, we write C as

$$\bigsqcap_{P \in \mathsf{N}_{\text{prim}}} \bigsqcap_{w \in D|_P} \forall\exists w.P \sqcap \bigsqcap_{P \in \mathsf{N}_{\text{prim}}} \bigsqcap_{w \in C|_P} \forall\exists w.P \sqcap \bigsqcap_{w \in C|_\top} \forall\exists w.\top.$$

Since $\forall\exists w.P \sqsubset \forall\exists w.\top$, we may (i) add subdescriptions $\forall\exists w.\top$ for which w also occurs in a subdescription referring to some $P \in \mathsf{N}_{\text{prim}}$; and (ii) add $\top$, obtaining

$$\begin{aligned} C \equiv \bigsqcap_{P \in \mathsf{N}_{\text{prim}}} \bigsqcap_{w \in D|_P} \forall\exists w.P \sqcap \bigsqcap_{P \in \mathsf{N}_{\text{prim}}} \bigsqcap_{w \in C|_P} \forall\exists w.P \\ \sqcap \bigsqcap_{w \in C|_\top} \forall\exists w.\top \sqcap \bigsqcap_{w \in \bigcup_{P \in \mathsf{N}_{\text{prim}} \cup \{\varepsilon\}} C|_P} \forall\exists w.\top. \end{aligned}$$

Exploiting the subset relation in Condition 2, C can be rewritten further to

$$\begin{aligned} \bigsqcap_{P \in \mathsf{N}_{\text{prim}}} \bigsqcap_{w \in D|_P} \forall\exists w.P \sqcap \bigsqcap_{P \in \mathsf{N}_{\text{prim}}} \bigsqcap_{w \in C|_P} \forall\exists w.P \\ \sqcap \bigsqcap_{w \in D|_\top} \forall\exists w.\top \sqcap \bigsqcap_{w \in C|_\top} \forall\exists w.\top \sqcap \bigsqcap_{w \in \bigcup_{P \in \mathsf{N}_{\text{prim}} \cup \{\varepsilon\}} C|_P} \forall\exists w.\top \end{aligned}$$

which equals

$$\begin{aligned} D \sqcap \bigsqcap_{P \in \mathsf{N}_{\text{prim}}} \bigsqcap_{w \in C|_P} \forall\exists w.P \\ \sqcap \bigsqcap_{w \in C|_\top} \forall\exists w.\top \sqcap \bigsqcap_{w \in \bigcup_{P \in \mathsf{N}_{\text{prim}} \cup \{\varepsilon\}} C|_P} \forall\exists w.\top. \end{aligned}$$

Hence, C is equivalent to or more specific than D, i.e., $C \sqsubseteq D$. □

The above characterization of subsumption allows a straightforward proof of correctness of the translation from $\mathcal{FL}_0$ to $\mathcal{L}_{\forall\exists}$.

Lemma 3.1.16 *Let $\mathcal{T}$ be an acyclic reduced $\mathcal{FL}_0$-TBox. Let $A, B \in \mathsf{N}_{\text{def}}$. Then, $A \sqsubseteq_{\mathcal{T}} B$ iff $A \sqsubseteq_{\mathsf{trans}(\mathcal{T})} B$.*

PROOF. Let $A \equiv C, B \equiv D \in \mathcal{T}$. Denote by $\tilde{\mathcal{T}}$ the expansion of $\mathcal{T}$ in which defined concepts occur only on left-hand sides of definitions. Let $A \equiv \tilde{C}_0, B \equiv \tilde{D}_0 \in \tilde{\mathcal{T}}$. Analogously,

let $A \equiv \tilde{C}_{\forall\exists}, B \equiv \tilde{D}_{\forall\exists} \in \widetilde{\mathsf{trans}}(\mathcal{T})$. It suffices to show that $\tilde{C}_0 \sqsubseteq \tilde{D}_0$ iff $\tilde{C}_{\forall\exists} \sqsubseteq \tilde{D}_{\forall\exists}$. As $\mathcal{T}$ is reduced, the translated TBox $\mathsf{trans}(\mathcal{T})$ differs from $\mathcal{T}$ only in the quantors occurring on the right-hand side of concept definitions. This implies $\tilde{C}_0|_P = \tilde{C}_{\forall\exists}|_P$ and $\tilde{D}_0|_P = \tilde{D}_{\forall\exists}|_P$ for all $P \in \mathsf{N}_{\text{prim}}$.

($\Leftarrow$) The subsumption $\tilde{C}_{\forall\exists} \sqsubseteq \tilde{D}_{\forall\exists}$ especially implies $\tilde{C}_{\forall\exists}|_P \supseteq \tilde{D}_{\forall\exists}|_P$ for all $P \in \mathsf{N}_{\text{prim}}$. As argued above, the equality of the role languages of $\tilde{C}_{\forall\exists}$ and $\tilde{C}_0$, and $\tilde{D}_{\forall\exists}$ and $\tilde{D}_0$, respectively, immediately yields $\tilde{C}_0|_P \supseteq \tilde{D}_0|_P$ for all $P \in \mathsf{N}_{\text{prim}}$. By the characterization of subsumption for $\mathcal{FL}_0$, $\tilde{C}_0 \sqsubseteq \tilde{D}_0$.

($\Rightarrow$) Then, by characterization of subsumption, $\tilde{C}_{\forall\exists}|_A \supseteq \tilde{D}_{\forall\exists}|_A$ for all $A \in \mathsf{N}_{\text{prim}}$. Hence, it suffices to show

$$\tilde{C}_{\forall\exists}|_\top \cup \bigcup_{A \in \mathsf{N}_{\text{prim}}} \tilde{C}_{\forall\exists}|_A \cup \{\varepsilon\} \supseteq \tilde{D}_{\forall\exists}|_\top.$$

To this end, we show that either $\tilde{D}_0|_\top = \emptyset$ or $\tilde{D}_0 = \top$ implying $\tilde{D}_0|_\top = \{\varepsilon\}$. Note that $\tilde{D}_0|_\top = \tilde{D}_{\forall\exists}|_\top$. Proof by induction on the cardinality $|\mathcal{T}|$ of $\mathcal{T}$.

- $|\mathcal{T}| = 1$
  Then $\mathcal{T} = \tilde{\mathcal{T}} = \{B \equiv D\}$ and $D = \tilde{D}_0$. As $\mathcal{T}$ is reduced, $\tilde{D}_0 = \top$ or $\tilde{D}_0$ is a conjunction of atomic concepts, implying $\tilde{D}_0|_\top = \{\varepsilon\}$ or $\tilde{D}_0|_\top = \emptyset$.

- $|\mathcal{T}| > 1$
  As $\mathcal{T}$ is acyclic and reduced, there is a concept definition $E \equiv F \in \mathcal{T}$ such that either $F = \top$ or F is a conjunction of atomic concepts. If $F \neq \top$ then replacing E by F on the right-hand side of any definition in $\mathcal{T}$ has no effect on $\tilde{D}_0|_\top$. Otherwise, every occurrence of E on any right-hand side has already been replaced by $\top$ by the second reduction rule. In both cases, we obtain the correct set $\tilde{D}_0|_\top$ by regarding E as an atomic concept and computing $\tilde{D}_0|_\top$ w.r.t. $\mathcal{T} \setminus \{E \equiv F\}$ for which the claim holds by IH. □

It has been shown in [Neb90] that the subsumption problem in $\mathcal{FL}_0$ w.r.t acyclic TBoxes (containing only definitions) is co-NP-complete. Hence, due to the above lemma, we obtain co-NP-hardness of the subsumption problem w.r.t. acyclic $\mathcal{L}_{\forall\exists}$-TBoxes.

In order to show co-NP-completeness of the subsumption problem w.r.t. acyclic $\mathcal{L}_{\forall\exists}$-TBoxes, we adapt the characterization of subsumption from Definition 3.1.15 to acyclic $\mathcal{L}_{\forall\exists}$-TBoxes. To this end, we represent the relevant role languages by nondeterministic finite automata.[1] W.l.o.g., we restrict ourselves to *normalized* acyclic $\mathcal{L}_{\forall\exists}$-TBoxes where every Definition is of the form

$$A \equiv \prod_{i=1}^{n} P_i \sqcap \prod_{j=1}^{m} \exists r_j.B_j,$$

where all P_i are atomic concepts and all B_j are defined concepts.

Definition 3.1.17 Let $\mathcal{T}$ be a normalized acyclic $\mathcal{L}_{\forall\exists}$-TBox, $A_0 \in \mathsf{N}_{\text{def}}^{\mathcal{T}}$, and $P \in \mathsf{N}_{\text{prim}}^{\mathcal{T}}$. Then the nondeterministic finite automaton $\mathfrak{A}_{\mathcal{T}}(A_0, P)$ is defined by $(\mathsf{N}_{\text{role}}, \mathsf{N}_{\text{def}}^{\mathcal{T}}, \delta, A_0, F)$, where N_{role} is the alphabet, $\mathsf{N}_{\text{def}}^{\mathcal{T}}$ the set of states, δ the transition function, A_0 the initial state and F the set of final states. The transition function δ is defined by

$$\begin{aligned} \delta\colon \mathsf{N}_{\text{def}}^{\mathcal{T}} \times \mathsf{N}_{\text{role}} &\longrightarrow \wp(\mathsf{N}_{\text{def}}^{\mathcal{T}}) \\ (A, r) &\longmapsto \{B \in \mathsf{N}_{\text{def}}^{\mathcal{T}} \mid \exists r.B \in \mathsf{def}_{\mathcal{T}}(A)\} \end{aligned}$$

and the set of final states is $F := \{B \in \mathsf{N}_{\text{def}}^{\mathcal{T}} \mid P \in \mathsf{def}_{\mathcal{T}}(B)\}$. □

[1] Note that our approach is very similar to the one for acyclic $\mathcal{FL}_0$-TBoxes shown in [Neb90].

Note that $\mathfrak{A}_\mathcal{T}(A_0, P)$ is acyclic for acyclic TBoxes $\mathcal{T}$. The next lemma shows that the above automata can be used to characterize subsumption of defined concepts w.r.t. $\mathcal{L}_{\forall\exists}$-TBoxes without expanding them first.

Lemma 3.1.18 *Let $\mathcal{T}$ be an acyclic normalized $\mathcal{L}_{\forall\exists}$-TBox and $A \equiv C \in \mathcal{T}$. Let $\tilde{\mathcal{T}}$ be the expansion of $\mathcal{T}$. Then, $\mathsf{deft}_{\tilde{\mathcal{T}}}(A)|_P = L(\mathfrak{A}_\mathcal{T}(A, P))$ for every $P \in \mathsf{N}^\mathcal{T}_{\text{prim}} \cup \{\top\}$.*

Proof. Let $P \in \mathsf{N}^\mathcal{T}_{\text{prim}} \cup \{\top\}$. Proof by induction on the cardinality $|\mathcal{T}|$ of $\mathcal{T}$.

- $|\mathcal{T}| = 1$
  Then $\mathcal{T} = \tilde{\mathcal{T}} = \{A \equiv C\}$, where C is a conjunction of atomic concepts, implying $\mathsf{deft}_{\tilde{\mathcal{T}}}(A)|_P = \{\varepsilon \mid P \in \mathsf{def}_\mathcal{T}(A)\}$. By definition, A is the only state in $\mathfrak{A}_\mathcal{T}(A, P)$, A is a finite state iff $P \in \mathsf{def}_\mathcal{T}(A)$, and $\mathfrak{A}_\mathcal{T}$ is acyclic. This implies $L(\mathfrak{A}_\mathcal{T}(A, P)) = \{\varepsilon \mid P \in \mathsf{def}_\mathcal{T}(A)\}$, as required.
- $|\mathcal{T}| > 1$
  If $\mathsf{def}_\mathcal{T}(A) \subseteq \mathsf{N}_{\text{prim}}$ then the proof is analogous to the case $|\mathcal{T}| = 1$. Otherwise,

  $$\mathsf{def}_{\tilde{\mathcal{T}}}(A) = \bigsqcap_{P \in \mathsf{def}_\mathcal{T}(A) \cap \mathsf{N}_{\text{prim}}} P \sqcap \bigsqcap_{\exists r.B \in \mathsf{def}_\mathcal{T}(A)} \exists r.B,$$

  Consequently,

  $$\mathsf{deft}_{\tilde{\mathcal{T}}}(A)|_P = \{\varepsilon \mid P \in \mathsf{def}_{\tilde{\mathcal{T}}}(A)\} \cup \bigcup_{\exists r.B \in \mathsf{def}_\mathcal{T}(A)} \{r\} \cdot \mathsf{deft}_{\tilde{\mathcal{T}}}(B)|_P,$$

  and since $\mathfrak{A}_\mathcal{T}(A, P)$ is acyclic,

  $$L(\mathfrak{A}_\mathcal{T}(A, P)) = \{\varepsilon \mid P \in \mathsf{def}_\mathcal{T}(A)\} \cup \bigcup_{\exists r.B \in \mathsf{def}_\mathcal{T}(A)} \{r\} \cdot L(\mathfrak{A}_\mathcal{T}(B, P)).$$

  By IH, $\mathsf{deft}_{(\mathcal{T} \setminus \{A \equiv C\})^\top}(B)|_P = L(\mathfrak{A}_{\mathcal{T} \setminus \{A \equiv C\}}(B, P))$ for every B occurring in the above conjunction. Since $\mathcal{T}$ is acyclic, the definition of every such B does not depend directly or indirectly on A. This implies, firstly, $\mathsf{deft}_{(\mathcal{T} \setminus \{A \equiv C\})^\top}(B)|_P = \mathsf{deft}_{\tilde{\mathcal{T}}}(B)|_P$; and secondly, $L(\mathfrak{A}_{\mathcal{T} \setminus \{A \equiv C\}}(B, P)) = L(\mathfrak{A}_\mathcal{T}(B, P))$ because A is unreachable from B in $\mathfrak{A}_\mathcal{T}(B, P)$. □

As $A \sqsubseteq_\mathcal{T} B$ iff $\mathsf{deft}_{\tilde{\mathcal{T}}}(A) \sqsubseteq \mathsf{deft}_{\tilde{\mathcal{T}}}(B)$, Lemma 3.1.15 and Lemma 3.1.18 immediately imply the following characterization of subsumption w.r.t. acyclic $\mathcal{L}_{\forall\exists}$-TBoxes.

Corollary 3.1.19 *Let $\mathcal{T}$ be an acyclic normalized $\mathcal{L}_{\forall\exists}$-TBox and $A, B \in \mathsf{N}^\mathcal{T}_{\text{def}}$. Then, $A \sqsubseteq_\mathcal{T} B$ iff*

1. *$L(\mathfrak{A}_\mathcal{T}(A, P)) \supseteq L(\mathfrak{A}_\mathcal{T}(B, P))$ for all $P \in \mathsf{N}^\mathcal{T}_{\text{prim}}$; and*
2. *$L(\mathfrak{A}_\mathcal{T}(A, \top)) \cup \bigcup_{P \in \mathsf{N}^\mathcal{T}_{\text{prim}}} L(\mathfrak{A}_\mathcal{T}(A, P)) \cup \{\varepsilon\} \supseteq L(\mathfrak{A}_\mathcal{T}(A, \top))$.*

Note that all relevant automata are of linear size in $|\mathcal{T}|$. Especially, the union on the left-hand side of the second condition can easily be represented by a single acyclic NFA of linear size in $|\mathcal{T}|$. To this end, introduce an additional automaton to represent $\{\varepsilon\}$ and construct the union-automaton in the usual way (see, e.g., [MY60]). As the inclusion problem for acyclic NFA is co-NP-complete [GJ79, p. 265], subsumption w.r.t. acyclic $\mathcal{L}_{\forall\exists}$-TBoxes is in co-NP, so that we obtain tight complexity bounds for $\mathcal{L}_{\forall\exists}$. Altogether, we yield the following results.

Theorem 3.1.20 *Deciding subsumption in $\mathcal{L}_{\forall\exists}$ w.r.t. acyclic TBoxes is co-NP-complete. Deciding subsumption in $\mathcal{EL}_{\forall\exists}$ w.r.t. acyclic TBoxes is co-NP-hard.*

We shall come back to the approach of characterizing subsumption by equations over formal languages in Section 4.2.1, where the DLs $\mathcal{FL}_\bot$, $\mathcal{FL}_\neg$, and $\mathcal{ALN}$ are studied.

3.1.3 Subsumption w.r.t. general TBoxes

For general TBoxes we show that tractability of the subsumption problem is lost when extending $\mathcal{EL}$ by one of the constructs value restriction, role value maps, atomic negation, disjunction, inverse roles, functional roles, at-most-one restrictions, at-least-two restrictions, non-p-admissible concrete domains, and one of the role constructors negation, union, and transitive closure. But for two exceptions, the subsumption problem in the extended languages does not only become intractable but leaps to EXPTIME-completeness. In one case, reasoning does even become undecidable. In the second case, we can currently only prove a PSPACE lower bound that does not match EXPTIME, the best known upper bound. We start with two results from the literature, EXPTIME-completeness for value restrictions and undecidability for role value maps.

Throughout this section, we will often have to refer to general TBoxes. In order to simplify our notation, general TBoxes will more concisely be called *IBoxes* ('I' stands for 'inclusion').

Value restrictions

Let $\mathcal{EL}_\forall$ be the extension of $\mathcal{EL}$ by the constructor $\forall$ introduced in Definitions 2.1.1 and 2.1.3. The following has been shown in [GMWK02].

Theorem 3.1.21 *(Givan et al.)*
Subsumption in $\mathcal{EL}_\forall$ w.r.t. IBoxes is EXPTIME-complete.

In Section 3.2.4, we improve upon this result by showing that subsumption w.r.t. IBoxes is already EXPTIME-complete in the logic $\mathcal{FL}_0$ providing only conjunction and value restriction.

Role value maps

Consider the extension of $\mathcal{EL}$ by *role value maps* which generalize RIs introduced in Definition 2.1.4. A *role value map* is of the form

$$r_1 \circ \cdots \circ r_k \sqsubseteq s_1 \circ \cdots \circ s_\ell$$

with $r_1, \ldots, r_k$ and $s_1, \ldots, s_\ell$ role names. Analogously to RIs, an interpretation $\mathcal{I}$ *satisfies* a role value map $r_1 \circ \cdots \circ r_k \sqsubseteq s_1 \circ \cdots \circ s_\ell$ iff $r_1^\mathcal{I} \circ \cdots \circ r_k^\mathcal{I} \subseteq s_1^\mathcal{I} \circ \cdots \circ s_\ell^\mathcal{I}$m, where $\circ$ is interpreted as composition of binary relations. The following has been proved in [Baa03b].

Theorem 3.1.22 *(Baader)*
Subsumption of $\mathcal{EL}$-concepts w.r.t. finite sets of role value maps is undecidable.

In the following, we discuss several other extensions of $\mathcal{EL}$, and show for each that tractability of subsumption w.r.t. IBoxes is not preserved.

Atomic Negation

Let $\mathcal{EL}_\neg$ be the extension of $\mathcal{EL}$ by the negation constructor introduced in Definitions 2.1.1 and 2.1.3. Analogously, let $\mathcal{EL}_{(\neg)}$ be the extension of $\mathcal{EL}$ by *atomic* negation.

The DL $\mathcal{ALC}$ is a notational variant of $\mathcal{EL}_\neg$ because every disjunction $C \sqcup D$ occurring in some GCI of an $\mathcal{ALC}$-IBox can be replaced applying the DeMorgan rules, i.e., $\neg(\neg C \sqcap \neg D)$. As satisfiability and subsumption in $\mathcal{ALC}$ w.r.t. IBoxes are EXPTIME-complete [Sch91], the same complexity bound is obtained for $\mathcal{EL}_\neg$.

Moreover, EXPTIME-completeness carries over to $\mathcal{EL}_{(\neg)}$ since full negation can be expressed using atomic negation and IBoxes: if some GCI in an $\mathcal{EL}_\neg$-IBox contains a complex negated concept $\neg C$ then this can be eliminated in two steps. Firstly, replace $\neg C$ with $\neg A$ with A a fresh concept name; and secondly, introduce two new GCIs $A \sqsubseteq C$ and $C \sqsubseteq A$. We thus obtain the following.

Theorem 3.1.23 *In $\mathcal{EL}_{(\neg)}$, satisfiability and subsumption w.r.t. IBoxes is EXPTIME-complete.*

Disjunction

Let $\mathcal{ELU}$ be the extension of $\mathcal{EL}$ with a disjunction constructor introduced in Definitions 2.1.1 and 2.1.3. We show that subsumption in $\mathcal{ELU}$ w.r.t. IBoxes is EXPTIME-complete.

The upper bound is simple since $\mathcal{ELU}$ is a fragment of $\mathcal{ALC}$ for which subsumption w.r.t. IBoxes is in EXPTIME. For the lower bound, we reduce satisfiability of $\mathcal{EL}_{(\neg)}$-concepts w.r.t. IBoxes to subsumption of $\mathcal{ELU}$-concepts. Note that the former is EXPTIME-hard by Theorem 3.1.23. Let C_0 be an $\mathcal{EL}_{(\neg)}$-concept and $\mathcal{C}$ an $\mathcal{EL}_{(\neg)}$-IBox. Satisfiability of C_0 w.r.t. $\mathcal{C}$ is to be decided. W.l.o.g., assume that C_0 is a concept name. Otherwise, decide satisfiability of a fresh concept name A w.r.t. $\mathcal{C} \cup \{A \sqsubseteq C_0\}$. For every concept name A occurring in $\mathcal{C}$, fix a fresh concept name A' (i.e., distinct from C_0 and not occurring in $\mathcal{C}$). Also fix an additional fresh concept name L. Then the $\mathcal{ELU}$-IBox $\mathcal{C}^*$ is obtained from $\mathcal{C}$ by first replacing every subconcept $\neg A$ with A', and then adding the following GCIs:

- $\top \sqsubseteq A \sqcup A'$ and $A \sqcap A' \sqsubseteq L$ for each concept name A occurring in $\mathcal{C}$;
- $\exists r.L \sqsubseteq L$.

Note that the GCI $\exists r.L \sqsubseteq L$ is equivalent to $\neg L \sqsubseteq \forall r.\neg L$. This lets L act like the bottom concept in (connected) countermodels of the subsumption $C_0 \sqsubseteq_{\mathcal{C}^*} L$. It is easy to see that C is satisfiable w.r.t. $\mathcal{C}$ iff $C_0 \not\sqsubseteq_{\mathcal{C}^*} L$.

Theorem 3.1.24 *In $\mathcal{ELU}$, subsumption w.r.t. IBoxes is EXPTIME-complete.*

This theorem improves upon the result that subsumption of $\mathcal{ELU}$ concepts w.r.t. IBoxes is co-NP-hard [Bra04b], see also Theorem 3.1.4, and it improves upon the result of Hladik and Sattler that satisfiability of $\mathcal{ELU}$ concepts extended with functional roles and the bottom concept w.r.t. IBoxes is EXPTIME-hard [HS03]. Note that satisfiability in $\mathcal{ELU}$ w.r.t. IBoxes is trivial since every concept is satisfiable w.r.t. every IBox.

Inverse Roles

Let $\mathcal{ELI}$ be the extension of $\mathcal{EL}$ by the concept constructor $\exists r^-.C$ with the following semantics:

$$(\exists r^-.C)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid \exists y\colon (y, x) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}.$$

We show that subsumption in $\mathcal{ELI}$ w.r.t. IBoxes is PSPACE-hard. Whether or not this lower bound is tight remains open. The result is established by a reduction of the satisfiability problem in the DL $\mathcal{ALE}$ w.r.t. so-called primitive TBoxes:

- $\mathcal{ALE}$ is obtained by extending $\mathcal{EL}_\forall$ with atomic negation;
- *primitive TBoxes* are IBoxes whose GCIs are of the form $A \sqsubseteq C$ with A a concept name.

It has been shown by Calvanese that satisfiability in $\mathcal{ALE}$ w.r.t. primitive TBoxes is PSPACE-complete [Cal96].

Let C_0 be an $\mathcal{ALE}$ concept, and $\mathcal{T}$ a primitive $\mathcal{ALE}$-TBox. Satisfiability of C_0 w.r.t. $\mathcal{T}$ is to be decided. As usual, assume that C_0 is a concept name. We also assume that $\mathcal{T}$ is in normal form, i.e., every GCI is of one of the following forms:

$$A \sqsubseteq B$$
$$A \sqsubseteq \neg B$$
$$A \sqsubseteq B \sqcap B'$$
$$A \sqsubseteq \exists r.B$$
$$A \sqsubseteq \forall r.B,$$

where A, B, and B' are concept names. Note that every primitive TBox can be converted into normal form by normalization rules similar to the ones in Figure 3.2.1. For the reduction, we reserve a fresh concept name L and define an $\mathcal{EL}$-IBox $\mathcal{C}$ with the following GCIs:

- $A \sqsubseteq D$ for all $A \sqsubseteq D \in \mathcal{T}$ if D is a concept name or of the form $\exists r.B$;
- $\exists r^-.A \sqsubseteq B$ for all $A \sqsubseteq \forall r.B \in \mathcal{T}$;
- $A \sqcap B \sqsubseteq L$ for all $A \sqsubseteq \neg B \in \mathcal{T}$; and
- $\exists r.L \sqsubseteq L$.

As in the case of $\mathcal{ELU}$, the GCI $\exists r.L \sqsubseteq L$ is equivalent to $\neg L \sqsubseteq \forall r.\neg L$ and lets L act like the bottom concept in (connected) countermodels of the subsumption $C_0 \sqsubseteq_\mathcal{C} L$. Additionally, $\exists r^-.A \sqsubseteq B$ is equivalent to $A \sqsubseteq \forall r.B$. Thus, it is easy to see that C_0 is satisfiable w.r.t. $\mathcal{T}$ iff $C_0 \not\sqsubseteq_\mathcal{C} L$.

Theorem 3.1.25 *In $\mathcal{ELI}$, subsumption w.r.t. IBoxes is PSPACE-hard.*

As in the case of $\mathcal{ELU}$, satisfiability in $\mathcal{ELI}$ w.r.t. IBoxes is trivial.

Functional Roles

Let IFBoxes be IBoxes which additionally may contain statements of the form $\mathsf{funct}(r)$, where r is a role name. An interpretation $\mathcal{I}$ *satisfies* $\mathsf{funct}(r)$ iff $r^\mathcal{I}$ is a partial function. We show that subsumption of $\mathcal{EL}$-concepts w.r.t. IFBoxes is EXPTIME-complete.

The upper bound is an immediate consequence of the fact that functional roles can be simulated in a DL providing number restrictions: instead of adding the statement $\mathsf{funct}(r)$ to the TBox, it suffices to replace every expression of the form $\exists r.C$ occurring in the IBox by the conjunction $\exists r.C \sqcap (\leqslant 1\ r)$. Subsumption w.r.t. IBoxes in the DL $\mathcal{ALC}$ extended with number restrictions is in EXPTIME [DGL94].

For the lower bound, the proof is by reduction of subsumption in the description logic $\mathcal{FL}_0^{\mathrm{tf}}$ w.r.t. IBoxes. $\mathcal{FL}_0^{\mathrm{tf}}$ is a variation of $\mathcal{FL}_0$, where all roles are interpreted as total functions. As noted below Corollary 12 of [TW05], the following result is an immediate consequence of the proof of Theorem 11 in the same paper:

Theorem 3.1.26 *(Toman, Weddell)*
Subsumption in $\mathcal{FL}_0^{\mathrm{tf}}$ w.r.t. IBoxes is EXPTIME-complete.

For our reduction, we exclude the $\top$-concept from $\mathcal{FL}_0^{\mathrm{tf}}$. This is justified since the proof of Theorem 3.1.26 also does not presuppose the presence of the $\top$-concept. Let C_0 and D_0 be $\mathcal{FL}_0^{\mathrm{tf}}$-concepts and $\mathcal{C}$ an $\mathcal{FL}_0^{\mathrm{tf}}$-IBox. The subsumption $C_0 \sqsubseteq_{\mathcal{C}} D_0$ is to be decided. Convert $\mathcal{C}$ into an $\mathcal{EL}$-IFBox $\mathcal{C}^*$ as follows.

- every value restriction $\forall r.C$ in $\mathcal{C}$ is replaced by $\exists r.C$;
- $\mathsf{funct}(r)$ is added to $\mathcal{C}^*$ for all role names $r \in \mathsf{N}_{\mathrm{role}}^{\mathcal{C}}$.

Moreover, every value restriction $\forall r.C$ in C_0 or D_0 is replaced by $\exists r.C$. We now show the following:

Lemma 3.1.27 *$C_0 \sqsubseteq_{\mathcal{C}} D_0$ iff $C_0 \sqsubseteq_{\mathcal{C}^*} D_0$.*

Proof. Proof by contraposition. ($\Rightarrow$) Assume $C_0 \not\sqsubseteq_{\mathcal{C}^*} D_0$, i.e., there is a model $\mathcal{I}$ of $\mathcal{C}^*$ and an $a_0 \in C_0^{\mathcal{I}} \setminus D_0^{\mathcal{I}}$. As $\mathsf{funct}(r) \in \mathcal{C}^*$ for all $r \in \mathsf{N}_{\mathrm{role}}^{\mathcal{C}}$, every $r^{\mathcal{I}}$ is a partial function. In a new interpretation $\mathcal{J}$, we extend these to total functions using an additional domain element $x_\perp$: for all $A \in \mathsf{N}_{\mathrm{con}}^{\mathcal{C}}$ and all $r \in \mathsf{N}_{\mathrm{role}}^{\mathcal{C}}$, define $\Delta^{\mathcal{J}}$ by

$$\begin{aligned}\Delta^{\mathcal{J}} &:= \Delta^{\mathcal{I}} \uplus \{x_\perp\}\\ A^{\mathcal{J}} &:= A^{\mathcal{I}}\\ r^{\mathcal{J}} &:= r^{\mathcal{I}} \cup \{(x, x_\perp) \mid (x,y) \notin r^{\mathcal{I}} \text{ for all } y \in \Delta^{\mathcal{I}}\}.\end{aligned}$$

Observe that $x_\perp$ is in the extension of no concept name. It is easy to see that $x \in C^{\mathcal{I}}$ iff $x \in C^{\mathcal{J}}$ for all $x \in \Delta^{\mathcal{I}}$ and all $\mathcal{EL}$-concepts C in which the $\top$-concept does not occur. Thus, $\mathcal{J}$ is a model of $\mathcal{C}^*$ and $a_0 \in C_0^{\mathcal{J}} \setminus D_0^{\mathcal{J}}$. As all $r \in \mathsf{N}_{\mathrm{role}}^{\mathcal{C}^*}$ are interpreted as total functions, $x \in (\exists r.C)^{\mathcal{J}}$ iff $x \in (\forall r.C)^{\mathcal{J}}$ for all $x \in \Delta^{\mathcal{J}}$, all $\mathcal{EL}$-concepts C, and all role names $r \in \mathsf{N}_{\mathrm{role}}^{\mathcal{C}}$. Hence, since $\mathcal{J}$ is a model of $\mathcal{C}^*$, it is one of $\mathcal{C}$ as well, and thus $C_0 \not\sqsubseteq_{\mathcal{C}} D_0$ as required.

($\Leftarrow$) Assume $C_0 \not\sqsubseteq_{\mathcal{C}} D_0$, i.e., there is a model $\mathcal{I}$ of $\mathcal{C}$ and an $a_0 \in C_0^{\mathcal{I}} \setminus D_0^{\mathcal{I}}$. Again, all $r \in \mathsf{N}_{\mathrm{role}}^{\mathcal{C}}$ are interpreted as total functions, implying $x \in (\exists r.C)^{\mathcal{I}}$ iff $x \in (\forall r.C)^{\mathcal{I}}$ for all $x \in \Delta^{\mathcal{I}}$, all $\mathcal{EL}$ concepts C, and all $r \in \mathsf{N}_{\mathrm{role}}^{\mathcal{C}}$. Since $\mathcal{I}$ is a model of $\mathcal{C}$, it is thus also a model of $\mathcal{C}^*$, yielding $C_0 \not\sqsubseteq_{\mathcal{C}^*} D_0$ as required. □

We thus obtain the following theorem.

Theorem 3.1.28 *Subsumption in $\mathcal{EL}$ w.r.t. IFBoxes is EXPTIME-complete.*

Note that satisfiability is trivial since every concept is satisfiable w.r.t. every IFBox.

At-most Restrictions

Let $\mathcal{EL}_{\leqslant 1}$ denote the extension of $\mathcal{EL}$ by number restrictions of the form $(\leqslant 1\ r)$ with the usual semantics from Definition 2.1.3. We show that subsumption in $\mathcal{EL}_{\leqslant 1}$ w.r.t. IBoxes is EXPTIME-complete.

The upper bound is an immediate consequence of the fact that subsumption w.r.t. IBoxes in the DL $\mathcal{ALC}$ extended with number restrictions is in EXPTIME [DGL94]. The lower bound is established by reduction of the subsumption problem w.r.t. IFBoxes, see above. As shown, $\mathcal{EL}_{\leqslant 1}$-IBoxes can be used to simulate IFBoxes, in which subsumption is EXPTIME-hard. Thus, the following theorem holds.

Theorem 3.1.29 *Subsumption in $\mathcal{EL}_{\leqslant 1}$ w.r.t. IBoxes is EXPTIME-complete.*

Note that the above result carries over to the extension of $\mathcal{EL}$ by arbitrary at-most restrictions and to $\mathcal{ELN}$, the extension of $\mathcal{EL}$ by arbitrary number restrictions.

Clearly, every $\mathcal{EL}_{\leqslant 1}$-concept is satisfiable w.r.t. every IBox. Note, however, that concept satisfiability is non-trivial when admitting number restrictions of the form $(\leqslant 0\ r)$. In this case, inconsistencies can be expressed as conjunction $\exists r.\top \sqcap (\leqslant 0\ r)$.

At-least Restrictions

Let $\mathcal{EL}_{\geqslant_2}$ be the extension of $\mathcal{EL}$ by an *at least-two* constructor $(\geqslant 2\ r)$ with the usual semantics defined in Definition 2.1.3. We show that subsumption in $\mathcal{EL}_{\geqslant_2}$ w.r.t. IBoxes is EXPTIME-complete.

The upper bound is an immediate consequence of the fact that, in the DL $\mathcal{ALC}$ extended with number restrictions, subsumption w.r.t. IBoxes is in EXPTIME [DGL94].

The lower bound is shown by reduction of subsumption in $\mathcal{ELU}$ w.r.t. IBoxes, which is EXPTIME-hard by Theorem 3.1.24. Thus, let C_0 and D_0 be $\mathcal{ELU}$-concepts and $\mathcal{C}$ an $\mathcal{ELU}$-IBox. W.l.o.g., assume that $\mathcal{C}$ is in normal form, i.e., all GCIs in $\mathcal{C}$ have one of the following forms:

$$\begin{aligned} C &\sqsubseteq D \\ C_1 \sqcap C_2 &\sqsubseteq C \\ C &\sqsubseteq C_1 \sqcup C_2 \\ C &\sqsubseteq \exists r.D \\ \exists r.C &\sqsubseteq D, \end{aligned}$$

where C, D, C_1, C_2 are concept names or $\top$. Every IBox can be converted into normal form by normalization rules similar to the ones presented in Figure 3.2.1. Note in particular that $C_1 \sqcup C_2 \sqsubseteq C$ can be replaced by the two rules $C_1 \sqsubseteq C$ and $C_2 \sqsubseteq C$, and that $C \sqsubseteq C_1 \sqcap C_2$ can similarly be replaced by $C \sqsubseteq C_1$ and $C \sqsubseteq C_2$. As usual, assume that C_0 and D_0 are concept *names*. The $\mathcal{EL}_{\geqslant_2}$-IBox $\mathcal{C}^*$ is now obtained from $\mathcal{C}$ by replacing every GCI of the form $I := C \sqsubseteq C_1 \sqcup C_2 \in \mathcal{C}$ by the GCIs

$$\begin{aligned} C &\sqsubseteq \exists r_I.A_I \sqcap \exists r_I.B_I \\ C \sqcap \exists r_I.(A_I \sqcap B_I) &\sqsubseteq C_1 \\ C \sqcap (\geqslant 2\ r_I) &\sqsubseteq C_2, \end{aligned}$$

where A_I, B_I are fresh concept names and r_I is a fresh role name. It is easy to see that $C_0 \sqsubseteq_{\mathcal{C}} D_0$ iff $C_0 \sqsubseteq_{\mathcal{C}^*} D_0$, which establishes the lower bound.

Theorem 3.1.30 *In $\mathcal{EL}_{\geqslant_2}$, subsumption w.r.t. IBoxes is EXPTIME-complete.*

Note that satisfiability is trivial since every concept is satisfiable w.r.t. every $\mathcal{EL}_{\geqslant_2}$-IBox.

Non-p-admissible Concrete Domains

Let $\mathcal{D}$ be a concrete domain that satisfies Condition 1 of Definition 2.3.3, but is not convex, i.e., violates Condition 2. Let $\mathcal{EL}(\mathcal{D})$ be the extension of $\mathcal{EL}$ by the concrete domain $\mathcal{D}$, see Section 2.3 for syntax and semantics. We show that subsumption in $\mathcal{EL}(\mathcal{D})$ is EXPTIME-complete.

For the lower bound, we first strengthen Theorem 3.1.24 as follows. Let a d-IBox be an IBox that, additionally, contains at most one GCI of the form $A \sqsubseteq B_1 \sqcup B_2$ with A, B_1, B_2 concept names. We show that subsumption of $\mathcal{EL}$-concepts w.r.t. d-IBoxes is EXPTIME-complete. The lower bound is proved by reduction of subsumption in $\mathcal{ELU}$ w.r.t. IBoxes, which is EXPTIME-hard by Theorem 3.1.24. Thus, let C_0 and D_0 be $\mathcal{ELU}$-concepts and $\mathcal{C}$ an $\mathcal{ELU}$-IBox. It is to be decided whether $C_0 \sqsubseteq_{\mathcal{C}} D_0$. Assume that C_0 and D_0 are concept names, and that $\mathcal{C}$ is in normal form as introduced in the previous section on at-least restrictions. For the reduction, introduce fresh concept names U_1, U_2 and a fresh role name r_{A,B_1,B_2} for each GCI $A \sqsubseteq B_1 \sqcup B_2 \in \mathcal{C}$. The d-IBox $\mathcal{C}^*$ is obtained from $\mathcal{C}$ by

- replacing each GCI $A \sqsubseteq B_1 \sqcup B_2$ by

$$\begin{aligned} \top &\sqsubseteq \exists r_{A,B_1,B_2}.\top \\ A \sqcap \exists r_{A,B_1,B_2}.U_1 &\sqsubseteq B_1 \\ A \sqcap \exists r_{A,B_1,B_2}.U_2 &\sqsubseteq B_2; \end{aligned}$$

- adding the GCI

$$\top \sqsubseteq U_1 \sqcup U_2.$$

It is easy to check that $C_0 \sqsubseteq_{\mathcal{C}} D_0$ iff $C_0 \sqsubseteq_{\mathcal{C}^*} D_0$. Together with the upper bound from Theorem 3.1.23, we obtain the following.

Theorem 3.1.31 *Subsumption of $\mathcal{EL}$-concepts w.r.t. d-IBoxes is EXPTIME-complete.*

In order to prove EXPTIME-completeness of subsumption in $\mathcal{EL}(\mathcal{D})$ with a non-convex concrete domain $\mathcal{D}$, we reduce $\mathcal{EL}$-subsumption w.r.t. d-IBoxes. Let C_0, D_0 be concept names and $\mathcal{C}$ a d-IBox. Since $\mathcal{D}$ is not convex, there is a satisfiable $\mathcal{D}$-conjunction c and a finite set Γ of concepts of the form $p(f_1, \ldots, f_k)$ such that c implies no concept from Γ, but every solution δ for c satisfies some concept in Γ. Fix a concept $X \in \Gamma$ such that some solutions of c satisfy X. By choice of X,

1. every solution of c satisfies either X or a concept in $\Gamma \setminus \{X\}$;
2. there is a solution of c that satisfies X; and
3. there is a solution of c that satisfies a concept in $\Gamma \setminus \{X\}$.

The $\mathcal{EL}(\mathcal{D})$-IBox $\mathcal{C}^*$ is obtained from $\mathcal{C}$ by replacing every GCI of the form $A \sqsubseteq B_1 \sqcup B_2$ by the following GCIs.

$$\begin{aligned} A \sqcap X &\sqsubseteq B_1 \\ A \sqcap Y &\sqsubseteq B_2 \quad \text{for every } Y \in \Gamma \setminus \{X\}. \end{aligned}$$

It is easy to see that $C_0 \sqsubseteq_{\mathcal{C}} D_0$ iff $C_0 \sqsubseteq_{\mathcal{C}^*} D_0$. Thus, subsumption in $\mathcal{EL}(\mathcal{D})$ w.r.t. IBoxes is EXPTIME-hard. By assumption, $\mathcal{D}$ satisfies Condition 1 of Definition 2.3.3, implying a matching upper bound by immediate consequence of the known EXPTIME-completeness of subsumption w.r.t. $\mathcal{ALC}(\mathcal{D})$-IBoxes. The relevant result is for $\mathcal{ALC}$ extended by concrete domains where only features (rather than sequences of features) are admitted inside the concrete domain constructor [Lut02].

Theorem 3.1.32 *Let $\mathcal{D}$ be a concrete domain that satisfies Condition 1 of Definition 2.3.3 but violates Condition 2, convexity. Then subsumption in $\mathcal{EL}(\mathcal{D})$ w.r.t. IBoxes is EXPTIME-complete.*

For example, this theorem applies to the concrete domains introduced at the end of Section 3.2.3.

Corollary 3.1.33 *For the following concrete domains $\mathcal{D}$, subsumption in $\mathcal{EL}(\mathcal{D})$ w.r.t. IBoxes is EXPTIME-complete:*

- *the concrete domain $\mathcal{Q}^{<_q,>_q}$;*
- *any concrete domain $\mathcal{S}^*$ with domain Σ^* for some finite alphabet Σ and the unary predicates pref_w and suff_w for every $w \in \Sigma^*$;*
- *any concrete domain $\mathcal{S}^*$ with domain Σ^* for some finite alphabet Σ, the unary predicates $\top_{\mathcal{S}^*}$ and $=_\varepsilon$, and the unary predicates pref_w, for each $w \in \Sigma^*$.*

Role Constructors

Consider the extension of $\mathcal{EL}$ by the role constructors $\neg$, $\cup$, or $\cdot^*$, with syntax and semantics defined as follows:

$$\begin{aligned}(\exists \neg r.C)^{\mathcal{I}} &:= \{x \in \Delta^{\mathcal{I}} \mid \exists y\colon (x,y) \notin r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}\\ (\exists r \cup s.C)^{\mathcal{I}} &:= \{x \in \Delta^{\mathcal{I}} \mid \exists y\colon (x,y) \in r^{\mathcal{I}} \cup s^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}\\ (\exists r^*.C)^{\mathcal{I}} &:= \{x \in \Delta^{\mathcal{I}} \mid \exists y\colon (x,y) \in (r^{\mathcal{I}})^* \wedge y \in C^{\mathcal{I}}\},\end{aligned}$$

where $(r^{\mathcal{I}})^*$ denotes the reflexive-transitive closure of $r^{\mathcal{I}}$. Let $\mathcal{EL}_{\neg r}$, $\mathcal{EL}_\cup$, and $\mathcal{EL}_*$ denote the respective extension of $\mathcal{EL}$ by one of the above constructors. We show that in all three cases subsumption w.r.t. IBoxes is EXPTIME-complete.

All lower bounds are established by reduction of subsumption in $\mathcal{ELU}$ w.r.t. IBoxes, which is EXPTIME-hard by Theorem 3.1.24. Let C_0 and D_0 be $\mathcal{ELU}$ concepts and $\mathcal{C}$ an $\mathcal{ELU}$-IBox. It is to be decided whether $C_0 \sqsubseteq_{\mathcal{C}} D_0$. As in the proof of Theorem 3.1.30, we assume that $\mathcal{C}$ is in normal form and that C_0 and D_0 are concept names. Let A be a fresh concept name and r, s be fresh role names. From $\mathcal{C}$, we construct an $\mathcal{EL}_{\neg r}$-IBox $\mathcal{C}_{\neg r}$, by replacing every GCI of the form $I := C \sqsubseteq C_1 \sqcup C_2$ by a the following GCIs:

$$\begin{aligned}C &\sqsubseteq \exists r.A\\ C \sqcap \exists s.A &\sqsubseteq C_1\\ C \sqcap \exists \neg s.A &\sqsubseteq C_2.\end{aligned}$$

Analogously, an $\mathcal{EL}_\cup$-IBox $\mathcal{C}_\cup$ is obtained by replacing every I by the GCIs

$$\begin{aligned}C &\sqsubseteq \exists r \cup s.A\\ C \sqcap \exists r.A &\sqsubseteq C_1\\ C \sqcap \exists s.A &\sqsubseteq C_2.\end{aligned}$$

For the $\mathcal{EL}_*$-IBox $\mathcal{C}_*$, every I is replaced by the GCIs

$$\begin{aligned}C &\sqsubseteq \exists r^*.A\\ C \sqcap A &\sqsubseteq C_1\\ C \sqcap \exists r.\exists r^*.A &\sqsubseteq C_2.\end{aligned}$$

It is easy to see that $C_0 \sqsubseteq_{\mathcal{C}} D_0$ iff $C_0 \sqsubseteq_{\mathcal{C}_{\neg_r}} D_0$ and analogously for $\mathcal{C}_\cup$ and $\mathcal{C}_*$. The EXPTIME upper bound holds due to the fact that subsumption w.r.t. IBoxes is in EXPTIME for $\mathcal{ALC}$ extended with the Boolean operators on roles [LS00]. The same holds for the description logic $\mathcal{ALC}_{\text{reg}}$ [FL79, Sch91].

Theorem 3.1.34 *In $\mathcal{EL}_{\neg_r}$, $\mathcal{EL}_\cup$, and $\mathcal{EL}_*$, subsumption w.r.t. IBoxes is EXPTIME-complete.*

In all three logics, satisfiability is trivial.

3.2 $\mathcal{EL}^{++}$-CBoxes with concrete domains

At first glance, the multitude of intractability results in the previous section might cast doubts on the quest for 'interesting' extensions of $\mathcal{EL}$ with tractable reasoning w.r.t. general TBoxes. In the present section, however, we show that there exists such an extension, namely $\mathcal{EL}^{++}$, in which, firstly, many constructs useful in ontology applications are provided; and secondly, even an extension of general TBoxes as well as ABoxes can be handled.

Syntax and semantics of $\mathcal{EL}^{++}$-concept descriptions have been introduced in Section 2.1. Recall that $\mathcal{EL}^{++}$ extends $\mathcal{EL}$ by the bottom concept ($\bot$) and nominals ($\{a\}$). Furthermore, we consider a finite number of p-admissible concrete domains $\mathcal{D}_1, \ldots, \mathcal{D}_n$. As underlying TBox formalism, we admit CBoxes which extend general TBoxes by RIs, together with ABoxes. Since in $\mathcal{EL}^{++}$ all other standard inference problems can be reduced to subsumption, see Section 2.1.1, we restrict our attention to subsumption w.r.t. $\mathcal{EL}^{++}(\mathcal{D}_1, \ldots, \mathcal{D}_n)$-CBoxes. To simplify notation, we do not always explicitly mention the concrete domain extension.

Before turning to decide subsumption w.r.t. $\mathcal{EL}^{++}$-CBoxes, we would like to emphasize that, though seemingly relatively inexpressive, $\mathcal{EL}^{++}$-CBoxes allow to express several notions important in KR-applications.

- *GCIs*: As CBoxes extend general TBoxes, GCIs can trivially be expressed.
- *Role hierarchies*: a role hierarchy states that one role, e.g., specific_surface_division_of, is more specific than another, e.g., part_of. Role hierarchies can be expressed by simple RIs of the form

  $$\mathsf{specific\_surface\_division\_of} \sqsubseteq \mathsf{part\_of}.$$

- *Transitive roles*: if a role, e.g., part_of, is declared transitive then it must always be interpreted by a transitive relation. Transitivity of roles can be declared by means of RIs of the form

  $$\mathsf{part\_of} \circ \mathsf{part\_of} \sqsubseteq \mathsf{part\_of}.$$

- *Right-identities*: right-identities have been discussed in Section 1.2. They can be immediately expressed by role inclusions (RIs). For instance, in order to state that a finding at a part implies the same finding at the whole, we can add the following RI to the CBox.

  $$\mathsf{finding\_at} \circ \mathsf{part\_of} \sqsubseteq \mathsf{finding\_at}.$$

- *Disjointness constraints*: a disjointness constraint between concept A, B enforces that no element of an interpretation is at the same time a witness of A and B. Such constraints can be expressed by GCIs. For instance, the GCI

  $$\mathit{Artery} \sqcap \mathit{Vein} \sqsubseteq \bot$$

enforces that the concept *Artery* and *Vein* are disjoint.

- *Unique name assumption*: the definition of ABox individuals, see Definition 2.1.7, usually states that the unique name assumption holds. In cases where this is not stated a priori and the unique name assumption is desired for (some) individuals occurring in the ABox, it can be expressed by adding GCIs of the form

$$\{\mathsf{a.basilaris}\} \sqcap \{\mathsf{a.subclavia}\} \sqsubseteq \bot$$

  for every pair of individual names for which the UNA is desired. In the above example, the UNA holds for the ABox individuals a.basilaris and a.subclavia.

- *Domain restrictions*: domain restrictions on roles are used to specify that only instances of a certain concept can have a successor w.r.t. the role in question. Such restrictions can be expressed by a GCI of the following form:

$$\exists \mathsf{has\_patient\_id}.\top \sqsubseteq \mathsf{Person}.$$

  In every model satisfying this GCI, the existence of a successor w.r.t. the role has_patient_id implies that the relevant element is in the interpretation of Person.

Our aim is to classify $\mathcal{EL}^{++}(\mathcal{D}_1, \ldots, \mathcal{D}_n)$-CBoxes in polynomial time. To this end, we introduce a normal form for CBoxes in Section 3.2.1 and show in Section 3.2.2 how this gives rise to a relatively simple subsumption algorithm based on certain saturation rules. As our language is restricted to p-admissible concrete domains, we show in Section 3.2.3 that there exist useful concrete domains that fall in this class. The purpose of Section 3.2.4 is to highlight the, to some extent astonishing, difference in computational complexity of the subsumption problem between extensions of $\mathcal{FL}_0$ on the one hand and extensions of $\mathcal{EL}$ on the other.

3.2.1 Formal preliminaries

In order to refer to atomic concepts and other 'simple' concept descriptions more easily, denote by P the following set of concepts.

$$\mathsf{P} := \mathsf{N}_{\mathrm{con}} \cup \{\top, \bot\} \cup \{\{a\} \mid a \in \mathsf{N}_{\mathrm{nom}}\} \cup \{p(f_1, \ldots, f_n) \mid p \in \mathcal{P}^{\mathcal{D}} \wedge f_1, \ldots, f_n \in \mathsf{N}_{\mathrm{fe}}\}$$

For a given CBox $\mathcal{C}$, denote by $\mathsf{P}^{\mathcal{C}}$ the subset of P whose elements actually occur in $\mathcal{C}$. Our first step towards classifying CBoxes is to devise a normal form for them. By means of the following definition, we restrict GCIs and RIs occurring in $\mathcal{EL}^{++}$-CBoxes to the most simple cases.

Definition 3.2.1 (Normal form for $\mathcal{EL}^{++}$-CBoxes)
An $\mathcal{EL}^{++}$-CBox $\mathcal{C}$ is in *normal form* iff

1. all GCIs are of the form $C_1 \sqsubseteq D$, $C_1 \sqcap C_2 \sqsubseteq D$, $C_1 \sqsubseteq \exists r.C_2$, or $\exists r.C_1 \sqsubseteq D$ with $C_1, C_2, D \in \mathsf{P}$ and $C_1, C_2 \neq \bot$;
2. all RIs are of the form $r \sqsubseteq s$ or $r_1 \circ r_2 \sqsubseteq s$ with $r, r_1, r_2, s \in \mathsf{N}_{\mathrm{role}}$. □

The above normal form imposes no real restriction on a potential subsumption algorithm because every $\mathcal{EL}^{++}$-CBox can be normalized easily, as the next lemma shows.

Lemma 3.2.2 *For every $\mathcal{EL}^{++}$-CBox $\mathcal{C}$, there is an equivalent $\mathcal{EL}^{++}$-CBox $\mathcal{C}'$ in normal form such that $|\mathcal{C}'|$ is linear in $|\mathcal{C}|$. Moreover, $\mathcal{C}'$ can be computed in time linear in $|\mathcal{C}|$.*

NF1	$r_1 \circ \cdots \circ r_k \sqsubseteq s \longrightarrow \{r_1 \circ \cdots \circ r_{k-1} \sqsubseteq u, u \circ r_k \sqsubseteq s\}$
NF2	$C \sqcap \hat{D} \sqsubseteq E \longrightarrow \{\hat{D} \sqsubseteq A, C \sqcap A \sqsubseteq E\}$
NF3	$\exists r.\hat{C} \sqsubseteq D \longrightarrow \{\hat{C} \sqsubseteq A, \exists r.A \sqsubseteq D\}$
NF4	$\bot \sqsubseteq D \longrightarrow \emptyset$
NF5	$\hat{C} \sqsubseteq \hat{D} \longrightarrow \{\hat{C} \sqsubseteq A, A \sqsubseteq \hat{D}\}$
NF6	$B \sqsubseteq \exists r.\hat{C} \longrightarrow \{B \sqsubseteq \exists r.A, A \sqsubseteq \hat{C}\}$
NF7	$B \sqsubseteq C \sqcap D \longrightarrow \{B \sqsubseteq C, B \sqsubseteq D\}$

$\hat{C}, \hat{D} \notin \mathsf{P}$, u denotes a fresh role name, and A a fresh concept name.

Figure 3.2.1: Normalization Rules

PROOF. We show that $\mathcal{C}$ can be converted into normal form using the translation rules shown in Figure 3.2.1. Note that $\hat{C}, \hat{D}$ denote non-atomic concept descriptions while B, C, D, E stand for arbitrary concept descriptions. Note also that Rule **NF2** is defined modulo commutativity of conjunction. In order to normalize $\mathcal{C}$, the above rules are applied in two phases:

1. exhaustively apply rules **NF1** to **NF4**;
2. exhaustively apply rules **NF5** to **NF7**.

Here 'rule application' means replacing the GCI on the left-hand side with all GCIs in the set on the right-hand-side. Normalizing $\mathcal{C}$ in the above sense produces a normalized CBox $\mathcal{C}'$ of linear size in $|\mathcal{C}|$ and takes at most $|\mathcal{C}|$ rule applications, and thus only linear time. □

Note that applying all normalization rules arbitrarily might cause a quadratic blow-up in the worst case due to the duplication of the concept B in Rule **NF7**.

In addition to normalization, we may w.l.o.g. restrict our subsumption algorithm to subsumption between concept *names*: for every $\mathcal{EL}^{++}$-CBox $\mathcal{C}$ and arbitrary $\mathcal{EL}^{++}$-concept descriptions C, D, $C \sqsubseteq_{\mathcal{C}} D$ iff $A \sqsubseteq_{\mathcal{C} \cup \{A \sqsubseteq C, D \sqsubseteq B\}} B$, where A, B are fresh concept names.

We are now ready to introduce the actual subsumption algorithm in the following section.

3.2.2 Deciding subsumption w.r.t. $\mathcal{EL}^{++}$-CBoxes

Given a normalized $\mathcal{EL}^{++}$-CBox $\mathcal{C}$, we not only want to decide single subsumption relations, but to classify $\mathcal{C}$, i.e., decide all subsumption relations w.r.t. pairs of names from $\mathcal{C}$. As a first step to this end, we introduce *implication sets* $\mathsf{Imp}(P)$ for simple concepts $P \in \mathsf{P}^{\mathcal{C}}$ and $\mathsf{Imp}(r)$ for role names r occurring in $\mathcal{C}$. The underlying idea is to collect all simple concepts subsuming a simple concept $P \in \mathsf{P}^{\mathcal{C}}$ in $\mathsf{Imp}(P)$ and all pairs (P, Q) of simple concepts in $\mathsf{Imp}(r)$ iff the concept name P is subsumed by $\exists r.Q$. For concept names A, B occurring in $\mathcal{C}$, the subsumption $A \sqsubseteq_{\mathcal{C}} B$ can then be tested by checking whether B is contained in $\mathsf{Imp}(A)$.

Definition 3.2.3 (Implication sets, conjunctions)
For every $P \in \mathsf{P}^{\mathcal{C}}$ and every $r \in \mathsf{N}^{\mathcal{C}}_{\text{role}}$, Imp maps P onto a subset of $\mathsf{P}^{\mathcal{C}} \cup \{\top, \bot\}$ and r onto a subset of $\mathsf{P}^{\mathcal{C}} \times \mathsf{P}^{\mathcal{C}}$. Initially, let $\mathsf{Imp}(P) := \{P, \top\}$ and $\mathsf{Imp}(r) := \emptyset$. The mapping

CR1	If $C' \in \mathsf{Imp}(C)$, $C' \sqsubseteq D \in \mathcal{C}$, and $D \notin \mathsf{Imp}(C)$ then $\mathsf{Imp}(C) := \mathsf{Imp}(C) \cup \{D\}$
CR2	If $C_1, C_2 \in \mathsf{Imp}(C)$, $C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{C}$, and $D \notin \mathsf{Imp}(C)$ then $\mathsf{Imp}(C) := \mathsf{Imp}(C) \cup \{D\}$
CR3	If $C' \in \mathsf{Imp}(C)$, $C' \sqsubseteq \exists r.D \in \mathcal{C}$, and $(C, D) \notin \mathsf{Imp}(r)$ then $\mathsf{Imp}(r) := \mathsf{Imp}(r) \cup \{(C, D)\}$
CR4	If $(C, D) \in \mathsf{Imp}(r)$, $D' \in \mathsf{Imp}(D)$, $\exists r.D' \sqsubseteq E \in \mathcal{C}$, and $E \notin \mathsf{Imp}(C)$ then $\mathsf{Imp}(C) := \mathsf{Imp}(C) \cup \{E\}$
CR5	If $(C, D) \in \mathsf{Imp}(r)$, $\bot \in \mathsf{Imp}(D)$, and $\bot \notin \mathsf{Imp}(C)$, then $\mathsf{Imp}(C) := \mathsf{Imp}(C) \cup \{\bot\}$
CR6	If $\{a\} \in \mathsf{Imp}(C) \cap \mathsf{Imp}(D)$, $C \rightsquigarrow D$, and $\mathsf{Imp}(D) \not\subseteq \mathsf{Imp}(C)$ then $\mathsf{Imp}(C) := \mathsf{Imp}(C) \cup \mathsf{Imp}(D)$
CR7	If $\mathsf{con}_j(\mathsf{Imp}(C))$ is unsatisfiable in $\mathcal{D}_j$ and $\bot \notin \mathsf{Imp}(C)$, then $\mathsf{Imp}(C) := \mathsf{Imp}(C) \cup \{\bot\}$
CR8	If $\mathsf{con}_j(\mathsf{Imp}(C))$ implies $p(f_1, \ldots, f_k) \in \mathsf{P}^{\mathcal{C}}$ in $\mathcal{D}_j$ and $p(f_1, \ldots, f_k) \notin \mathsf{Imp}(C)$, then $\mathsf{Imp}(C) := \mathsf{Imp}(C) \cup \{p(f_1, \ldots, f_k)\}$
CR9	If $p(f_1, \ldots, f_k), p'(f'_1, \ldots, f'_{k'}) \in \mathsf{Imp}(C)$, $p \in \mathcal{P}^{\mathcal{D}_j}$, $p' \in \mathcal{P}^{\mathcal{D}_\ell}$, $j \neq \ell$, $f_s = f'_t$ for some s, t, and $\bot \notin \mathsf{Imp}(C)$, then $\mathsf{Imp}(C) := \mathsf{Imp}(C) \cup \{\bot\}$
CR10	If $(C, D) \in \mathsf{Imp}(r)$, $r \sqsubseteq s \in \mathcal{C}$, and $(C, D) \notin \mathsf{Imp}(s)$ then $\mathsf{Imp}(s) := \mathsf{Imp}(s) \cup \{(C, D)\}$
CR11	If $(C, D) \in \mathsf{Imp}(r_1)$, $(D, E) \in \mathsf{Imp}(r_2)$, $r_1 \circ r_2 \sqsubseteq r_3 \in \mathcal{C}$, and $(C, E) \notin \mathsf{Imp}(r_3)$ then $\mathsf{Imp}(r_3) := \mathsf{Imp}(r_3) \cup \{(C, E)\}$

Table 3.2.1: Completion Rules

Imp is then extended by means of the completion rules shown in Table 3.2.1 until no more rule is applicable. $\mathsf{Imp}(P)$ and $\mathsf{Imp}(r)$ are called *implication sets* of P and r, respectively.

Rule **CR6** is defined in terms of a binary relation $\rightsquigarrow \subseteq \mathsf{P}^{\mathcal{C}} \times \mathsf{P}^{\mathcal{C}}$ defined as follows. For every $P, Q \in \mathsf{P}^{\mathcal{C}}$, $P \rightsquigarrow Q$ iff there exists some $k \in \mathbb{N} \setminus \{0\}$ and $C_1, \dots, C_k \in \mathsf{P}^{\mathcal{C}}$ with

- $C_1 = P$ or $C_1 = \{b\}$ for some $b \in \mathsf{N}_{\mathrm{nom}}$;
- for all $j \in \{1, \dots, k-1\}$ there exists some $r \in \mathsf{N}^{\mathcal{C}}_{\mathrm{role}}$ with $(C_j, C_{j+1}) \in \mathsf{Imp}(r)$; and
- $C_k = Q$.

Rules **CR7** and **CR8** are defined in terms of the conjunction $\mathsf{con}_j(\mathsf{Imp}(C))$. For every set $\Gamma \subseteq \mathsf{P}^{\mathcal{C}}$, $\mathsf{con}_j(\Gamma)$ is defined as follows.

$$\mathsf{con}_j(\Gamma) := \bigwedge_{p(f_1,\dots,f_k)\in\Gamma \text{ with } p\in\mathcal{P}^{\mathcal{D}_j}} p(f_1, \dots, f_k)$$

A *solution* for $\mathsf{con}_j(\Gamma)$ is a mapping $\delta\colon \mathsf{N}_{\mathrm{fe}} \to \Delta^{\mathcal{D}_j}$ with $(\delta(f_1), \dots, \delta(f_k)) \in p^{\mathcal{D}_j}$ for every conjunct $p(f_1, \dots, f_k)$ of $\mathsf{con}_j(\Gamma)$. □

Note that the conjunction $\mathsf{con}_j(\Gamma)$ merely collects all predicates in Γ that appertain to the concrete domain $\mathcal{D}_j$. A conjunction $\mathsf{con}_j(S(C))$ is satisfiable iff there exists a solution for it. Denote by $\delta \models p(f_1, \dots, f_k)$ the fact that $(\delta(f_1), \dots, \delta(f_k)) \in p^{\mathcal{D}_j}$.

The following two lemmas show that, after exhaustive application of the above completion rules, deciding subsumption between names from a given $\mathcal{EL}^{++}$-CBox can be reduced to looking up names in implication sets. We begin by showing soundness.

Lemma 3.2.4 *(Soundness)*
Let $\mathcal{C}$ be a normalized CBox. Let Imp be the mapping obtained after exhaustive application of the rules in Table 3.2.1 to $\mathcal{C}$. Let $A, B \in \mathsf{N}^{\mathcal{C}}_{\mathrm{con}}$. Then $A \sqsubseteq_{\mathcal{C}} B$ if one of the following two conditions holds:

S1 $\mathsf{Imp}(A) \cap \{B, \bot\} \neq \emptyset$*; or*

S2 there is an $\{a\} \in \mathsf{P}^{\mathcal{C}}$ *with* $\bot \in \mathsf{Imp}(\{a\})$.

Proof. Assume that the algorithm is applied to $\mathcal{C}$ yielding the sequence of mappings $\mathsf{Imp}_0, \dots, \mathsf{Imp}_n$. Let A_0, B_0 be two concept names such that (at least) one of the Conditions $S1$ and $S2$ is satisfied. To show $A_0 \sqsubseteq_{\mathcal{C}} B_0$, we prove the following.

Claim. For all $n \in \mathbb{N}$, all models $\mathcal{I}$ of $\mathcal{C}$, all $r \in \mathsf{N}^{\mathcal{C}}_{\mathrm{role}}$, and all $x \in C^{\mathcal{I}}$, it holds that

(a) if $D \in \mathsf{Imp}_n(C)$ then $x \in D^{\mathcal{I}}$; and

(b) if $(C, D) \in \mathsf{Imp}_n(r)$ then there is a $y \in \Delta^{\mathcal{I}}$ with $(x, y) \in r^{\mathcal{I}}$ and $y \in D^{\mathcal{I}}$.

The claim is proved by induction on n. Let $\mathcal{I}$ be a model of $\mathcal{C}$ and $x \in C^{\mathcal{I}}$.

($n = 0$) Trivial because $\mathsf{Imp}_0(C) = \{C, \top\}$ and $\mathsf{Imp}_0(r) = \emptyset$. Hence, (a) obviously holds while the precondition of (b) does not.

($n > 0$) (a) Assume $D \in \mathsf{Imp}_n(C) \setminus \mathsf{Imp}_{n-1}(C)$ because otherwise the claim holds by induction hypothesis (IH). We make a case distinction depending on the rule by which $\mathsf{Imp}_n(C)$ is extended with D.

CR1 Then there exists some $C' \in \mathsf{Imp}_{n-1}(C)$ and a GCI $I := C' \sqsubseteq D \in \mathcal{C}$. By Claim (a) of IH, $x \in C'^{\mathcal{I}}$, implying $x \in D^{\mathcal{I}}$ by I.

CR2 Then there exist $C_1, C_2 \in \mathsf{Imp}_{n-1}(C)$ and a GCI $I := C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{C}$. By Claim (a) of IH, $C_1, C_2 \in \mathsf{Imp}_{n-1}(A)$ yields $x \in C_1^{\mathcal{I}}$ and $x \in C_2^{\mathcal{I}}$, implying by I that $x \in D^{\mathcal{I}}$.

CR4 Then there exist $E, E' \in \mathsf{P}^{\mathcal{C}}$, a role name $r \in \mathsf{N}^{C}_{\mathrm{role}}$, and a GCI $I := \exists r.E' \sqsubseteq D \in \mathcal{C}$ with $(C, E) \in \mathsf{Imp}_{n-1}(r)$ and $E' \in \mathsf{Imp}_{n-1}(E)$. By Claim (b) of IH, there is a $y \in \Delta^{\mathcal{I}}$ with $(x, y) \in r^{\mathcal{I}}$ and $y \in E^{\mathcal{I}}$. By Claim (a) of IH, $y \in E'^{\mathcal{I}}$. Thus, I yields $x \in D^{\mathcal{I}}$.

CR5 Then $D = \bot$ and there is an $E \in \mathsf{P}^{\mathcal{C}}$ such that $(C, E) \in \mathsf{Imp}_{n-1}(r)$ for some $r \in \mathsf{N}^{\mathcal{C}}_{\mathrm{role}}$ and $\bot \in \mathsf{Imp}_{n-1}(E)$. By Claim (b) of IH, there is a $y \in \Delta^{\mathcal{I}}$ with $(x, y) \in r^{\mathcal{I}}$ and $y \in E^{\mathcal{I}}$. By Claim (a) of IH, $y \in \bot^{\mathcal{I}} = \emptyset$. Hence, there are no models $\mathcal{I}$ of $\mathcal{C}$ with $C^{\mathcal{I}} \neq \emptyset$. Thus, adding $\bot$ to $\mathsf{Imp}_n(C)$ trivially preserves Claim (a).

CR6 Then there exists some $E \in \mathsf{P}^{\mathcal{C}}$ and $a \in \mathsf{N}_{\mathrm{nom}}$ with $\{a\} \in \mathsf{Imp}_{n-1}(C) \cap \mathsf{Imp}_{n-1}(E)$, $D \in \mathsf{Imp}_{n-1}(E)$, and there are $C_1, \ldots, C_k \in \mathsf{P}^{\mathcal{C}}$ such that

(i) $C_1 = C$ or $C_1 = \{b\}$ for some individual name b;

(ii) $C_k = E$;

(iii) for all $i \in \{1, \ldots, k-1\}$ there exists some $r_i \in \mathsf{N}^{\mathcal{C}}_{\mathrm{role}}$ with $(C_i, C_{i+1}) \in \mathsf{Imp}_{n-1}(r_i)$.

By Claim (b) of IH and (iii), there are $y_1, \ldots, y_k \in \Delta^{\mathcal{I}}$ with $y_1 \in \{x\} \cup \{b^{\mathcal{I}} | b \in \mathsf{N}_{\mathrm{nom}}\}$, $y_k \in C_k^{\mathcal{I}} = E^{\mathcal{I}}$, and, for every $i \in \{1, \ldots, k-1\}$, $(y_i, y_{i+1}) \in r_i^{\mathcal{I}}$ for some $r_i \in \mathsf{N}^{\mathcal{C}}_{\mathrm{role}}$. By Claim (a) of IH, $x \in C^{\mathcal{I}}$ and $\{a\} \in \mathsf{Imp}_{n-1}(C) \cap \mathsf{Imp}_{n-1}(E)$ implies $x = a^{\mathcal{I}} = y_k$. Also by Claim (a), $D \in \mathsf{Imp}_{n-1}(E)$ implies $y_k \in D^{\mathcal{I}}$. Thus, $x \in D^{\mathcal{I}}$ as required.

CR7 Then $D = \bot$ and $\mathsf{con}_j(\mathsf{Imp}_{n-1}(C))$ is unsatisfiable for some j. Define a function $\delta \colon \mathsf{N}_{\mathrm{fe}} \to \Delta^{\mathcal{D}_j}$ with $\delta(f) := f^{\mathcal{I}}(x)$ for every $f \in \mathsf{N}_{\mathrm{fe}}$. By Claim (a) of IH, $x \in p(f_1, \ldots, f_k)^{\mathcal{I}}$ for every conjunct $p(f_1, \ldots, f_k)$ of $\mathsf{con}_j(\mathsf{Imp}_{n-1}(C))$. Thus, δ is a solution for $\mathsf{con}_j(\mathsf{Imp}_{n-1}(C))$, contradicting its unsatisfiability. Hence, there can be no model $\mathcal{I}$ of $\mathcal{C}$ with $C^{\mathcal{I}} \neq \emptyset$. Adding $\bot$ to $S(C)$ thus (trivially) preserves Claim (a).

CR8 Then D is of the form $p(f_1, \ldots, f_k)$ with $p \in \mathcal{P}^{\mathcal{D}_j}$ for some i, and $\mathsf{con}_j(\mathsf{Imp}_{n-1}(C))$ implies D. As in the previous case, $x \in p(f_1, \ldots, f_k)^{\mathcal{I}}$ for every conjunct $p(f_1, \ldots, f_k)$ of $\mathsf{con}_j(\mathsf{Imp}_{n-1}(C))$ by Claim (a) of IH. Since $\mathsf{con}_j(\mathsf{Imp}_{n-1}(C))$ implies D, $x \in D^{\mathcal{I}}$ as required.

CR9 Then $D = \bot$ and there are $p(f_1, \ldots, f_k) \in \mathsf{Imp}_{n-1}(C)$ and $p'(f'_1, \ldots, f'_{k'}) \in \mathsf{Imp}_{n-1}(C)$ such that $p \in \mathcal{P}^{\mathcal{D}_i}$ and $p' \in \mathcal{P}^{\mathcal{D}_j}$ with $i \neq j$. By Claim (a) of IH, $x \in p(f_1, \ldots, f_k)^{\mathcal{I}} \cap p'(f'_1, \ldots, f'_{k'})^{\mathcal{I}}$. Thus $f_i^{\mathcal{I}} \in \Delta^{\mathcal{D}_i} \cap \Delta^{\mathcal{D}_j}$, contradicting the disjointness of $\Delta^{\mathcal{D}_i}$ and $\Delta^{\mathcal{D}_j}$. Again, Claim (a) is trivially preserved.

For (b), assume $(C, D) \in \mathsf{Imp}_n(r) \setminus \mathsf{Imp}_{n-1}(r)$ and make a case distinction according to the rule by which $R_n(r)$ is extended by (C, D).

CR3 Then there is a $C' \in \mathsf{P}^{\mathcal{C}}$ with $C' \in \mathsf{Imp}_{n-1}(C)$ and a GCI $I := C' \sqsubseteq \exists r.D \in \mathcal{C}$. By Claim (a) of IH, $x \in C^{\mathcal{I}}$ implies $x \in C'^{\mathcal{I}}$. By I, there is a y such that $(x, y) \in r^{\mathcal{I}}$ and $y \in D^{\mathcal{I}}$ as required.

CR10 Then $(C, D) \in \mathsf{Imp}_{n-1}(s)$ for some $s \in \mathsf{N}^{\mathcal{C}}_{\mathrm{role}}$ with $I := s \sqsubseteq r \in \mathcal{C}$. By Claim (b) of IH, there is a $y \in \Delta^{\mathcal{I}}$ such that $(x, y) \in s^{\mathcal{I}}$ and $y \in D^{\mathcal{I}}$. Due to $\mathcal{I}$, $(x, y) \in r^{\mathcal{I}}$.

CR11 Then there is an $E \in \mathsf{P}^{\mathcal{C}}$ such that $(C, E) \in \mathsf{Imp}_{n-1}(r_1)$ and $(E, D) \in \mathsf{Imp}_{n-1}(r_2)$ for some $r_1, r_2 \in \mathsf{N}^{\mathcal{C}}_{\text{role}}$ with $I := r_1 \circ r_2 \sqsubseteq r \in \mathcal{C}$. By Claim (b) of IH, there is a $y \in \Delta^{\mathcal{I}}$ such that $(x, y) \in r_1^{\mathcal{I}}$ and $y \in E^{\mathcal{I}}$. Another application of Claim (b) yields the existence of a $z \in \Delta^{\mathcal{I}}$ with $(y, z) \in r_2^{\mathcal{I}}$ and $z \in D^{\mathcal{I}}$. Because of I, $(x, z) \in r^{\mathcal{I}}$.

This finishes the proof of our Claim. It is now easy to prove $A_0 \sqsubseteq_{\mathcal{C}} B_0$, distinguishing whether Condition $S1$ or $S2$ is satisfied.

$S1$ Let $B_0 \in \mathsf{Imp}_m(A_0)$. By Claim (a), $x \in B_0^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{C}$ and all $x \in A_0^{\mathcal{I}}$. In other words, $A_0 \sqsubseteq_{\mathcal{C}} B_0$. Now let $\bot \in \mathsf{Imp}_m(A_0)$. By Claim (a), $x \in \bot^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{C}$ and all $x \in A_0^{\mathcal{I}}$. Hence, there are no models $\mathcal{I}$ of $\mathcal{C}$ with $A_0^{\mathcal{I}} \neq \emptyset$, implying $A_0 \sqsubseteq_{\mathcal{C}} B_0$.

$S2$ Let $\bot \in \mathsf{Imp}_m(\{a\})$ for some $a \in \mathsf{N}_{\text{nom}}$. By Claim (a), $a^{\mathcal{I}} \in \bot^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{C}$. Hence, $\mathcal{C}$ has no models, implying $A_0 \sqsubseteq_{\mathcal{C}} B_0$. □

It remains to show that, after exhaustive application of the completion rules, every subsumption relation between names in the CBox $\mathcal{C}$ is reflected in the relevant implication sets.

Lemma 3.2.5 *(Completeness)*
Let Imp *be the mapping obtained after exhaustive application of the rules in Table 3.2.1 for the normalized CBox* $\mathcal{C}$*. Let* $A, B \in \mathsf{N}^{\mathcal{C}}_{\text{con}}$*. Then* $A \sqsubseteq_{\mathcal{C}} B$ *implies that one of the following two conditions holds:*

S1 $\mathsf{Imp}(A) \cap \{B, \bot\} \neq \emptyset$*, or*

S2 there is an $\{a\} \in \mathsf{P}^{\mathcal{C}}$ *with* $\bot \in \mathsf{Imp}(\{a\})$*.*

PROOF. We show the contrapositive. Assume that $S1$ and $S2$ do not hold after exhaustive rule application. We show that this implies $A_0 \not\sqsubseteq_{\mathcal{C}} B_0$ by constructing a model $\mathcal{I}$ of $\mathcal{C}$ with $a \in A_0^{\mathcal{I}} \setminus B_0^{\mathcal{I}}$ for some $a \in \Delta^{\mathcal{I}}$.

Let $m \in \mathbb{N}$ be the least index with $\mathsf{Imp}_m = \mathsf{Imp}_{m+1}$. For convenience, denote Imp_m by Imp. Let $\mathsf{P}^{\mathcal{C}}_{-} := \{C \in \mathsf{P}^{\mathcal{C}} \mid A_0 \rightsquigarrow C\}$. Then define a binary relation $\sim \subseteq \mathsf{P}^{\mathcal{C}}_{-} \times \mathsf{P}^{\mathcal{C}}_{-}$ by:

$$C \sim D \text{ iff } C = D \text{ or } \{a\} \in \mathsf{Imp}(C) \cap \mathsf{Imp}(D) \text{ for some } a \in \mathsf{N}^{\mathcal{C}}_{\text{nom}}.$$

Rule **CR6** guarantees that $\sim$ is an equivalence relation on $\mathsf{P}^{\mathcal{C}}_{-}$. For every $C \in \mathsf{P}^{\mathcal{C}}_{-}$, denote by $[C]$ the equivalence class of C w.r.t. $\sim$. These equivalence classes will be the domain elements of the model to be constructed. Before actually defining this model, we prove two claims:

CLAIM 1. For all $C, C' \in \mathsf{P}^{\mathcal{C}}_{-}$ with $C \sim C'$ and all $r \in \mathsf{N}^{\mathcal{C}}_{\text{role}}$, we have

(a) $\mathsf{Imp}(C) = \mathsf{Imp}(C')$; and

(b) $(C, D) \in \mathsf{Imp}(r)$ implies $(C', D) \in \mathsf{Imp}(r)$.

Proof of Claim 1: If Rule **CR6** is applied exhaustively then Claim (a) immediately holds. We show Claim (b) by induction on the smallest index i such that $(C, D) \in \mathsf{Imp}_i(r)$.

$(i = 0)$ Trivial because $\mathsf{Imp}_0(r) = \emptyset$.

$(i > 0)$ Let $(C, D) \in \mathsf{Imp}_i(r) \setminus \mathsf{Imp}_{i-1}(r)$. We make a case distinction for the rule by which $\mathsf{Imp}_i(r)$ is extended by (C, D).

CR3 Then there is an $E \in \mathsf{Imp}_{i-1}(C)$ and a GCI $E \sqsubseteq \exists r.D \in \mathcal{C}$. Since $C \sim C'$, **CR6** ensures that $E \in \mathsf{Imp}_j(C')$ for some $j \geq 0$. Thus, **CR3** yields $(D, E) \in \mathsf{Imp}(r)$.

CR10 Then $(C, D) \in \mathsf{Imp}_{i-1}(s)$ for some role name s with $s \sqsubseteq r \in \mathcal{C}$. By IH, this implies $(C', D) \in \mathsf{Imp}(j)s$ for some $j \geq 0$. Thus, **CR12** yields $(C', D) \in \mathsf{Imp}(r)$.

CR11 Then there is an $E \in \mathsf{P}^{\mathcal{C}}$ with $(C, E) \in \mathsf{Imp}_{i-1}(r_1)$ and $(E, D) \in \mathsf{Imp}_{i-1}(r_2)$ for some role names r_1, r_2 with $r_1 \circ r_2 \sqsubseteq r \in \mathcal{C}$. By definition of $\rightsquigarrow$, $C \in \mathsf{P}^{\mathcal{C}}_{-}$ implies $D \in \mathsf{P}^{\mathcal{C}}_{-}$. Thus, the IH yields $(C, E) \in \mathsf{Imp}_{i-1}(r_1)$, implying $(C', E) \in \mathsf{Imp}_j(r_1)$ for some $j \geq 0$. **CR13** will eventually be applied to $(C', E) \in \mathsf{Imp}_\ell(r_1)$ and $(E, D) \in \mathsf{Imp}_\ell(r_2)$ for some $\ell \geq 0$, yielding $(C', D) \in \mathsf{Imp}_{\ell+1}(r) \subseteq \mathsf{Imp}(r)$.

This finishes the proof of Claim 1. Claim (a) allows us to unambiguously identify a given equivalence class $[C]$ with the set $\mathsf{Imp}(C)$. This will be used implicitly in the following.

Claim 2. For each $C \in \mathsf{P}^{\mathcal{C}}_{-}$ and each $i \in \{1, \dots, n\}$, there exist solutions $\delta([C], i)$ for $\mathsf{con}_i(\mathsf{Imp}(C))$ such that, for all concepts $D \in \mathsf{P}^{\mathcal{C}}$ of the form $p(f_1, \dots, f_k)$ with $p \in \mathcal{P}^{\mathcal{D}_i}$, we have $\delta([C], t) \models D$ iff $D \in \mathsf{Imp}(C)$.

Proof of Claim 2: By Conditions $S1$ and $S2$, $\bot \notin \mathsf{Imp}(A_0)$ and $\bot \notin \mathsf{Imp}(\{a\})$ for all $\{a\} \in \mathsf{P}^{\mathcal{C}}$. Due to Rule **CR5** and by definition of $\mathsf{P}^{\mathcal{C}}_{-}$, $\bot \notin \mathsf{Imp}(C)$. Thus, by Rule **CR7** there exists a solution for $\mathsf{con}_i(\mathsf{Imp}(C))$. It remains to show that this solution can be chosen not to satisfy any concept $p(f_1, \dots, f_k) \in \mathsf{P}^{\mathcal{C}} \setminus \mathsf{Imp}(C)$. Let Γ be the set of all solutions for $\mathsf{con}_i(\mathsf{Imp}(C))$. Moreover, assume on the contrary that there exists a set $\Psi \subseteq \mathsf{P}^{\mathcal{C}} \setminus \mathsf{Imp}(C)$ of concepts of the form $p(f_1, \dots, f_k)$ with $p \in \mathcal{P}^{\mathcal{D}_i}$ such that each solution from Γ satisfies a concept from Ψ, i.e., $\mathsf{con}_i(\mathsf{Imp}(C))$ implies the disjunction of all concepts in Ψ. By Property 2 of p-admissibility, $\mathsf{con}_i(\mathsf{Imp}(C))$ implies a single concept X from Ψ. By Rule **CR8**, this implies $X \in \mathsf{Imp}(C)$, in contradiction to $X \in \Psi$.

This finishes the proof of Claim 2. For each $C \in \mathsf{P}^{\mathcal{C}}_{-}$ and each $i \in \{1, \dots, n\}$, fix a solution $\delta([C], i)$ for $\mathsf{con}_i(\mathsf{Imp}(C))$ as in Claim 2. We now define an interpretation $\mathcal{I}$ as follows:

$$\Delta^{\mathcal{I}} := \{[C] \mid C \in \mathsf{P}^{\mathcal{C}}_{-}\}$$

$$A^{\mathcal{I}} := \begin{cases} \{[C] \in \Delta^{\mathcal{I}} \mid A \in \mathsf{Imp}(C)\} & \text{if } A \in \mathsf{N}^{\mathcal{C}}_{\mathrm{con}} \\ \emptyset & \text{otherwise} \end{cases}$$

$$a^{\mathcal{I}} := \begin{cases} [\{a\}] & \text{if } \{a\} \in \mathsf{P}^{\mathcal{C}} \\ [A_0] & \text{otherwise} \end{cases}$$

$$r^{\mathcal{I}} := \begin{cases} \{([C], [D]) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \exists D' \in [D] \colon (C, D') \in \mathsf{Imp}(r)\} & \text{if } r \in \mathsf{N}^{\mathcal{C}}_{\mathrm{role}} \\ \emptyset & \text{otherwise} \end{cases}$$

$$f^{\mathcal{I}}([C]) := \delta([C], i) \text{ if there exists a } p(f_1, \dots, f_m) \in \mathsf{Imp}(C) \text{ with } p \in \mathcal{P}^{\Delta_i}$$
$$\text{and } f_j = f \text{ for some } j \in \{1, \dots, m\} \text{ for all } f \in \mathsf{N}^{\mathcal{C}}_{\mathrm{fe}} \text{ and } [C] \in \Delta^{\mathcal{I}}.$$

Note that $a^{\mathcal{I}}$ is well-defined for each $a \in \mathsf{N}_{\mathrm{nom}}$ since equivalence classes w.r.t. $\sim$ are used. Moreover, the assignment $a^{\mathcal{I}} := [A_0]$ for every $\{a\} \notin \mathsf{N}^{C}_{\mathrm{con}}$ is arbitrary. Note also that $r^{\mathcal{I}}$ is well-defined for every $r \in \mathsf{N}_{\mathrm{role}}$ because of Claim 1(b), and $f^{\mathcal{I}}$ is well-defined for every $f \in \mathsf{N}_{\mathrm{fe}}$ since $\bot \notin \mathsf{Imp}(C)$ for all $C \in \mathsf{P}^{\mathcal{C}}_{-}$ and due to **CR9**. We now establish an additional, central claim.

Claim 3. For all $[C] \in \Delta^{\mathcal{I}}$ and $D \in \mathsf{P}^{\mathcal{C}} \cup \{\bot\}$, $[C] \in D^{\mathcal{I}}$ iff $D \in \mathsf{Imp}(C)$.

The proof distinguishes four cases depending on the form of D:

- $D = \top$. Trivial since $\top \in \mathsf{Imp}(C)$ for all $C \in \mathsf{P}^{\mathcal{C}}_{-}$.

- $D = \bot$. Trivial since, as shown in the proof of Claim 2, $\bot \notin \mathsf{Imp}(C)$ for all $C \in \mathsf{P}_-^{\mathcal{C}}$.
- $D \in \mathsf{N}_{\mathrm{con}}^{\mathcal{C}}$. Then, $[C] \in D^{\mathcal{I}}$ iff $D \in \mathsf{Imp}(C)$ immediately by definition of $\mathcal{I}$.
- $D = \{a\}$ for some $a \in \mathsf{N}_{\mathrm{nom}}^{\mathcal{C}}$. Then $[C] \in \{a\}^{\mathcal{I}}$ implies $a^{\mathcal{I}} = [C]$ and thus $[C] = [\{a\}]$ by definition of $\{a\}^{\mathcal{I}}$. This yields $\{a\} \in \mathsf{Imp}(C)$ since $\{a\} \in \mathsf{Imp}_0(\{a\})$. Conversely, $\{a\} \in \mathsf{Imp}(C)$ implies $[C] = [\{a\}]$ by definition of $\sim$. Hence, $a^{\mathcal{I}} = [C]$ implying $[C] \in \{a\}^{\mathcal{I}}$ by definition of $\mathcal{I}$ and the semantics.
- $D = p(f_1, \ldots, f_k)$ with $p \in \mathcal{P}^{\mathcal{D}_i}$ for some i. Then $[C] \in D^{\mathcal{I}}$ iff $\delta([C], i) \models D$ iff $D \in \mathsf{Imp}(C)$. The first 'iff' is by definition of $\mathcal{I}$ and the semantics and the latter by choice of $\delta([C], i)$.

This finishes the proof of Claim 3. We now show that $\mathcal{I}$ is a model of $\mathcal{C}$ with $x \in A_0^{\mathcal{I}} \setminus B_0^{\mathcal{I}}$ for some $x \in \Delta^{\mathcal{I}}$. Since $A_0 \in \mathsf{P}_-^{\mathcal{C}}$, by definition of $\mathsf{P}_-^{\mathcal{C}}$, $[A_0] \in \Delta^{\mathcal{I}}$. By $S1$, $B_0 \notin \mathsf{Imp}(A_0)$. Due to the definition of Imp_0, $A_0 \in \mathsf{Imp}(A_0)$, implying by Claim 3 that $[A_0] \in A_0^{\mathcal{I}} \setminus B_0^{\mathcal{I}}$. It remains to show that $\mathcal{I}$ is a model of $\mathcal{C}$. To this end, we distinguish four types of GCIs and two types of RIs in $\mathcal{C}$.

- $C \sqsubseteq D$. Let $[C'] \in C^{\mathcal{I}}$. By Claim 3, $C \in \mathsf{Imp}(C')$. Due to Rule **CR1**, this implies $D \in \mathsf{Imp}(C')$ and thus $[C'] \in D^{\mathcal{I}}$ by Claim 3.
- $C \sqcap D \sqsubseteq E$. Similar to the previous case using Rule **CR2**.
- $C \sqsubseteq \exists r.D$. Let $[C'] \in C^{\mathcal{I}}$. Then $C \in \mathsf{Imp}(C')$ by Claim 3. Thus, by Rule **CR3**, $(C', D) \in \mathsf{Imp}(r)$. By definition of $r^{\mathcal{I}}$, this implies $([C'], [D]) \in r^{\mathcal{I}}$. Moreover, $D \in \mathsf{Imp}_0(D)$ implies $D \in \mathsf{Imp}(D)$. Thus, Claim 3 yields $[D] \in D^{\mathcal{I}}$. Together, this yields $[C'] \in (\exists r.D)^{\mathcal{I}}$ as required.
- $\exists r.C \sqsubseteq D$. Let $[E] \in (\exists r.C)^{\mathcal{I}}$. Hence, there is an $[F] \in \Delta^{\mathcal{I}}$ such that $([E], [F]) \in r^{\mathcal{I}}$ and $[F] \in C^{\mathcal{I}}$. Thus, by definition of $\mathcal{I}$, there is some $F' \in [F]$ with $(E, F') \in \mathsf{Imp}(r)$. Moreover, $[F'] = [F] \in C^{\mathcal{I}}$ implies $C \in \mathsf{Imp}(F')$ by Claim 3. By Rule **CR4**, $D \in \mathsf{Imp}(E)$, implying $[E] \in D^{\mathcal{I}}$ by Claim 3 as required.
- $r \sqsubseteq s$. Let $([C], [D]) \in r^{\mathcal{I}}$. Then there is a $D' \in [D]$ such that $(C, D') \in \mathsf{Imp}(r)$, implying $(C, D') \in \mathsf{Imp}(s)$ by **CR10**. Thus, by definition of $\mathcal{I}$, $([C], [D']) = ([C], [D]) \in s^{\mathcal{I}}$ as required.
- $r_1 \circ r_2 \sqsubseteq s$. Let $([C], [D]) \in r_1^{\mathcal{I}}$ and $([D], [E]) \in r_2^{\mathcal{I}}$. Then there are $D' \in [D]$ and $E' \in [E]$ with $(C, D') \in \mathsf{Imp}(r)$ and $(D, E') \in \mathsf{Imp}(r)$. By Claim 1(b), the latter yields $(D', E') \in \mathsf{Imp}(r)$. Hence, $(C, E') \in R(s)$ by **CR10**. By definition of $\mathcal{I}$, $([C], [E']) = ([C], [E]) \in s^{\mathcal{I}}$ as required.

□

Consequently, implication sets can be used to decide subsumption w.r.t. $\mathcal{EL}^{++}$-CBoxes. We still have to show that the mapping Imp, i.e., a complete set of implication sets, can be computed in polynomial time.

Lemma 3.2.6 *Let $\mathcal{C}$ be a normalized CBox. Then the rules in Table 3.2.1 can only be applied a polynomial number of times, and each rule application takes only polynomial time.*

PROOF. The cardinality of $\mathsf{P}^{\mathcal{C}}$ and $\mathsf{N}_{\mathrm{role}}^{\mathcal{C}}$ is bounded by $|\mathcal{C}|$. Applying one rule from Table 3.2.1 either adds some $P \in \mathsf{P}^{\mathcal{C}} \cup \{\bot\}$ to $\mathsf{Imp}(C)$ for some $C \in \mathsf{P}^{C}$, or adds a new

tuple $(C, D) \in \mathsf{P}^{\mathcal{C}} \times \mathsf{P}^{\mathcal{C}}$ to $\mathsf{Imp}(r)$ for some $r \in \mathsf{N}^{\mathcal{C}}_{\text{role}}$. Since no elements are removed from any implication set, the total number of rule applications is polynomial. It is easy to check that each rule application can be performed in polynomial time. In particular, note that the relation $\rightsquigarrow$ can be computed using polynomial time graph reachability. □

We obtain the following result as a consequence of Lemmas 3.2.2, 3.2.6, and 3.2.4, and the reduction of satisfiability, consistency, and the instance problem to subsumption shown in Section 2.1.

Theorem 3.2.7 *Satisfiability, subsumption, ABox consistency, and the instance problem in $\mathcal{EL}^{++}(\mathcal{D}_1, \ldots, \mathcal{D}_n)$ can be decided in polynomial time.*

Taken together, the proofs of Lemma 3.2.4 and 3.2.5 yield a small model property for $\mathcal{EL}^{++}$. Taking into account the reductions of satisfiability and ABox consistency to subsumption from Section 2.1.1, we obtain the following.

Theorem 3.2.8 *Let C and D be concepts, $\mathcal{A}$ an ABox, and $\mathcal{C}$ a CBox. Then:*

1. *if C is satisfiable w.r.t. $\mathcal{C}$, then C and $\mathcal{C}$ have a common model of size linear in $|C| + |\mathcal{C}|$;*
2. *if C is not subsumed by D w.r.t. $\mathcal{C}$, then there exists a model $\mathcal{I}$ of $\mathcal{C}$ of size linear in $|C| + |D| + |\mathcal{C}|$ such that $x \in C^{\mathcal{I}} \setminus D^{\mathcal{I}}$ for some $x \in \Delta^{\mathcal{I}}$;*
3. *If $\mathcal{A}$ is consistent w.r.t. $\mathcal{C}$, then $\mathcal{A}$ and $\mathcal{C}$ have a common model of size linear in $|\mathcal{A}| + |\mathcal{C}|$;*
4. *if an individual a is not an instance of C in $\mathcal{A}$ w.r.t. $\mathcal{C}$, then there exists a model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{C}$ of size linear in $|C| + |\mathcal{A}| + |\mathcal{C}|$ such that $a^{\mathcal{I}} \notin C^{\mathcal{I}}$.*

In order to complete the picture of $\mathcal{EL}^{++}$-CBoxes, it remains to find useful concrete domains matching our definition of p-admissibility.

3.2.3 P-admissible concrete domains

In order to obtain concrete DLs of the form $\mathcal{EL}^{++}(\mathcal{D}_1, \ldots, \mathcal{D}_n)$ for $n > 0$ to which Theorem 3.2.7 applies, p-admissible concrete domains are needed. In the following, we introduce two p-admissible concrete domains, and show that small extensions of them are no longer p-admissible. To simplify notation, we call every finite conjunction of atomic formulae $p(f_1, \ldots, f_k)$ from a concrete domain $\mathcal{D}$ a *$\mathcal{D}$-conjunction*.

Definition 3.2.9 (Concrete domains $\mathcal{Q}$ and $\mathcal{S}$)
The concrete domain $\mathcal{Q} = (\mathbb{Q}, \mathcal{P}^{\mathcal{Q}})$ has as its domain the set $\mathbb{Q}$ of rational numbers, and its set of predicates $\mathcal{P}^{\mathcal{Q}}$ consists of the following predicates:

- a unary predicate $\top_{\mathcal{Q}}$ with $(\top_{\mathcal{Q}})^{\mathcal{Q}} = \mathbb{Q}$;
- unary predicates $=_q$ and $>_q$ for each $q \in \mathbb{Q}$;
- a binary predicate $=$;
- a binary predicate $+_q$, for each $q \in \mathbb{Q}$, with $(+_q)^{\mathcal{Q}} = \{(q', q'') \in \mathbb{Q}^2 \mid q' + q = q''\}$.

The concrete domain $\mathcal{S}$ is defined as $(\Sigma^*, \mathcal{P}^{\mathcal{S}})$, where Σ is the ISO 8859-1 (Latin-1) character set and $\mathcal{P}^{\mathcal{S}}$ consists of the following predicates:

- a unary predicate $\top_{\mathcal{S}}$ with $(\top_{\mathcal{S}})^{\mathcal{S}} = \Sigma^*$;
- a unary predicate $=_w$, for each $w \in \Sigma^*$;
- a binary predicate $=$;
- a binary predicate conc_w, for each $w \in \Sigma^*$, with $conc_w^{\mathcal{Q}} = \{(w', w'') \mid w'' = w'w\}$. □

We now show that both $\mathcal{Q}$ and $\mathcal{S}$ are p-admissible. By Definition 2.3.3, satisfiability and implication in $\mathcal{Q}$ and $\mathcal{S}$ must be decidable in polynomial time (Condition 1) and $\mathcal{Q}$ and $\mathcal{S}$ must be convex (Condition 2).

Proposition 3.2.10 *The concrete domain $\mathcal{Q}$ is p-admissible.*

PROOF. First for Condition 1 of p-admissibility. Assume that, in $\mathcal{Q}$-conjunctions, we admit the following additional predicates:

- a unary predicate $<_q$ for each $q \in \mathbb{Q}$ with $(P_<)^{\mathcal{Q}} = \{q' \in \mathbb{Q} \mid q' < q\}$;
- a binary predicate $<$ with the obvious extension.

In this extended set of predicates, $\mathcal{Q}$-implication can be reduced to $\mathcal{Q}$-satisfiability: assume that we want to decide whether the $\mathcal{Q}$-conjunction c implies a formula $p(f_1, \ldots, f_k)$ with $p \in \mathcal{P}^{\mathcal{Q}}$. We make a case distinction according to p:

$=_q$ the implication holds if neither $c \wedge <_q(f_1)$ nor $c \wedge >_q(f_1)$ is satisfiable;

$>_q$ the implication holds if neither $c \wedge <_q(f_1)$ nor $c \wedge =_q(f_1)$ is satisfiable;

$=$ the implication holds if neither $c \wedge <(f_1, f_2)$ nor $c \wedge <(f_2, f_1)$ is satisfiable;

$+_q$ the implication holds if neither $c \wedge +_q(f_1, f) \wedge <(f, f_2)$ nor $c \wedge +_q(f_1, f) \wedge <(f_2, f)$ is satisfiable, where f is a feature name not appearing in c.

By reduction to linear programming, it is shown in [Lut03] that satisfiability of $\mathcal{Q}$-conjunctions over the extended set of predicates is decidable in polynomial time.

Now for Condition 2 of p-admissibility. Let c be a $\mathcal{Q}$-conjunction, and let Γ be a finite set of formulae of the form $p(f_1, \ldots, f_k)$ such that c implies no formula from Γ. Obviously, c is satisfiable. Assume that c implies the disjunction over all formulae in Γ. W.l.o.g., assume that c does not contain conjuncts of the form $\top_{\mathcal{Q}}(f)$ since the conjunction obtained from c by removing such conjuncts is equivalent to c. Moreover, the fact that c does not imply any formula from Γ means that Γ also contains no predicates of the form $\top_{\mathcal{Q}}(f)$. Our aim is to construct a solution δ for c such that $\delta \not\models C$ for all $C \in \Gamma$, in contradiction to our assumption. To this end, we first define a solution for c that does not satisfy any formula $>_q(f) \in \Gamma$, and then modify this solution such that no other formulae from Γ are satisfied.

For the first step, start by defining a relation $\sim$ on the set of features N_{fe} as follows:

$$f \sim f' \text{ iff } f = f' \text{ or } f \text{ and } f' \text{ occur jointly in a conjunct of } c.$$

Clearly, the transitive closure $\sim^*$ of $\sim$ is an equivalence relation. For each equivalence class Δ of $\sim^*$, we define a *distance function* d_Δ that maps each pair of features $f, f' \in \Delta$

to a rational number:

$$\begin{aligned}
&d_\Delta(f,f) := 0 && \\
&d_\Delta(f,f') := 0 && \text{if } =(f,f') \in c \text{ or } =(f',f) \in c \\
&d_\Delta(f,f') := q && \text{if } +_q(f,f') \in c \\
&d_\Delta(f,f') := -q && \text{if } +_q(f',f) \in c \\
&d_\Delta(f,f') := d_\Delta(f,f'') + d_\Delta(f'',f'). &&
\end{aligned}$$

Note that d_Δ is total on Δ due to the definition of $\sim$ and well-defined since c is satisfiable. We call a feature f *fixed by* c iff there exists a feature f' with $f \sim^* f'$ and $=_q(f') \in c$ for some $q \in \mathbb{Q}$. For a given $\sim^*$-equivalence class, either all contained features are fixed or all are not fixed. In this sense, entire equivalence classes can be called *(not) fixed*.

Let $\Delta_1, \ldots, \Delta_k$ be the equivalence classes of $\sim^*$. We define a solution δ_0 for c. This is done separately for each Δ_i with $i \in \{1, \ldots, k\}$.

1. If Δ_i is fixed then take a feature $f \in \Delta_i$ with $=_q(f) \in c$ and let $\delta_0(f) := q$. For all other features $f' \in \Delta_i$, let $\delta_0(f') := \delta_0(f) + d_{\Delta_i}(f,f')$.

2. If Δ_i is not fixed then choose a feature $f \in \Delta_i$. Then choose a value $\delta_0(f) \in \mathbb{Q}$ such that two conditions hold:

   - $\delta_0(f) + d_{\Delta_i}(f,f') > q$ for all $f' \in \Delta_i$ and all q with $>_q(f') \in c$; and
   - $\delta_0(f) + d_{\Delta_i}(f,f') \leq q$ for all $f' \in \Delta_i$ and all q with $>_q(f') \in \Gamma$.

   For all other $f' \in \Delta_i$, let $\delta_0(f') := \delta_0(f) + d_{\Delta_i}(f,f')$.

   To verify that a rational number $\delta_0(f)$ as required above always exists, assume the contrary. Then there is a $>_q(f') \in c$ and a $>_{q'}(f'') \in \Gamma$ with $d_{\Delta_i}(f,f') - d_{\Delta_i}(f,f'') = d_{\Delta_i}(f',f'') \geq q - q'$. Hence, by definition of d_{Δ_i}, there is no solution δ_0 for c that does not satisfy $>_{q'}(f'') \in \Gamma$, in contradiction to the fact that c does not imply any element of Γ.

By definition of d_Δ and of δ_0, δ_0 is a solution for c satisfying none of the formulae $>_q(f)$ in Γ. The latter is obvious if Δ_i is not fixed (Case 2 above). If Δ_i is fixed (Case 1) then $\delta(f) > q$ clearly yields that c implies $>_q(f)$, and thus $>_q(f) \notin \Gamma$.

Now for the second step which deals with formulae $=_q(f)$, $=(f,f')$, and $+_q(f,f')$ in Γ that may be satisfied 'accidentally' by δ_0. We destroy such satisfactions by 'shifting down' values of δ_0. To this end, choose $b \in \mathbb{Q}$ such that the following conditions are satisfied:

1. $b > 0$;
2. for all conjuncts $>_q(f)$ of c, $b < \delta_0(f) - q$;
3. for all $=_q(f) \in \Gamma$ with $\delta_0(f) \neq q$, $b < |\delta_0(f) - q|$;
4. for all $=(f,f') \in \Gamma$ with $\delta_0(f) \neq \delta_0(f')$, $b < |\delta_0(f) - \delta_0(f')|$; and
5. for all $+_q(f,f') \in \Gamma$ with $\delta_0(f') \neq \delta_0(f) + q$, $b < |\delta_0(f') - (\delta_0(f) + q)|$.

Define a new solution δ of c as follows:

$$\delta(f) := \begin{cases} \delta_0(f) - b & \text{if } f \text{ is not fixed by } c \\ \delta_0(f) & \text{otherwise.} \end{cases}$$

It is easy to see that δ solves c: conjuncts $>_q(f)$ are satisfied by choice of b (Point 2); conjuncts $=_q(f)$ are satisfied since they are satisfied by δ_0 and their presence implies that f is fixed by c; and conjuncts $=(f, f')$ and $+_q(f, f')$ are satisfied since they are satisfied by δ_0 and their presence implies that f is fixed by c iff f' is fixed by c.

Moreover, the new solution δ does not satisfy any formula in Γ: formulae $>_q(f)$ have not been satisfied by δ_0, and we only shifted *down* when moving to δ; formulae of the other form are not satisfied by definition of δ and choice of b. □

It remains to show that the second concrete domain from Definition 3.2.9 is also p-admissible.

Proposition 3.2.11 *The concrete domain $\mathcal{S}$ is p-admissible.*

PROOF. Condition 1 of p-admissibility: consider the concrete domain $\mathcal{S}' = (\Sigma^*, \mathcal{P}^{\mathcal{S}'})$, with $\mathcal{P}^{\mathcal{S}'}$ containing the following predicates:

1. a unary predicate $\top_{\mathcal{S}}$ as in $\mathcal{S}$;
2. a unary predicate $=_\varepsilon$ as in $\mathcal{S}$, but only for the empty word, and its negation $\neq_\varepsilon$ with the obvious extension;
3. binary predicates $=$ and $\neq$ with the obvious extensions;
4. binary predicates conc_w and $\overline{\mathsf{conc}}_w$ for each $w \in \Sigma^*$, where the extension of conc_w is as in $\mathcal{S}$, and the extension of $\overline{\mathsf{conc}}_w$ is complementary.

We claim that satisfiability and implication in $\mathcal{S}$ can be polynomially reduced to satisfiability in $\mathcal{S}'$:

- To check satisfiability of an $\mathcal{S}$-conjunction c, first extend c with the conjunct $=_\varepsilon (e)$, where e is a feature name not occurring in c, and then replace each conjunct $=_w(f)$ in c with $w \neq \varepsilon$ by the conjunct $\mathsf{conc}_w(e, f)$. Finally, check satisfiability of the resulting conjunction c' in $\mathcal{S}'$.
- To check whether an $\mathcal{S}$-conjunction c implies a formula $p(f_1, \ldots, f_n)$, first transform c into c' as in the satisfiability case above. If p is of the form $=_\varepsilon$, $=$, or conc_w, then simply check whether c' extended by the conjunct $\overline{p}(f_1, \ldots, f_n)$ is unsatisfiable. If p is of the form $=_w$ with $w \neq \varepsilon$ then check whether c' extended by the conjunct $\mathsf{conc}_w(e, f_1)$ is unsatisfiable.

It has been shown in [Lut01] that satisfiability in $\mathcal{S}'$ is decidable in polynomial time, which by the above reduction carries over to satisfiability and implication in $\mathcal{S}$.

Condition 2 of p-admissibility: first, let c be an $\mathcal{S}$-conjunction and Γ a finite set of formulae of the form $p(f_1, \ldots, f_k)$ such that c implies no formula from Γ. Again, in this case c is satisfiable. Assume that c implies the disjunction over all formulae in Γ. As in the case of the concrete domain $\mathcal{Q}$, we may assume that the predicate $\top_{\mathcal{S}}(f)$ does not occur in c and Γ. Our aim is to construct a solution δ for c such that $\delta \not\models C$ for all $C \in \Gamma$, in contradiction to the assumption.

To this end, let δ_0 be an arbitrary solution for c. We modify this solution such that no formula from Γ is satisfied. We begin by defining a relation $\sim$ on the set N_{fe} of features as follows:

$$f \sim f' \text{ iff } f = f' \text{ or } f \text{ and } f' \text{ occur jointly in a conjunct of } c.$$

The transitive closure $\sim^*$ of $\sim$ is an equivalence relation. Analogous to the case of $\mathcal{Q}$, we call a feature f *fixed by* c iff there exists a feature f' with $f \sim^* f'$ and $=_w(f') \in c$ for some $w \in \Sigma^*$. Again, this gives rise to the notion of *(not) fixed* equivalence classes. Denote by $\alpha_1, \dots, \alpha_n$ the not fixed equivalence classes of $\sim^*$. Then choose words $w_1, \dots, w_n \in \Sigma^*$ with the following properties.

1. for every $i \in \{1, \dots, n\}$, w_i is no prefix of w, and $=_w$ is a predicate occurring in Γ;
2. for every $i, j \in \{1, \dots, n\}$ with $i \neq j$, w_i is no prefix of w_j; and
3. for every $i \in \{1, \dots, n\}$ and every $f \in \mathsf{N}_{\mathrm{fe}}$ occurring in c, w_i is no prefix of $\delta(f)$.

Now define a new solution δ of c by:

$$\delta(f) := \begin{cases} w_i \cdot \delta_0(f) & \text{if } f \in \alpha_i \\ \delta_0(f) & \text{if there is no such } \alpha_i \ . \end{cases}$$

It remains to show that $\delta \models c$ and $\delta \not\models C$ for all $C \in \Gamma$. For the former, we argue as follows, distinguishing three cases for the different types of predicates occurring in c:

- For all predicates in c of the form $=_w(f)$, f is not fixed by definition, implying $\delta(f) = \delta_0(f) = w$.
- All predicates in c of the form $=(f, g)$ remain satisfied because their existence implies that f and g are in the same equivalence class and thus $\delta(f) = w \cdot \delta_0(f)$ and $\delta(g) = w \cdot \delta_0(g)$ for some $w \in \Sigma^*$.
- All predicates $\mathsf{conc}_w(f, g)$ remain satisfied for the same reason.

Now for the latter, i.e., $\delta \not\models C$. Again, we distinguish three cases.

- Consider some $=_w(f) \in \Gamma$. If f is fixed and $\delta_0 \models =_w(f)$ then c implies $=_w(f)$, in contradiction to the assumption that c does not imply any element of Γ. Thus, either f is not fixed or $\delta_0 \not\models =_w(f)$. If f is not fixed then $\delta \not\models =_w(f)$ because of Property 1 of the words $w_1, \dots, w_n$. If f is fixed and $\delta_0 \not\models =_w(f)$ then clearly also $\delta \not\models =_w(f)$.
- Now consider $=(f, g) \in \Gamma$. If f and g are in the same equivalence class and $\delta_0 \models =(f, g)$ then c implies $=(f, g)$, contradicting our assumption. Thus, either f and g are not in the same equivalence class or $\delta_0 \not\models =(f, g)$. In f and g are not in the same equivalence class, $\delta \not\models =(f, g)$ due to Properties 2 and 3 of the words $w_1, \dots, w_n$. If they are in the same equivalence class and $\delta_0 \not\models =(f, g)$ then clearly also $\delta \not\models =(f, g)$.
- The case $\mathsf{conc}_w(f, g) \in \Gamma$ is analogous. □

It has been shown in Section 3.1.3 that p-admissibility of a concrete domain $\mathcal{D}$ is a necessary condition for polynomial-time reasoning w.r.t. $\mathcal{EL}(\mathcal{D})$-IBoxes. P-admissibility, however, is a fragile property, as the following examples suggest:

- Consider the concrete domain $\mathcal{Q}^{\leq_q, >_q}$ with domain $\mathbb{Q}$ that has the predicates $(>_q)_{q \in \mathbb{Q}}$ from $\mathcal{Q}$ and, additionally, unary predicates $(\leq_q)_{q \in \mathbb{Q}}$ with

  $$(\leq_q)^{\mathcal{Q}^{\leq_q, >_q}} := \{q' \in \mathbb{Q} \mid q' \leq q\}.$$

  Then the $\mathcal{Q}^{\leq_q, >_q}$-conjunction $c := >_0(f')$ does not imply any formula of $\Gamma := \{\leq_0(f), >_0(f)\}$, but every solution of c satisfies some formula of Γ.

- Any concrete domain $\mathcal{S}^*$ with domain Σ^* for some finite alphabet Σ and the unary predicates pref_w and suff_w for every $w \in \Sigma^*$ with

$$\mathsf{pref}_w^{\mathcal{S}^*} := \{w' \mid w \text{ is a prefix of } w'\}$$
$$\mathsf{suff}_w^{\mathcal{S}^*} := \{w' \mid w \text{ is a suffix of } w'\}.$$

  Assume $a \in \Sigma$. Then the $\mathcal{S}^*$-conjunction $c := \mathsf{suff}_a(f)$ implies no formula from $\Gamma := \{\mathsf{pref}_\sigma(f) \mid \sigma \in \Sigma\}$, but every solution of c satisfies some formula from Γ.

- Any concrete domain $\mathcal{S}^*$ with domain Σ^* for some finite alphabet Σ, the unary predicates $\top_{\mathcal{S}^*}$ and $=_\varepsilon$ with the obvious semantics, and the unary predicates pref_w, $w \in \Sigma^*$, as in the previous example. Then the $\mathcal{S}^*$-conjunction $c := \top_{\mathcal{S}^*}(f)$ implies no formula from $\Gamma := \{=_\varepsilon(f)\} \cup \{\mathsf{pref}_\sigma(f) \mid \sigma \in \Sigma\}$, but every solution of c satisfies some formula from Γ.

3.2.4 Comparison to $\mathcal{FL}_0$

In the previous sections, we have shown that reasoning w.r.t. $\mathcal{EL}^{++}(\mathcal{D}_1, \ldots, \mathcal{D}_n)$-CBoxes is tractable for p-admissible concrete domains $\mathcal{D}_1, \ldots, \mathcal{D}_n$ and that non-trivial concrete domains fall into this class. In order to support our notion that this tractability result is relatively surprising, we compare the computational complexity of $\mathcal{EL}$ with that of its sibling DL $\mathcal{FL}_0$. Though seemingly equally harmless as $\mathcal{EL}$, $\mathcal{FL}_0$ is far less robust than $\mathcal{EL}$ w.r.t. the addition of TBox formalisms. Summing up the results on $\mathcal{EL}$ obtained in this and previous works, we obtain the following picture:

- Deciding subsumption for $\mathcal{EL}$ concepts without TBoxes is tractable [BKM99].
- Deciding subsumption in $\mathcal{EL}$ w.r.t. *cyclic TBoxes* is still tractable [Baa03b]. Note that general TBoxes are excluded.
- Deciding subsumption in $\mathcal{EL}$ plus role hierarchies w.r.t. *general* TBoxes is still tractable [Bra04b].
- Deciding subsumption remains tractable even w.r.t. $\mathcal{EL}^{++}$-CBoxes extended by p-admissible concrete domains, i.e., even for several extensions of $\mathcal{EL}$ both w.r.t. language constructors and TBox formalisms [BBL05].

Summarizing the work on $\mathcal{FL}_0$ produces a very different picture. Adding a more powerful TBox formalism usually results in an increase of the complexity of reasoning:

- Deciding subsumption of $\mathcal{FL}_0$ concepts without TBoxes is tractable [BL84].
- Deciding subsumption in $\mathcal{FL}_0$ w.r.t. acyclic TBoxes is co-NP-complete [Neb90].
- Deciding subsumption in $\mathcal{FL}_0$ w.r.t. cyclic TBoxes is PSPACE-complete [Baa90, Baa96, KN03].

In order to complete the picture for $\mathcal{FL}_0$ and to illustrate the robustness of the computational behavior of $\mathcal{EL}$, we prove that subsumption in $\mathcal{FL}_0$ w.r.t. general TBoxes is EXPTIME-complete. As containment in EXPTIME follows from the fact that subsumption in $\mathcal{ALC}$ w.r.t. general TBoxes is in EXPTIME, it remains to establish the lower bound. The proof is by reduction of subsumption in $\mathcal{FL}_0^{\mathrm{tf}}$ w.r.t. general TBoxes which is EXPTIME-hard by Theorem 3.1.26.

Lemma 3.2.12 *Let C_0, D_0 be $\mathcal{FL}_0^{\text{tf}}$-concepts and $\mathcal{T}$ a general $\mathcal{FL}_0^{\text{tf}}$-TBox. Then $C_0 \sqsubseteq_{\mathcal{T}} D_0$ in $\mathcal{FL}_0$ iff $C_0 \sqsubseteq_{\mathcal{T}} D_0$ in $\mathcal{FL}_0^{\text{tf}}$.*

PROOF. W.l.o.g., assume that C_0, D_0 are concept names and that $\mathcal{T}$ is in normal form, i.e., only contains concept definitions of the following forms:

$$\begin{aligned} A &\sqsubseteq B \\ A &\sqsubseteq B \sqcap B' \\ A &\sqsubseteq \forall r.B \\ A \sqcap A' &\sqsubseteq B \\ \forall r.A &\sqsubseteq B, \end{aligned}$$

where A, A', B, B' are concept names. It is easy to see that every general $\mathcal{FL}_0^{\text{tf}}$-TBox can be converted into normal form in polynomial time by normalization rules similar to those in Figure 3.2.1.

($\Rightarrow$) Proof by contraposition. $C_0 \not\sqsubseteq_{\mathcal{T}} D_0$ in $\mathcal{FL}_0^{\text{tf}}$ implies $C_0 \not\sqsubseteq_{\mathcal{T}} D_0$ in $\mathcal{FL}_0$. As every interpretation witnessing the former is also a witness for the latter, the claim holds trivially.

($\Leftarrow$) Assume that $C_0 \not\sqsubseteq_{\mathcal{T}} D_0$, i.e., there is an $\mathcal{FL}_0$ model $\mathcal{I}$ of $\mathcal{T}$ and an $a_0 \in C_0^{\mathcal{I}} \setminus D_0^{\mathcal{I}}$. Our aim is to convert $\mathcal{I}$ into an $\mathcal{FL}_0^{\text{tf}}$ interpretation $\mathcal{J}$ witnessing $C_0 \not\sqsubseteq_{\mathcal{T}} D_0$ in $\mathcal{FL}_0^{\text{tf}}$. Denote by $\mathcal{S}$ the set of all sequences of role names from $\mathsf{N}_{\text{role}}^{\mathcal{T}}$, including the empty sequence ε. For every $S \in \mathcal{S}$ and $a \in \Delta^{\mathcal{I}}$, let $S^{\mathcal{I}}(a) := \{b \in \Delta^{\mathcal{I}} \mid (a,b) \in S^{\mathcal{I}}\}$, where $S^{\mathcal{I}}$ is defined in the obvious way using composition of relations and $\varepsilon^{\mathcal{I}}(a) = \{a\}$. Now we construct $\mathcal{J}$ as follows:

$$\begin{aligned} \Delta^{\mathcal{J}} &:= \mathcal{S} \\ A^{\mathcal{J}} &:= \{S \mid S^{\mathcal{I}}(a_0) \subseteq A^{\mathcal{I}}\} \\ r^{\mathcal{J}} &:= \{(S, S') \mid S' = Sr\} \quad \text{for all } r \in \mathsf{N}_{\text{role}}^{\mathcal{T}} \end{aligned}$$

By definition of $\mathcal{J}$, $\varepsilon \in C_0^{\mathcal{J}} \setminus D_0^{\mathcal{J}}$. It is readily checked that all role names are interpreted as total functions. To show $C_0 \not\sqsubseteq_{\mathcal{T}} D_0$, it thus remains to show that $\mathcal{J}$ satisfies all concept inclusions in $\mathcal{T}$:

1. $A \sqsubseteq B$. Let $S \in A^{\mathcal{J}}$. Then $S^{\mathcal{I}}(a_0) \subseteq A^{\mathcal{I}}$. Since $\mathcal{I}$ satisfies $A \equiv B$, $S^{\mathcal{I}}(a_0) \subseteq B^{\mathcal{I}}$ and thus $S \in B^{\mathcal{J}}$ as required.
2. $A \sqcap A' \sqsubseteq B$. Similar to the previous case.
3. $\forall r.A \sqsubseteq B$. Let $S \in (\forall r.A)^{\mathcal{J}}$. Then $Sr \in A^{\mathcal{J}}$. Thus, $Sr^{\mathcal{I}}(a_0) \subseteq A^{\mathcal{I}}$, implying $S^{\mathcal{I}}(a_0) \subseteq (\forall r.A)^{\mathcal{I}}$. Hence, $S^{\mathcal{I}}(a_0) \subseteq B^{\mathcal{I}}$ since $\mathcal{I}$ satisfies $\forall r.A \sqsubseteq B$. It follows that $S \in B^{\mathcal{J}}$ as required.
4. $A \sqsubseteq \forall r.B$. Let $S \in A^{\mathcal{J}}$. Then $S^{\mathcal{I}}(a_0) \subseteq A^{\mathcal{I}}$, and, since $\mathcal{I}$ satisfies $A \sqsubseteq \forall r.B$, $Sr^{\mathcal{I}}(a_0) \subseteq B^{\mathcal{I}}$. This implies $Sr \in B^{\mathcal{J}}$, yielding $S \in (\forall r.B)^{\mathcal{J}}$ since Sr is the only r-successor of S in $\mathcal{J}$. □

The following theorem is an immediate consequence of Theorem 3.1.26 and Lemma 3.2.12.

Theorem 3.2.13 *Subsumption in $\mathcal{FL}_0$ w.r.t. general TBoxes is EXPTIME-complete.*

Thus, subsumption w.r.t. general TBoxes is polynomial in the fragment $\mathcal{EL}$ of $\mathcal{ALC}$, but it is EXPTIME-complete in the equally harmless looking fragment $\mathcal{FL}_0$—which means, just as hard as subsumption in full $\mathcal{ALC}$. In parallel to our work, Theorem 3.2.13 was independently proved by Hofmann by reduction of the existence of winning strategies in pushdown games [Hof05].

3.3 Hybrid $\mathcal{EL}$-TBoxes

In Section 2.4, we have formally introduced hybrid $\mathcal{EL}$-TBoxes which comprise a cyclic $\mathcal{EL}$-TBox $\mathcal{T}$ on top of a general $\mathcal{EL}$-TBox $\mathcal{F}$ ('foundation') in which atomic concepts from $\mathcal{T}$ can be constrained by means of arbitrary GCIs. While the foundation is interpreted by descriptive semantics, gfp-semantics is used for $\mathcal{T}$. In the present section, we show that subsumption w.r.t. hybrid $\mathcal{EL}$-TBoxes can be decided in polynomial time.

Before turning to hybrid TBoxes, some technical details related to cyclic $\mathcal{EL}$-TBoxes with gfp-semantics are recalled in Section 3.3.1. Additionally, we introduce a normal form for hybrid $\mathcal{EL}$-TBoxes before showing in Section 3.3.2 how subsumption w.r.t. such terminologies can be decided in polynomial time.

3.3.1 Formal preliminaries

Although gfp-semantics has already been defined in Section 2.1.2, it will be necessary for our purposes to know, firstly, how gfp-models can actually be computed; and secondly, how subsumption w.r.t. cyclic $\mathcal{EL}$-TBoxes with gfp-semantics can be decided. Both has been examined in Detail in [Baa03b]. Finally, we introduce a normal form for hybrid TBoxes on which our subsumption algorithm for hybrid $\mathcal{EL}$-TBoxes will be based.

Computing gfp-models

In order to show how the gfp-model $\mathrm{gfp}(\mathcal{T},\mathcal{J})$ can be obtained, we need to introduce the iteration of the fixed point operator $O_{\mathcal{T},\mathcal{J}}$ introduced in Definition 2.1.10 over *ordinals*.

Definition 3.3.1 Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox and $\mathcal{J}$ a primitive interpretation of $\mathsf{N}_{\mathrm{prim}}$ and $\mathsf{N}_{\mathrm{role}}$. Define $\mathcal{I}_{\mathcal{J}}^{\mathrm{top}} \in \mathrm{Int}(\mathcal{J})$ by $\mathcal{I}_{\mathcal{J}}^{\mathrm{top}}(A) := \Delta^{\mathcal{J}}$ for every $A \in \mathsf{N}_{\mathrm{def}}$. For every ordinal α, define

- $\mathcal{I}_{\mathcal{T},\mathcal{J}}^{\downarrow\alpha} := \mathcal{I}_{\mathcal{J}}^{\mathrm{top}}$ if $\alpha = 0$;
- $\mathcal{I}_{\mathcal{T},\mathcal{J}}^{\downarrow\alpha+1} := O_{\mathcal{T},\mathcal{J}}(\mathcal{I}_{\mathcal{T},\mathcal{J}}^{\downarrow\alpha})$;
- $\mathcal{I}_{\mathcal{T},\mathcal{J}}^{\downarrow\alpha} := \mathrm{glb}(\{\mathcal{I}_{\mathcal{T},\mathcal{J}}^{\downarrow\beta} \mid \beta < \alpha\})$ if α is a limit ordinal. □

The following corollary now shows that computing $\mathrm{gfp}(\mathcal{T},\mathcal{J})$ is equivalent to computing $\mathcal{I}_{\mathcal{T},\mathcal{J}}^{\downarrow\alpha}$, given an appropriate ordinal α.

Corollary 3.3.2 *Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox over $\mathsf{N}_{\mathrm{prim}}$, $\mathsf{N}_{\mathrm{role}}$, and $\mathsf{N}_{\mathrm{def}}$. Let $\mathcal{J}$ be a primitive interpretation of $\mathsf{N}_{\mathrm{prim}}$ and $\mathsf{N}_{\mathrm{role}}$. Then there exists an ordinal α such that $\mathrm{gfp}(\mathcal{T},\mathcal{J}) = \mathcal{I}_{\mathcal{T},\mathcal{J}}^{\downarrow\alpha}$.*

Note that if α is a limit ordinal then $\mathcal{I}_{\mathcal{T},\mathcal{J}}^{\downarrow\alpha}$ equals $\bigcap_{\beta<\alpha} \mathcal{I}_{\mathcal{T},\mathcal{J}}^{\downarrow\beta}$.

Deciding subsumption w.r.t. cyclic $\mathcal{EL}$-TBoxes with gfp-semantics

In [Baa03b], a decision procedure for the subsumption problem w.r.t. cyclic $\mathcal{EL}$-TBoxes with descriptive semantics has been presented. We repeat the notions central to this procedure in so far as they are required for our subsumption algorithm w.r.t. hybrid $\mathcal{EL}$-TBoxes.

Definition 3.3.3 An $\mathcal{EL}$-TBox $\mathcal{T}$ is *normalized* iff $A \equiv D \in \mathcal{T}$ implies that D is of the form

$$P_1 \sqcap \cdots \sqcap P_m \sqcap \exists r_1.B_1 \sqcap \ldots \exists r_\ell.B_\ell,$$

where for $m, \ell \geq 0$, $P_1, \ldots, P_m \in \mathsf{N}_{\text{prim}}$ and $B_1, \ldots, B_\ell \in \mathsf{N}_{\text{def}}$. If $m = \ell = 0$ then $D = \top$. □

The subsumption algorithm in [Baa03b] represents normalized $\mathcal{EL}$-TBoxes by means of *description graphs*. These are introduced next.

Definition 3.3.4 (Description graph)
An *$\mathcal{EL}$-description graph* is a graph $\mathcal{G} = (V, E, L)$ where

- V is a set of nodes;
- $E \subseteq V \times \mathsf{N}_{\text{role}} \times V$ is a set of edges labeled by role names;
- $L\colon V \to \wp(\mathsf{N}_{\text{prim}})$ is a function that labels nodes with sets of primitive concepts. □

Description graphs can be used to represent both TBoxes and primitive interpretations. The description graph of a TBox is defined as follows.

Definition 3.3.5 (Description graph of normalized $\mathcal{EL}$-TBoxes)
Let $\mathcal{T}$ be a normalized $\mathcal{EL}$-TBox. Then the $\mathcal{EL}$-description graph $\mathcal{G}_\mathcal{T} = (\mathsf{N}_{\text{def}}^\mathcal{T}, E_\mathcal{T}, L_\mathcal{T})$ of $\mathcal{T}$ is defined as follows:

- the nodes of $\mathcal{G}_\mathcal{T}$ are the defined concepts of $\mathcal{T}$;
- if A is defined in $\mathcal{T}$ and

  $$A \equiv P_1 \sqcap \cdots \sqcap P_m \sqcap \exists r_1.B_1 \sqcap \cdots \sqcap \exists r_\ell.B_\ell$$

  is its definition then $L_\mathcal{T}(A) := \{P_1, \ldots, P_m\}$, and A is the source of the edges $(A, r_1, B_1), \ldots, (A, r_\ell, B_\ell) \in E_\mathcal{T}$. □

Recall Definition 2.1.5, in which an auxiliary definition has been introduced to refer to the definition of a defined concept more conveniently. For every $A \in \mathsf{N}_{\text{def}}^\mathcal{T}$, $L_\mathcal{T}(A)$ can be written as $\mathsf{def}_\mathcal{T}(A) \cap \mathsf{N}_{\text{prim}}^\mathcal{T}$. For primitive interpretations, we define the respective description graphs in the following way.

Definition 3.3.6 (Description graph of primitive interpretations)
Let $\mathcal{J} = (\Delta^\mathcal{J}, \cdot^\mathcal{J})$ be a primitive interpretation. Then the $\mathcal{EL}$-description graph $\mathcal{G}_\mathcal{J} = (\Delta^\mathcal{J}, E_\mathcal{J}, L_\mathcal{J})$ of $\mathcal{J}$ is defined as follows:

- the nodes of $\mathcal{G}_\mathcal{J}$ are the elements of $\Delta^\mathcal{J}$;
- $E_\mathcal{J} := \{(x, r, y) \mid (x, y) \in r^\mathcal{J}\}$;
- $L_\mathcal{J}(x) = \{P \in \mathsf{N}_{\text{prim}} \mid x \in P^\mathcal{J}\}$ for all $x \in \Delta^\mathcal{J}$. □

In preparation for the characterization of subsumption we need to introduce simulation relations on description graphs.

Definition 3.3.7 (Simulation relation)
Let $\mathcal{G}_i = (V_i, E_i, L_i)$, $i = 1, 2$, be two $\mathcal{EL}$-description graphs. The binary relation $Z \subseteq V_1 \times V_2$ is a *simulation relation* from $\mathcal{G}_1$ to $\mathcal{G}_2$ ($Z \colon \mathcal{G}_1 \rightleftharpoons \mathcal{G}_2$) iff

(S1) $(v_1, v_2) \in Z$ implies $L_1(v_1) \subseteq L_2(v_2)$; and

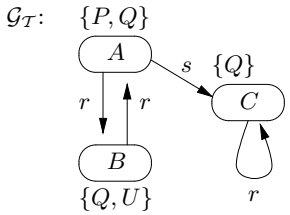
(S2) if $(v_1, v_2) \in Z$ and $(v_1, r, v_1') \in E_1$ then there exists a node $v_2' \in V_2$ such that $(v_1', v_2') \in Z$ and $(v_2, r, v_2') \in E_2$.

In order to get an impression of how description graphs look like, consider the following simple example.

Example 3.3.8 Let $\mathsf{N}_{\text{prim}} := \{P, Q, U\}$ and $\mathsf{N}_{\text{role}} := \{r, s\}$. In our example TBox $\mathcal{T}$, we define the concepts A, B, and C as follows.

$$A \equiv P \sqcap Q \sqcap \exists r.B \sqcap \exists s.C$$
$$B \equiv Q \sqcap U \sqcap \exists r.A$$
$$C \equiv Q \sqcap \exists r.C$$

As $\mathcal{T}$ is already normalized in the sense of Definition 3.3.3, we immediately obtain the following description graph:



where all label sets $L_\mathcal{T}(\cdot)$ are shown above or underneath the corresponding vertex. It is easy to check that a valid simulation relation $Z \colon \mathcal{G}_\mathcal{T} \rightleftharpoons \mathcal{G}_\mathcal{T}$ is, e.g.,

$$Z := \{(C, A), (C, B)\}. \qquad \square$$

It has been shown in [Baa03b] that simulation relations can be concatenated in the sense of the following lemma.

Lemma 3.3.9 *Let $\mathcal{G}_i := (V_i, E_i, L_i)$, $i = 1, 2, 3$, be $\mathcal{EL}$-description graphs, and let $Z_1 \colon \mathcal{G}_1 \rightleftharpoons \mathcal{G}_2$ and $Z_2 \colon \mathcal{G}_2 \rightleftharpoons \mathcal{G}_3$. Then $Z_1 \circ Z_2 \colon \mathcal{G}_1 \rightleftharpoons \mathcal{G}_3$, where*

$$Z_1 \circ Z_2 := \{(v, v'') \mid \exists v' \in V_2 \colon (v, v') \in Z_1 \wedge (v', v'') \in Z_2\}.$$

One of the main results in [Baa03b] is a characterization of gfp-subsumption w.r.t. cyclic $\mathcal{EL}$-TBoxes by simulation relations over description graphs. The following results provide the relevant characterizations.

Proposition 3.3.10 *Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox over N_{prim}, N_{role}, and N_{def} and $A \in \mathsf{N}_{\text{def}}^\mathcal{T}$. Let $\mathcal{J}$ be a primitive interpretation of N_{prim} and N_{role} and let $x \in \Delta^\mathcal{J}$. Then $x \in A^{\text{gfp}(\mathcal{T},\mathcal{J})}$ iff there is a simulation relation $Z \colon \mathcal{G}_\mathcal{T} \rightleftharpoons \mathcal{G}_\mathcal{J}$ such that $(A, x) \in Z$.*

Theorem 3.3.11 *Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox and A, B be defined concepts in $\mathcal{T}$. Then $A \sqsubseteq_{\text{gfp},\mathcal{T}} B$ iff there is a simulation relation $Z \colon \mathcal{G}_\mathcal{T} \rightleftharpoons \mathcal{G}_\mathcal{T}$ such that $(B, A) \in Z$.*

Since the description graph of a TBox is of polynomial size in the size of the TBox and since the existence of simulation relations with the required properties can be tested in polynomial time, the following complexity result is obtained by [Baa03b].

Corollary 3.3.12 *Subsumption w.r.t. cyclic $\mathcal{EL}$-TBoxes with gfp-semantics is decidable in polynomial time.*

Normal form for hybrid TBoxes

As a last preparatory step to our subsumption algorithm for hybrid $\mathcal{EL}$-TBoxes, we introduce a normal form for hybrid $\mathcal{EL}$-TBoxes in order to simplify the presentation of our approach. Since hybrid $\mathcal{EL}$-TBoxes comprise a cyclic $\mathcal{EL}$-TBox 'on top of' a general $\mathcal{EL}$-TBox (see Definition 2.4), it seems natural to normalize the GCIs as defined in Definition 3.2.1 and the cyclic TBox as in Definition 3.3.3.

Definition 3.3.13 (Normalized hybrid TBox)
Let $(\mathcal{F}, \mathcal{T})$ be a hybrid TBox. Then, $(\mathcal{F}, \mathcal{T})$ is *normalized* iff

1. Every GCI in $\mathcal{F}$ is of one of the following forms:

$$\begin{aligned} A &\sqsubseteq B \\ A_1 \sqcap A_2 &\sqsubseteq B \\ A &\sqsubseteq \exists r.B \\ \exists r.A &\sqsubseteq B, \end{aligned}$$

   where $r \in \mathsf{N}_{\text{role}}$ and $A, A_1, A_2, B \in \mathsf{N}_{\text{prim}} \cup \{\top\}$;

2. $\mathcal{T}$ is normalized in the sense of Definition 3.3.3; and

3. for every primitive concept P and for every existential restriction $\exists r.P$ occurring in $\mathcal{F}$, $\mathcal{T}$ contains a definition of the form $A_P \equiv P$ and $A_{\exists r.P} \equiv \exists r.P$, respectively. □

The purpose of the third normalization condition will be clarified in the following section. Note that the first two conditions can be satisfied easily for any hybrid TBox $(\mathcal{F}, \mathcal{T})$, see Lemma 3.2.2 for $\mathcal{F}$ and [Baa03b] for $\mathcal{T}$. For the third condition, a conservative extension $\mathcal{T}'$ of $\mathcal{T}$ of size at most the size of $(\mathcal{F}, \mathcal{T})$ can be found such that $(\mathcal{T}', \mathcal{F})$ is normalized. All subsumption relations between concept names defined in $\mathcal{T}$ remain unchanged.

Normalization serves as an internal preprocessing step to classification and does not replace the original hybrid TBox from the perspective of the user of a DL system.

3.3.2 Deciding Subsumption w.r.t. hybrid $\mathcal{EL}$-TBoxes

In this section we show that subsumption w.r.t hybrid $\mathcal{EL}$-TBoxes $(\mathcal{F}, \mathcal{T})$ can be reduced to subsumption w.r.t. cyclic $\mathcal{EL}$-TBoxes interpreted by gfp-semantics. The underlying idea is to use the *descriptive* subsumption relations induced by the GCIs in $\mathcal{F}$ to extend the definitions in $\mathcal{T}$ accordingly. To this end, we view the union of $\mathcal{F}$ and $\mathcal{T}$ as a general TBox and ask for all descriptive implications in $\mathcal{T}$ directly involving names from $\mathcal{F}$. These implications are then added to the definitions in $\mathcal{T}$. This notion is formalized as follows.

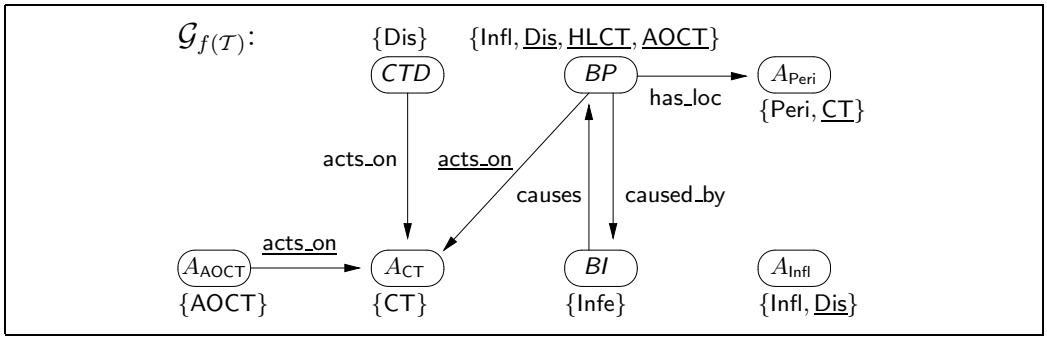


Figure 3.3.1: Example $\mathcal{EL}$-description graph

Definition 3.3.14 ($\mathcal{F}$-completion)
Let $(\mathcal{F}, \mathcal{T})$ be a normalized hybrid $\mathcal{EL}$-TBox. For every $A \in \mathsf{N}_{\text{def}}^{\mathcal{T}}$, let

$$f(A) := \bigsqcap_{P \in \{P' \in \mathsf{N}_{\text{prim}}^{\mathcal{F}} \mid A \sqsubseteq_{\mathcal{F} \cup \mathcal{T}} P'\}} P \sqcap \bigsqcap_{r \in \mathsf{N}_{\text{role}}^{\mathcal{T}}} \bigsqcap_{Q \in \{Q' \in \mathsf{N}_{\text{prim}}^{\mathcal{F}} \mid A \sqsubseteq_{\mathcal{F} \cup \mathcal{T}} \exists r.Q'\}} \exists r.A_Q \ .$$

The *$\mathcal{F}$-completion* $f(\mathcal{T})$ extends the definitions in $\mathcal{T}$ as follows.

$$f(\mathcal{T}) := \{A \equiv C \sqcap f(A) \mid A \equiv C \in \mathcal{T}\}$$ □

Note that $f(\mathcal{T})$ is still a normalized $\mathcal{EL}$-TBox. To preserve normalization, $f(A)$ adds $\exists r.A_Q$ instead of $\exists r.Q$ whenever A implies $\exists r.Q$.

Example 3.3.15 Consider the hybrid TBox $(\mathcal{F}, \mathcal{T})$ from Example 2.4.4 in Section 2.4 after normalization. Our goal is to compute the $\mathcal{F}$-completion of $\mathcal{T}$. To this end, for every defined concept in $\mathcal{T}$, we need to find all descriptive consequences of the form P and $\exists r.P$ implied by $\mathcal{F} \cup \mathcal{T}$, where $P \in \mathsf{N}_{\text{prim}}^{\mathcal{F}}$. Obviously, $A_{\mathsf{Inflammation}}$ implies Disease and $A_{\mathsf{Pericardium}}$ implies ConnTissue. Moreover, $A_{\mathsf{ActsOnConnTissue}}$ yields $\exists$acts_on.ConnTissue. Finally, it is easy to check that *BactPericarditis* implies both Disease and HasLocConnTissue, and therefore also ActsOnConnTissue, yielding $\exists$acts_on.ConnTissue.

Using these descriptive consequences, the completion $f(A)$ can be computed for every defined name A. Figure 3.3.1 shows the "interesting" part of the resulting description graph $\mathcal{G}_{f(\mathcal{T})}$ of the $\mathcal{F}$-completion $f(\mathcal{T})$, omitting some isolated vertices. Long concept names are abbreviated, i.e., the vertex A_{AOCT} stands for the concept $A_{\mathsf{ActsOnConnTissue}}$, A_{HLCT} for $A_{\mathsf{HasLocConnTissue}}$ and so on. For every vertex A with a non-empty label set $L_{f(\mathcal{T})}(A)$, the label set is denoted above or underneath the relevant vertex. Underlined entries are descriptive consequences absent in the original TBox $\mathcal{T}$. As $L_{f(\mathcal{T})}(\mathit{CTD}) \subseteq L_{f(\mathcal{T})}(\mathit{BP})$ and as *BP* has the same successor as *CTD* w.r.t. the edge acts_on, it is easy to check that there exists a simulation relation Z on $\mathcal{G}_{f(\mathcal{T})}$ with $(\mathit{CTD}, \mathit{BP}) \in Z$. Therefore, *BactPericarditis* is subsumed by *ConnTissueDisease* w.r.t. $f(\mathcal{T})$ interpreted with gfp-semantics. □

Our goal now is to show for a given hybrid $\mathcal{EL}$-TBox $(\mathcal{F}, \mathcal{T})$ and arbitrary names A, B defined in $\mathcal{T}$ that B subsumes A w.r.t. $(\mathcal{F}, \mathcal{T})$ if and only if B subsumes A w.r.t. the $\mathcal{F}$-completion of $\mathcal{T}$ interpreted by gfp-semantics. To this end, we first show that $(\mathcal{F}, \mathcal{T})$ and the $\mathcal{F}$-completed hybrid TBox $(\mathcal{F}, f(\mathcal{T}))$ induce the same subsumption relations.

Lemma 3.3.16 *Let $(\mathcal{F}, \mathcal{T})$ be a normalized hybrid $\mathcal{EL}$-TBox. Let $A, B \in \mathsf{N}_{\text{def}}^{\mathcal{T}}$. Then, $A \sqsubseteq_{\text{gfp},\mathcal{F},\mathcal{T}} B$ iff $A \sqsubseteq_{\text{gfp},\mathcal{F},f(\mathcal{T})} B$.*

PROOF. We show that $\mathrm{gfp}(\mathcal{T},\mathcal{J}) = \mathrm{gfp}(f(\mathcal{T}),\mathcal{J})$, implying for every primitive interpretation $\mathcal{J}$ with $\mathcal{J} \models \mathcal{F}$ that $A^{\mathrm{gfp}(\mathcal{T},\mathcal{J})} \subseteq B^{\mathrm{gfp}(\mathcal{T},\mathcal{J})}$ iff $A^{\mathrm{gfp}(f(\mathcal{T}),\mathcal{J})} \subseteq B^{\mathrm{gfp}(f(\mathcal{T}),\mathcal{J})}$, implying the proposition.

In order to show $\mathrm{gfp}(\mathcal{T},\mathcal{J}) = \mathrm{gfp}(f(\mathcal{T}),\mathcal{J})$, it suffices to show that, firstly, every model $\mathcal{I} \in \mathrm{Int}(\mathcal{J})$ of $\mathcal{T}$ is also a model of $f(\mathcal{T})$, implying $\mathrm{gfp}(\mathcal{T}_f,\mathcal{J}) \succeq_{\mathcal{J}} \mathrm{gfp}(\mathcal{T},\mathcal{J})$; and secondly, $\mathrm{gfp}(\mathcal{T}_f,\mathcal{J}) \preceq_{\mathcal{J}} \mathrm{gfp}(\mathcal{T},\mathcal{J})$.

Consider some model $\mathcal{I} \in \mathrm{Int}(\mathcal{J})$ with $\mathcal{I} \models \mathcal{T}$ and an arbitrary $A \in \mathsf{N}^{\mathcal{T}}_{\mathrm{def}}$. As $\mathcal{I} \models \mathcal{T}$, $A^{\mathcal{I}} = \mathsf{deft}_{\mathcal{T}}(A)^{\mathcal{I}}$. Since also $\mathcal{J} \models \mathcal{F}$, $\mathcal{I}$ respects all descriptive implications of $\mathcal{F}$. Hence, $A^{\mathcal{I}} \subseteq f(A)^{\mathcal{I}}$, implying $A^{\mathcal{I}} = \mathsf{deft}_{\mathcal{T}}(A)^{\mathcal{I}} \cap f(A)^{\mathcal{I}} = (\mathsf{deft}_{\mathcal{T}}(A) \sqcap f(A))^{\mathcal{I}}$. By definition of $f(\mathcal{T})$, this yields $\mathcal{I} \models f(\mathcal{T})$.

We show $\mathrm{gfp}(f(\mathcal{T}),\mathcal{J}) \preceq_{\mathcal{J}} \mathrm{gfp}(\mathcal{T},\mathcal{J})$ by transfinite induction on the fixedpoint iteration. By Corollary 3.3.2, there exists an ordinal α such that $\mathrm{gfp}(f(\mathcal{T}),\mathcal{J}) = \mathcal{I}^{\downarrow\alpha}_{f(\mathcal{T}),\mathcal{J}}$ and $\mathrm{gfp}(\mathcal{T},\mathcal{J}) = \mathcal{I}^{\downarrow\alpha}_{\mathcal{T},\mathcal{J}}$. We distinguish the case of α being a successor or a limit ordinal.

(α *successor ordinal*). Induction base: if $\alpha = 0$ then $\mathcal{I}^{\downarrow\alpha}_{f(\mathcal{T}),\mathcal{J}} = \mathcal{I}^{\mathrm{top}}_{\mathcal{J}} = \mathcal{I}^{\downarrow\alpha}_{\mathcal{T},\mathcal{J}}$, implying $\mathcal{I}^{\downarrow 0}_{f(\mathcal{T}),\mathcal{J}} \preceq_{\mathcal{J}} \mathcal{I}^{\downarrow 0}_{\mathcal{T},\mathcal{J}}$. Induction step: for every $\beta < \alpha$, assume (IH) that $\mathcal{I}^{\downarrow\beta}_{f(\mathcal{T}),\mathcal{J}} \preceq_{\mathcal{J}} \mathcal{I}^{\downarrow\beta}_{\mathcal{T},\mathcal{J}}$. Consider an arbitrary $A \in \mathsf{N}^{\mathcal{T}}_{\mathrm{def}}$ defined in $\mathcal{T}$ by

$$A \equiv P_1 \sqcap \cdots \sqcap P_m \sqcap \exists r_1.B_1 \sqcap \ldots \exists r_\ell.B_\ell.$$

We have to show $A^{\mathcal{I}^{\downarrow\beta+1}_{f(\mathcal{T}),\mathcal{J}}} \subseteq A^{\mathcal{I}^{\downarrow\beta+1}_{\mathcal{T},\mathcal{J}}}$. The concept name A is interpreted by $\mathcal{I}^{\downarrow\beta+1}_{f(\mathcal{T}),\mathcal{J}}$ as

$$A^{\mathcal{I}^{\downarrow\beta+1}_{f(\mathcal{T}),\mathcal{J}}} = \bigcap_{1\le i\le m} P_i^{\mathcal{J}} \cap \bigcap_{1\le j\le \ell} (\exists r_j.B_j)^{\mathcal{I}^{\downarrow\beta}_{f(\mathcal{T}),\mathcal{J}}} \cap f(A)^{\mathcal{I}^{\downarrow\beta}_{f(\mathcal{T}),\mathcal{J}}}.$$

For the original TBox $\mathcal{T}$ we analogously have

$$A^{\mathcal{I}^{\downarrow\beta+1}_{\mathcal{T},\mathcal{J}}} = \bigcap_{1\le i\le m} P_i^{\mathcal{J}} \cap \bigcap_{1\le j\le \ell} (\exists r_j.B_j)^{\mathcal{I}^{\downarrow\beta}_{\mathcal{T},\mathcal{J}}}.$$

Hence, it suffices to show for every $r \in \mathsf{N}^{\mathcal{T}}_{\mathrm{role}}$ and every $B \in \mathsf{N}^{\mathcal{T}}_{\mathrm{def}}$ that the subset relation $(\exists r.B)^{\mathcal{I}^{\downarrow\beta}_{f(\mathcal{T}),\mathcal{J}}} \subseteq (\exists r.B)^{\mathcal{I}^{\downarrow\beta}_{\mathcal{T},\mathcal{J}}}$ holds. By IH, $B^{\mathcal{I}^{\downarrow\beta}_{f(\mathcal{T}),\mathcal{J}}} \subseteq B^{\mathcal{I}^{\downarrow\beta}_{\mathcal{T},\mathcal{J}}}$. As $r^{\mathcal{I}^{\downarrow\beta}_{f(\mathcal{T}),\mathcal{J}}} = r^{\mathcal{J}} = r^{\mathcal{I}^{\beta}_{\mathcal{T},\mathcal{J}}}$, the subset relation immediately carries over to the interpretations of $\exists r.B$.

(α *limit ordinal*). Assume (IH) that $\mathcal{I}^{\downarrow\beta}_{f(\mathcal{T}),\mathcal{J}} \preceq_{\mathcal{J}} \mathcal{I}^{\downarrow\beta}_{\mathcal{T},\mathcal{J}}$ for every $\beta < \alpha$. By definition, in the limit ordinal case it holds for every $B \in \mathsf{N}^{\mathcal{T}}_{\mathrm{def}}$ that $B^{\mathcal{I}^{\downarrow\alpha}_{f(\mathcal{T}),\mathcal{J}}}$ equals $\bigcap_{\beta<\alpha} B^{\mathcal{I}^{\downarrow\beta}_{f(\mathcal{T}),\mathcal{J}}}$ which due to IH is a subset of $\bigcap_{\beta<\alpha} B^{\mathcal{I}^{\downarrow\beta}_{\mathcal{T},\mathcal{J}}}$ which in turn equals $B^{\mathcal{I}^{\downarrow\alpha}_{\mathcal{T},\mathcal{J}}}$. □

Hence, the $\mathcal{F}$-completion $(\mathcal{F}, f(\mathcal{T}))$ preserves the subsumption relations of the original hybrid TBox $(\mathcal{F},\mathcal{T})$. Note that it suffices to show the above for concept names $A, B \in \mathsf{N}^{\mathcal{T}}_{\mathrm{def}}$ only. For arbitrary concept descriptions C, D, the analogous claim holds because $C \sqsubseteq_{\mathrm{gfp},\mathcal{F},\mathcal{T}} D$ iff $A \sqsubseteq_{\mathrm{gfp},\mathcal{F},\mathcal{T}\cup\{A\equiv C, B\equiv D\}} B$, where A, B are fresh concept names. The next lemma shows that, after $\mathcal{F}$-completing $\mathcal{T}$, we may 'forget' $\mathcal{F}$ and still obtain the same subsumptions.

Lemma 3.3.17 *Let $(\mathcal{F},\mathcal{T})$ be a normalized hybrid $\mathcal{EL}$-TBox and $A, B \in \mathsf{N}^{\mathcal{T}}_{\mathrm{def}}$. Then, $A \sqsubseteq_{\mathrm{gfp},\mathcal{F},f(\mathcal{T})} B$ iff $A \sqsubseteq_{\mathrm{gfp},f(\mathcal{T})} B$*

PROOF. ($\Leftarrow$) trivial. ($\Rightarrow$) Assume $A \not\sqsubseteq_{\mathrm{gfp},f(\mathcal{T})} B$. We construct a countermodel showing $A \not\sqsubseteq_{\mathrm{gfp},\mathcal{F},f(\mathcal{T})} B$, i.e., a primitive interpretation $\mathcal{J}$ with $\mathcal{J} \models \mathcal{F}$ and $A^{\mathrm{gfp}(f(\mathcal{T}),\mathcal{J})} \not\subseteq B^{\mathrm{gfp}(f(\mathcal{T}),\mathcal{J})}$.

Define $\mathcal{J} =: (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ as follows.

- $\Delta^{\mathcal{J}} := \{x_A \mid A \in \mathsf{N}_{\text{def}}^{f(\mathcal{T})}\}$;
- $P^{\mathcal{J}} := \{x_A \in \Delta^{\mathcal{J}} \mid P \in \mathsf{def}_{f(\mathcal{T})}(A)\}$ for all $P \in \mathsf{N}_{\text{prim}}^{f(\mathcal{T})}$;
- $r^{\mathcal{J}} := \{(x_A, x_B) \in (\Delta^{\mathcal{J}})^2 \mid \exists r.B \in \mathsf{def}_{f(\mathcal{T})}(A)\}$ for all $r \in \mathsf{N}_{\text{role}}^{f(\mathcal{T})}$.

We first show $x_A \in A^{\text{gfp}(f(\mathcal{T}),\mathcal{J})} \setminus B^{\text{gfp}(f(\mathcal{T}),\mathcal{J})}$. By Proposition 3.3.10 it suffices to find a simulation relation $Z \colon \mathcal{G}_{f(\mathcal{T})} \rightleftharpoons \mathcal{G}_{\mathcal{J}}$ with $(A, x_a) \in Z$. Define $Z := \{(A, x_A) \mid A \in \mathsf{N}_{\text{def}}^{f(\mathcal{T})}\}$. As obviously $(A, x_A) \in Z$, it remains to show that Z respects Definition 3.3.7. (S1) For every $(A, x_A) \in Z$, $L_{\mathcal{G}_{f(\mathcal{T})}}(A)$ equals $\mathsf{def}_{f(\mathcal{T})}(A) \cap \mathsf{N}_{\text{prim}}^{f(\mathcal{T})}$ which equals $\{P \in \mathsf{N}_{\text{prim}}^{f(\mathcal{T})} \mid P \in \mathsf{def}_{f(\mathcal{T})}(A)\} = L_{\mathcal{G}_{\mathcal{J}}}(A)$. (S2) If $(A, x_A) \in Z$ and $(A, r, B) \in E_{\mathcal{G}_{f(\mathcal{T})}}$ then $\exists r.B \in \mathsf{def}_{f(\mathcal{T})}(A)$, implying $(x_A, x_B) \in r^{\mathcal{J}}$, implying $(x_A, r, x_B) \in E_{\mathcal{G}_{\mathcal{J}}}$. Moreover, $(B, x_B) \in Z$. Hence, by (S1) and (S2), $Z \colon \mathcal{G}_{f(\mathcal{T})} \rightleftharpoons \mathcal{G}_{\mathcal{J}}$.

Observe that under (S1) we proved *equality* of $L_{\mathcal{G}_{f(\mathcal{T})}}(A)$ and $L_{\mathcal{G}_{\mathcal{J}}}(A)$. Moreover, (S2) also holds in the direction from $\mathcal{G}_{\mathcal{J}}$ to $\mathcal{G}_{f(\mathcal{T})}$: whenever $(A, x_A) \in Z$ and $(x_A, r, x_B) \in E_{\mathcal{G}_{\mathcal{J}}}$ then $(A, r, B) \in E_{\mathcal{G}_{f(\mathcal{T})}}$. Hence, $Z^{-1} \colon \mathcal{G}_{\mathcal{J}} \rightleftharpoons \mathcal{G}_{f(\mathcal{T})}$.

Assume $x_A \in B^{\text{gfp}(f(\mathcal{T}),\mathcal{J})}$. Then, by Proposition 3.3.10, there is a simulation relation $Y \colon \mathcal{G}_{f(\mathcal{T})} \rightleftharpoons \mathcal{G}_{\mathcal{J}}$ with $(B, x_A) \in Y$. But then, by Lemma 3.3.9, $Y \circ Z^{-1}$ is a simulation relation on $\mathcal{G}_{f(\mathcal{T})}$ with $(B, A) \in Y \circ Z^{-1}$, implying $A \sqsubseteq_{\text{gfp},f(\mathcal{T})} B$, in contradiction to the assumption.

It remains to show that $\mathcal{J} \models \mathcal{F}$. As $\mathcal{F}$ is normalized, we have four types of GCIs in $\mathcal{F}$.

1. $P \subseteq Q \in \mathcal{F}$. If $x_A \in P^{\mathcal{J}}$ then $P \in \mathsf{def}_{f(\mathcal{T})}(A)$, implying $A \sqsubseteq_{\mathcal{F} \cup \mathcal{T}} Q$ which implies $f(A) \sqsubseteq Q$. Hence, $Q \in \mathsf{def}_{f(\mathcal{T})}(A)$, implying $x_A \in Q^{\mathcal{J}}$.
2. $P_1 \sqcap P_2 \subseteq Q \in \mathcal{F}$. If $x_A \in P_1^{\mathcal{J}} \cap P_2^{\mathcal{J}}$ then $P_1, P_2 \in \mathsf{def}_{f(\mathcal{T})}(A)$. This implies $A \sqsubseteq_{\mathcal{F} \cup \mathcal{T}} Q$ which analogously yields $x_A \in Q^{\mathcal{J}}$.
3. $P \sqsubseteq \exists r.Q \in \mathcal{F}$. If $x_A \in P^{\mathcal{J}}$ then $P \in \mathsf{def}_{f(\mathcal{T})}(A)$, implying $A \sqsubseteq_{\mathcal{F} \cup \mathcal{T}} \exists r.Q$. Hence, $\exists r.A_Q \in \mathsf{def}_{f(\mathcal{T})}(A)$, implying $(x_A, r, x_{A_{\exists r.Q}}) \in r^{\mathcal{J}}$. By definition, $Q \in \mathsf{def}_{f(\mathcal{T})}(A_{\exists r.Q})$, implying $x_{A_{\exists r.Q}} \in Q^{\mathcal{J}}$.
4. $\exists r.Q \sqsubseteq P \in \mathcal{F}$. If $x_A \in (\exists r.Q)^{\mathcal{J}}$ then there exists some $x_B \in \Delta^{\mathcal{J}}$ such that $(x_A, r, x_B) \in r^{\mathcal{J}}$ and $x_B \in Q^{\mathcal{J}}$. Hence, $\exists r.B \in \mathsf{def}_{f(\mathcal{T})}(A)$ and $Q \in \mathsf{def}_{f(\mathcal{T})}(B)$. This implies $A \sqsubseteq_{\mathcal{F} \cup \mathcal{T}} P$, implying $P \in \mathsf{def}_{f(\mathcal{T})}(A)$ which yields $x_A \in P^{\mathcal{J}}$.

□

As an immediate consequence of Lemmas 3.3.16 and 3.3.17, we obtain the following theorem summarizing our reduction from hybrid $\mathcal{EL}$-TBoxes to cyclic $\mathcal{EL}$-TBoxes.

Theorem 3.3.18 *Let $(\mathcal{F}, \mathcal{T})$ be a normalized hybrid $\mathcal{EL}$-TBox and $A, B \in \mathsf{N}_{\text{def}}^{\mathcal{T}}$. Then, $A \sqsubseteq_{\text{gfp},\mathcal{F},\mathcal{T}} B$ iff $A \sqsubseteq_{\text{gfp},f(\mathcal{T})} B$.*

It remains to show that subsumption w.r.t. hybrid TBoxes is decidable in polynomial time.

Corollary 3.3.19 *Subsumption w.r.t. hybrid $\mathcal{EL}$-TBoxes can be decided in polynomial time in the size of the hybrid TBox.*

PROOF. By Corollary 3.3.12, gfp-subsumption w.r.t. cyclic $\mathcal{EL}$-TBoxes can be decided in polynomial time. Hence, given $(\mathcal{F}, \mathcal{T})$, it suffices to show for every $A \in \mathsf{N}_{\text{def}}^{\mathcal{T}}$ that $f(A)$ is of polynomial size in the size of $(\mathcal{F}, \mathcal{T})$ and can be computed in polynomial time.

By definition, every concept description $f(A)$ contains only conjuncts of the form P and $\exists r.A_P$ with $P \in \mathsf{N}_{\text{prim}}^{\mathcal{T}}$ and $r \in \mathsf{N}_{\text{role}}^{\mathcal{T}}$ occurring in $\mathcal{F}$. The size of $f(A)$ is therefore linear in the size of $(\mathcal{F}, \mathcal{T})$. It has been shown in [Bra04b], see also Section 3.2.2, that subsumption w.r.t. general $\mathcal{EL}$-TBoxes is tractable, implying tractability of subsumption w.r.t. $\mathcal{F} \cup \mathcal{T}$. □

In Sections 4.4.5 and 4.4.6, the non-standard inferences least-common subsumer and matching, respectively, will be introduced for hybrid $\mathcal{EL}$-TBoxes. In particular, we shall see how the above reduction from hybrid to cyclic TBoxes with gfp-semantics can be used to compute the lcs and to solve matching problems w.r.t. the more general DL formalism.

3.4 Related work

In this section, we discuss some of the related work mentioned in Section 1.7.2 in more detail. Especially, we cast more light on the relationship between the 'theoretical' subsumption algorithm for $\mathcal{EL}^{++}$-TBoxes presented in Section 3.2.2 and its implementation for the CEL reasoner. Moreover, we recur to the DL formalism DL-Lite and review its expressive power more closely.

The CEL reasoner

As mentioned previously, the reasoner CEL [BLS05] in its current release can handle $\mathcal{EL}^{+}$-CBoxes, i.e., the bottom concept, nominals, and p-admissible concrete domains are not yet supported. Hence, all parts of the theoretical algorithm related to these constructs are missing in the implementation.

For the remaining algorithm, the implementation in CEL exhibits several improvements over the theoretical algorithm. Although the normalization procedure for a CBox C described in Lemma 3.2.2 increases the size of $\mathcal{C}$ only linearly, experiments have shown that the number of auxiliary concept introduced during normalization still poses a problem in the implementation. For this reason, CEL uses a normal form that is improved in three respects.

- Synonymous concept names induced by GCIs of the form $A \equiv B$, are not handled separately. Instead, the synonym information is stored separately and every occurrence of the synonym B in the CBox is replaced by the original name A.
- Whenever a new name A has to be introduced for a concept description C during the normalization phase, the system checks whether a definition for C has already been introduced at an earlier stage. In this case, the existing name is used instead of introducing A.
- In the normal form used in Lemma 3.2.2, only binary conjunctions of concept names are allowed on left-hand sides of GCIs. In order to reduce the number of auxiliary concept names, CEL accepts conjunctions of arbitrary arity on the left-hand side as long as only concept names are involved.

For the actual completion algorithm, the implementation also follows a refined strategy aimed at improving the system's overall performance. According to Definition 3.2.3, computing all relevant implication sets for $\mathcal{C}$ only means to exhaustively apply the completion rules from Figure 3.2.1 to appropriately initialized sets. Experience with the implementation, however, has shown that an arbitrary choice of completion rules leads to an unsatisfactory performance [BLS05]. In order to overcome this, the authors have adapted

a strategy known from a linear-time algorithm deciding the satisfiability of propositional Horn formulae [DG84].

Following this strategy, the CEL reasoner maintains a certain queue for every implication set to be computed. Depending on the contents of the queues, the reasoner decides which rule to apply to which implication set in the relevant next computation step. Intuitively, the queue of an implication set $\mathsf{Imp}(A)$ stores all modifications of $\mathsf{Imp}(A)$ that still need to be carried out. One immediate advantage of this strategy is that queues can be restricted to applicable modifications, i.e., the resulting algorithm never has to look for applicable completion rules. Moreover, CEL uses a sophisticated selection procedure in order to execute the modifications stored in the above mentioned queues in such a way that the overall work is minimized. For details, refer to [BLS05].

DL-Lite

In Section 1.7.2 we have already mentioned the DL formalism DL-Lite [CDGL$^+$05a] as another interesting approach to tractable reasoning w.r.t. 'useful' DL knowledge bases. Recall that DL-Lite knowledge bases consist of inclusion axioms of the form $B \sqsubseteq C$ with B and C concept descriptions defined by the grammar

$$B ::= P \mid \exists r \mid \exists r^- \qquad C ::= B \mid \neg B \mid C_1 \sqcap C_2,$$

where P stands for arbitrary atomic concepts. Moreover, DL-Lite-TBoxes may contain so-called functionality assertions of the form $(\mathsf{funct}\ r)$ or $(\mathsf{funct}\ r^-)$ by which roles or their inverses can be declared functional. The following example from [CDGL$^+$05a] illustrates what can be expressed in DL-Lite TBoxes.

Example 3.4.1 Assume an underlying DL-Lite TBox $\mathcal{T}$ in which the *basic* concepts *Professor* and *Student* are defined. The following inclusion axioms show an extension of $\mathcal{T}$ in which relations between the two concepts are expressed using the roles has_tutor and teaches_to.

$$\begin{array}{rr} \mathit{Professor} \sqsubseteq \exists \mathsf{teaches\_to} & \mathit{Student} \sqsubseteq \exists \mathsf{has\_tutor} \\ \exists \mathsf{teaches\_to}^- \sqsubseteq \mathit{Student} & \exists \mathsf{has\_tutor}^- \sqsubseteq \mathit{Professor} \\ \mathit{Professor} \sqsubseteq \neg \mathit{Student} & (\text{funct } \mathsf{has\_tutor}) \end{array}$$

As expressed in the left column, the above extension of $\mathcal{T}$ specifies that every professor teaches to someone, that everyone who is taught (by someone) is a student, and that professors and students are disjoint. Due to the right column, every student has some tutor and everyone who is tutor (of someone) is a professor. Moreover, every student has exactly one tutor.

Note that the first two lines on the left imply that every professor teaches to a student and the first two lines on the right that every student is supervised by a professor. Note also that both *Professor* and *Student* must be *basic* concepts as defined by the non-terminal symbol B in the above grammar. Otherwise, they would not be allowed on left-hand sides of inclusion axioms. Although, e.g., *Student* may occur in other inclusion axioms, the means of *defining* *Student* any further are limited. For instance, a student could be defined as anything that has a student ID because the definition

$$\mathit{Student} \equiv \exists \mathsf{has\_student\_id}$$

can be expressed in DL-Lite by two inclusion axioms. However, it is *not* possible to express a definition involving conjunction or qualified existential restrictions, e.g.,

$$\mathit{Student} \equiv \mathsf{Person} \sqcap \exists \mathsf{has\_tutor}.\mathit{Student}.$$ □

It should be noted that both domain and range restrictions on roles can be expressed in DL-Lite-TBoxes. To restrict a role r to the domain C_d and the range C_r, where C_d, C_r can be complex concepts, it suffices to add the inclusion axioms $\exists r \sqsubseteq C_d$ and $\exists r^- \sqsubseteq C_r$ to the TBox.

The combination of inverse roles and inclusion axioms also allows to express a restricted form of value restrictions in DL-Lite TBoxes: for every *basic* concept B, every complex concept C, and every role r it is easy to check that the following equivalences hold.

$$\begin{aligned} B \sqsubseteq \neg\exists r \quad &\text{iff} \quad B \sqsubseteq \forall r.\bot \\ \exists r^- \sqsubseteq C \quad &\text{iff} \quad \top \sqsubseteq \forall r.C \end{aligned}$$

Nevertheless, value restrictions of the form $\forall r.B \sqsubseteq C$ or $B \sqsubseteq \forall r.C$ cannot be encoded. Moreover, although the TBox $\mathcal{T}$ in Example 3.4.1 implies

$$\mathsf{Professor} \sqsubseteq \exists \mathsf{teaches\_to}.\mathsf{Student},$$

this existential restriction cannot be expressed without at the same time enforcing that everyone who is being taught is a student. Hence, even simple existential restrictions representing partonomic structures or causal relations, such as, e.g.,

$$\begin{aligned} \mathsf{Heart} &\sqsubseteq \exists \mathsf{has\_part}.\mathsf{Atrium} \sqcap \exists \mathsf{has\_part}.\mathsf{Ventricle} \\ \mathsf{SevereHypertension} &\sqsubseteq \exists \mathsf{causes}.\mathsf{CoronaryHypertrophy} \end{aligned}$$

cannot be expressed in DL-Lite TBoxes, because every part of the heart would be both atrium and ventricle and every phenomenon causing something would also cause coronary hypertrophy. It should also be clarified that an inclusion axiom of the form $\exists r \sqsubseteq \exists s$ is not equivalent to the RI $r \sqsubseteq s$ because the former only implies the *existence* of a successor w.r.t. s—but does not force the successors w.r.t. both roles to be the same. Hence, DL-Lite does not allow to express role hierarchies. As a consequence, the above problem with partonomic relations cannot be remedied by a work-around of the form

$$\mathsf{Heart} \sqsubseteq \exists \mathsf{has\_atrium}.\mathsf{Atrium} \sqcap \exists \mathsf{has\_ventricle}.\mathsf{Ventricle} \qquad (*)$$

together with the RIs $\mathsf{has\_atrium} \sqsubseteq \mathsf{has\_part}$ and $\mathsf{has\_ventricle} \sqsubseteq \mathsf{has\_part}$. Encoded in a DL-Lite-TBox, the inclusion axioms

$$\begin{aligned} \mathsf{Heart} &\sqsubseteq \exists \mathsf{has\_atrium} & \mathsf{Heart} &\sqsubseteq \exists \mathsf{has\_ventricle} \\ \exists \mathsf{has\_atrium}^- &\sqsubseteq \mathsf{Atrium} & \exists \mathsf{has\_ventricle}^- &\sqsubseteq \mathsf{Ventricle} \end{aligned}$$

indeed do entail $(*)$ but the roles $\mathsf{has\_atrium}$ and $\mathsf{has\_ventricle}$ cannot be defined in such a way as to imply a super-role $\mathsf{has\_part}$. In case of this particular example, it might also be worth commenting that several other anatomical structures exist under the name atrium, e.g., laryngeal atrium and atrium of the ear. In general, most biomedical terminologies build their partonomic hierarchies by means of a very small number of roles, in case of the GO even only one. Hence, even if sub-roles were available, they might be rejected for reasons of inadequate modeling.

On the other hand, DL-Lite does provide constructs unavailable in $\mathcal{EL}^{++}$-CBoxes that are highly useful for real-world modeling tasks, namely inverse roles, functional roles, and range restrictions on roles. Moreover, DL-Lite comes with extremely powerful ABox reasoning capabilities and, as far as experiments show, can handle ABoxes containing millions of individuals [CDGL$^+$05a]. Moreover, a unique feature of the DL-Lite formalism is that a powerful query language for conjunctive queries over ABoxes is available that goes beyond the usual capabilities of ABox reasoning. See [CDGL$^+$05a, ACDG$^+$05] for details.

It has been pointed out in [CDGL+05b] that the DL $\mathcal{EL}$ does not exhibit the highly desirable property that ABox-reasoning can be performed by means of relational Database Management Systems. As a consequence, this strategy is available neither to an extension of DL-Lite by qualified existential restrictions nor to $\mathcal{EL}^{++}$-CBoxes.

It should be stressed that the DL formalisms DL-Lite and $\mathcal{EL}^{++}$-CBoxes have quite different underlying motivations: DL-Lite has been developed to represent conceptual data models, such as entity relationship diagrams, and object-oriented formalisms, such as UML class diagrams, and in particular to process queries over very large ABoxes. In contrast, $\mathcal{EL}^{++}$-CBoxes are intended as an efficient general purpose knowledge representation language, though tailored with a view to the Life Sciences.

CHAPTER 4

MATCHING

In the present chapter, we introduce the non-standard inference problem *matching* formally. In fact, four different types of matching problems (modulo equivalence) will be discussed, namely *concept* matching, concept matching *under side conditions*, matching *w.r.t. cyclic TBoxes*, and matching *w.r.t. hybrid TBoxes*. Common to all types is that concept descriptions are extended to *concept patterns* by *variables* which may occur in the place of atomic concepts. Solving a concept matching problem means, for a given concept description and a concept pattern, to replace all variables occurring in the pattern by concept descriptions such that equivalence to the given concept description holds. Subsumption conditions are suited to express further restrictions on the set of admissible solutions. Finally, matching w.r.t. cyclic or hybrid TBoxes can be seen as an extension of concept matching by a background terminology.

Our contribution to plain concept matching is an implementation of already existing matching algorithms for the DLs $\mathcal{ALE}$ and $\mathcal{ALN}$. The underlying algorithms are presented in Sections 4.1 and 4.2 as starting points for the other, more involved matching algorithms. In Section 4.3, we show how to solve matching problems under side conditions in $\mathcal{ALN}$ and sublanguages. Matching w.r.t. cyclic and hybrid $\mathcal{EL}$-TBoxes is presented in Section 4.4. The actual implementations for matching in $\mathcal{ALE}$ and $\mathcal{ALN}$ are discussed in Sections 4.5.1 and 4.5.2, respectively.

We begin by introducing the most basic form of matching problems, concept matching problems modulo equivalence.

Preliminaries

Denote by N_{var} a finite set of *variables* pairwise disjoint to N_{con} and N_{role}. *Concept patterns* extend concept descriptions by variables. In continuation of Definition 2.1.1, we introduce concept patterns without reference to a specific DL and restrict them later.

Definition 4.0.2 (Concept patterns)
The set of *concept patterns* over N_{con}, N_{role}, $\mathsf{N}_{\leqslant}$, $\mathsf{N}_{\geqslant}$, and N_{var} is inductively defined as follows.

- Every concept description over N_{con}, N_{role}, $\mathsf{N}_{\leqslant}$, and $\mathsf{N}_{\geqslant}$ is a concept pattern;
- every variable $X \in \mathsf{N}_{\text{var}}$ is a concept pattern; and
- if $r \in \mathsf{N}_{\text{role}}$ and D_1, D_2 are concept patterns then so are $D_1 \sqcap D_2$, $\forall r.D_1$, and $\exists r.D_1$.

□

Note that concept variables must not be negated. Trivially, every concept description is a concept pattern. For a given DL $\mathcal{L}$, the set of $\mathcal{L}$-concept patterns is defined as the set of all concept patterns in which only those constructors provided by $\mathcal{L}$ occur. Unless otherwise specified, in the remainder of this work all concept patterns are defined over the sets N_{con}, N_{role}, $\mathsf{N}_{\leqslant}$, $\mathsf{N}_{\geqslant}$, and N_{var}, with N_{var} an arbitrary but fixed finite set pairwise disjoint to the other sets of atomic names. In order to assign values to variables, we introduce substitutions.

Definition 4.0.3 (Substitution)
A mapping σ from the set of $\mathcal{L}$-concept patterns into the set of $\mathcal{L}$-concept descriptions is called a *substitution* iff

- $\sigma(H) = H$ for every $H \in \mathsf{N}_{\text{con}} \cup \{\neg P \mid P \in \mathsf{N}_{\text{con}}\} \cup \{\bot, \top\} \cup \mathsf{N}_{\leqslant} \cup \mathsf{N}_{\geqslant}$;
- $\sigma(D_1 \sqcap D_2) = \sigma(D_1) \sqcap \sigma(D_2)$ and $\sigma(Qr.D_1) = Qr.\sigma(D_1)$ for all concept patterns D_1, D_2, every $Q \in \{\forall, \exists\}$, and every $r \in \mathsf{N}_{\text{role}}$.

For two substitutions σ, τ, σ *is more specific than* τ *($\sigma \sqsubseteq \tau$)* iff $\sigma(X) \sqsubseteq \tau(X)$ for every $X \in \mathsf{N}_{\text{var}}$. Moreover, σ *is strictly more specific than* τ *($\sigma \sqsubset \tau$)* iff $\sigma \sqsubseteq \tau$ and $\sigma(X) \sqsubset \tau(X)$ for some $X \in \mathsf{N}_{\text{var}}$, and σ *is equivalent to* τ *($\sigma \equiv \tau$)* iff $\sigma \sqsubseteq \tau$ and $\tau \sqsubseteq \sigma$. □

Hence, a substitution σ replaces every variable in D by a concept description without changing D otherwise. We are now prepared to introduce the most basic form of matching problems formally.

Definition 4.0.4 (Matching problem)
Let C be an $\mathcal{L}$-concept description and D an $\mathcal{L}$-concept pattern. Then, $C \equiv^? D$ is an *$\mathcal{L}$-matching problem modulo equivalence*. A substitution σ is a *matcher* of $C \equiv^? D$ iff $C \equiv \sigma(D)$.

In general, there are several matchers to a given solvable matching problem. One way to restrict the attention to 'interesting' sets matchers is to compute s-complete sets of matchers as defined below.

Definition 4.0.5 (s-Completeness)
Let $\mathcal{P}$ be an $\mathcal{L}$-matching problem. A matcher σ of $\mathcal{P}$ is called *minimal w.r.t. $\mathcal{P}$* iff, for every matcher τ of $\mathcal{P}$, $\tau \sqsubseteq \sigma$ implies $\sigma \equiv \tau$. A set S of matchers for $\mathcal{P}$ is called *s-complete w.r.t. $\mathcal{P}$* iff S contains all minimal matchers of $\mathcal{P}$. Finally, σ is the *least matcher of $\mathcal{P}$* iff $\{\sigma\}$ is s-complete w.r.t. $\mathcal{P}$. □

Intuitively, $\sigma \sqsubseteq \tau$ iff σ contains more information than τ. In this sense, s-completeness guarantees that a set of matchers contains those solutions that reflect as many aspects of the relevant concept description as possible. Note that, by definition, the least matcher is uniquely determined up to equivalence.

It has been shown in [BKBM99] that the above notion of matching problems can be extended easily to matching modulo subsumption and finite systems of matching problems. For arbitrary concept descriptions C_1 and C_2, $C_1 \sqsubseteq C_2$ iff $C_1 \equiv C_1 \sqcap C_2$. Therefore, matching modulo *subsumption*, i.e., matching problems of the form $C \sqsubseteq^? D$, can be reduced to matching modulo equivalence in linear time. Moreover, a substitution σ is a matcher of two matching problems $C_1 \equiv^? D_1$ and $C_2 \equiv^? D_2$ iff it is a matcher of $Qr_1.C_1 \sqcap Qr_2.C_2 \equiv^? Qr_1.D_1 \sqcap Qr_2.D_2$, where $Q \in \{\exists, \forall\}$ and r_1, r_2 are fresh roles. Hence, finite *systems* of matching problems can be reduced to single matching problems in linear time. We will use this generalized form in order to present our solution strategy

more succinctly. Given a matching problem that computes minimal matchers, we can reduce every lcs computation of a set of concept descriptions $\{C_1, \ldots, C_n\}$ to matching the pattern $Qr_1.X \sqcap \cdots \sqcap Qr_n.X$ (modulo subsumption) against the concept description $Qr_1.C_1 \sqcap \cdots \sqcap Qr_n.C_n$, where $Q \in \{\exists, \forall\}$ and $r_1, \ldots, r_n$ are fresh role names. In this sense, our results on matching also have implications for the non-standard inference lcs.

4.1 Matching in $\mathcal{ALE}$

In the present section, we introduce the $\mathcal{ALE}$-matching algorithm from [BK00a]. In preparation, we require a characterization of subsumption for $\mathcal{ALE}$-concept descriptions to be introduced in the following subsection.

4.1.1 Formal preliminaries

Subsumption of $\mathcal{ALE}$-concept descriptions has been characterized by means of homomorphisms between so-called description trees [BKM99] defined as follows.

Definition 4.1.1 ($\mathcal{ALE}$-description tree)
An $\mathcal{ALE}$-description tree is a tree of the form $\mathcal{G} = (N, E, n_0, L)$ where

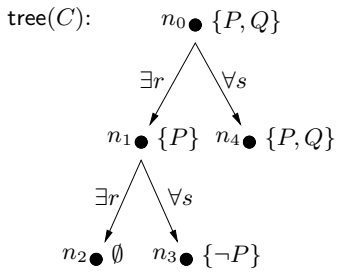
1. N is a finite set of nodes;
2. $E \subseteq N \times \{\exists, \forall\} \times \mathsf{N}_{\text{role}} \times N$ is a finite set of edges each labeled by a quantor and a role name;
3. n_0 is the root node of $\mathcal{G}$;
4. L is a labeling function with $L(n) \subseteq \{\bot\} \cup \mathsf{N}_{\text{con}} \cup \{\neg A \mid A \in \mathsf{N}_{\text{con}}\} \cup \mathsf{N}_{\text{var}}$ for all $n \in N$. □

Description trees correspond to syntax trees of concept descriptions (or concept patterns). In case of concept patterns, variables are treated like atomic concepts. It is easy to see that concept descriptions and patterns can be translated into description trees and back (See [BKM98] for a formal translation). By $\mathsf{tree}(C)$ we denote the description tree of the concept description (or concept pattern) C. For every node n in $\mathsf{tree}(C)$ we denote by $C{\downarrow_n}$ the subdescription obtained by translating the subtree of $\mathsf{tree}(C)$ induced by n back into a concept description. The following example illustrates these notions.

Example 4.1.2 Let $\mathsf{N}_{\text{con}} := \{P, Q\}$ and $\mathsf{N}_{\text{role}} := \{r, s\}$. Then the $\mathcal{ALE}$-concept description

$$C := P \sqcap Q \sqcap \exists r.(P \sqcap \exists r.\top \sqcap \forall s.\neg P) \sqcap \forall s.(P \sqcap Q)$$

has an $\mathcal{ALE}$-description tree of the following form,



where for every $i \in \{0, \ldots, 4\}$, the label set $L(n_i)$ is denoted next to the corresponding node. It is now easy to check that, e.g., $C{\downarrow_{n_2}} = \top$ and $C{\downarrow_{n_1}} = P \sqcap \exists r.\top \sqcap \forall s.\neg P$. □

Subsumption of $\mathcal{ALE}$-concept descriptions has been characterized by means of homomorphisms between the relevant description trees. The following definition introduces the notion of homomorphism used for this purpose.

Definition 4.1.3 (Homomorphism between $\mathcal{ALE}$-description trees)
A mapping $\varphi\colon N_\mathcal{G} \to N_\mathcal{H}$ from an $\mathcal{ALE}$-description tree $\mathcal{G} := (N_\mathcal{G}, E_\mathcal{G}, m_0, L_\mathcal{G})$ to an $\mathcal{ALE}$-description tree $\mathcal{H} := (N_\mathcal{H}, E_\mathcal{H}, n_0, L_\mathcal{H})$ is called *homomorphism* iff the following conditions hold:

1. $\varphi(m_0) = n_0$;
2. for all nodes $n \in N_\mathcal{G}$, $L_\mathcal{G}(n) \setminus \mathsf{N}_{\text{var}} \subseteq L_\mathcal{H}(\varphi(n))$ or $\bot \in L_\mathcal{H}(\varphi(n))$;
3. For all edges $(n\, Q r\, m) \in E_\mathcal{G}$, either $(\varphi(n)\, Q r\, \varphi(m)) \in E_\mathcal{H}$, or $\varphi(n) = \varphi(m)$ and $\bot \in L_\mathcal{H}(\varphi(n))$. □

Note that concept variables are ignored for the definition of homomorphisms. As shown in [BKM99], subsumption of $\mathcal{ALE}$-concept descriptions can now be characterized as follows.

Theorem 4.1.4 *Let C, D be $\mathcal{ALE}$-concept descriptions. Then, $C \sqsubseteq D$ iff there exists a homomorphism φ from $\mathsf{tree}(D)$ onto $\mathsf{tree}(C)$.*

Note, however, that the above definition of homomorphism includes the case of D being a concept pattern while for this case there is no corresponding notion of subsumption. In preparation for the actual $\mathcal{ALE}$-matching algorithm from [BK00a], we still need to introduce $\top$-patterns and their normal form.

Definition 4.1.5 ($\top$-pattern, $\top$-normal form)
Let D be an $\mathcal{ALE}$-concept pattern. Then, D' is called a *$\top$-pattern of D* iff D' is obtained from D by replacing some, i.e., zero or more, variables occurring in D by $\top$. Moreover, the *$\top$-normal form $D'^{\top}$ of D'* is defined by exhaustive application of the simplification rule

$$\forall r.\top \longrightarrow \top,$$

with $r \in \mathsf{N}_{\text{role}}$, to D' and all of its subdescriptions.

The following simple example illustrates the notion of $\top$-patterns and their normal forms.

Example 4.1.6 Consider the concept pattern $D := \exists r.(X \sqcap Y) \sqcap \forall r.X$. Replacing X by $\top$ we obtain a $\top$-pattern D' of D with $D' = \exists r.(\top \sqcap Y) \sqcap \forall r.\top$. Consequently, $D'^{\top}$ is of the form $\exists r.(\top \sqcap Y)$.

4.1.2 Solving $\mathcal{ALE}$-matching problems

Definition 4.1.7 Let $C \equiv^? D$ be an $\mathcal{ALE}$-matching problem. The algorithm $\mathsf{match}_{\mathcal{ALE}}$ is defined as shown in Figure 4.1.1. □

It has been shown in [BK99, BK00a] that the above $\mathcal{ALE}$-matching algorithm in fact computes s-complete sets of matchers, that the number of returned matchers is at most exponential, and that each matcher is of size at most exponential in the size of the input matching problem.

Input: $\mathcal{ALE}$-matching problem $C \equiv^? D$
Output: s-complete set $\mathcal{C}$ of matchers for $C \equiv^? D$

$\mathcal{C} := \emptyset$
For all $\top$-patterns D' of D do
  For all homomorphisms φ from $\mathcal{G} := \mathsf{tree}(D'^{\top})$ into $\mathsf{tree}(C)$

$$\text{Define } \sigma \text{ by } \sigma(X) := \begin{cases} \mathsf{lcs}\{C{\downarrow_{\varphi(m)}} \mid m \in N_{\mathcal{G}}, X \in L_{\mathcal{G}}(m)\} & \text{if } X \in \mathrm{var}(D') \\ \top & \text{otherwise} \end{cases}$$

  If $C \sqsupseteq \sigma(D)$ then $\mathcal{C} := \mathcal{C} \cup \{\sigma\}$

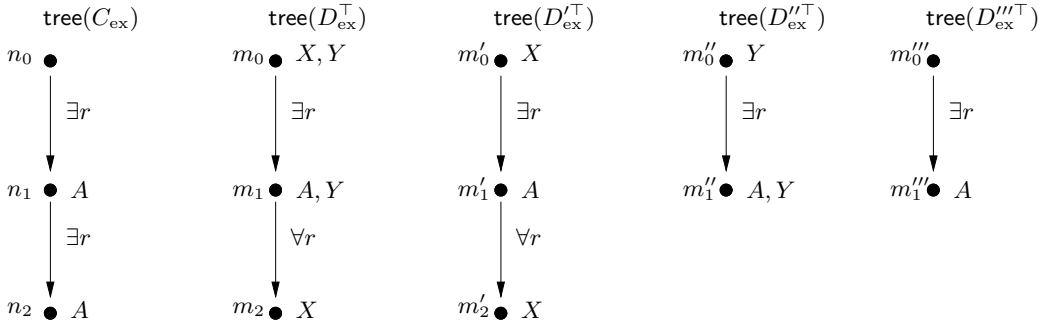
Return $\mathcal{C}$.

Figure 4.1.1: The $\mathcal{ALE}$-Matching Algorithm

In [BK00a] it is also shown that the matching algorithm is a deterministic exponential space algorithm. It is still open whether this upper bound is 'tight', and especially, if sets of s-complete matchers can also be computed in exponential time—currently the best lower bound for this computation problem.

Example 4.1.8 Let $\mathsf{N}_{\mathrm{con}} := \{A\}$ and $\mathsf{N}_{\mathrm{role}} := \{r\}$. Consider the small matching problem $C_{\mathrm{ex}} \equiv^? D_{\mathrm{ex}}$ with $C_{\mathrm{ex}} := \exists r.(A \sqcap \exists r.A)$ and $D_{\mathrm{ex}} := X \sqcap Y \sqcap \exists r.(A \sqcap Y \sqcap \forall r.X)$. The relevant description trees are shown below:

In order to apply the matching algorithm shown in Figure 4.1.1, we have to start by computing all $\top$-patterns D'_{ex} of D_{ex}. Apart from D_{ex} itself, these are $X \sqcap \exists r.(A \sqcap \forall r.X) =: D'_{\mathrm{ex}}$, $Y \sqcap \exists r.(A \sqcap Y \sqcap \forall r.\top) =: D''_{\mathrm{ex}}$, and $\exists r.(A \sqcap \forall r.\top) =: D'''_{\mathrm{ex}}$. The next step is to compute the respective $\top$-normal forms. Clearly, the $\top$-normal form of D_{ex} and D'_{ex} is equivalent to the original concepts. For D''_{ex} and D'''_{ex}, however, the value restriction $\forall r.\top$ is removed. The description trees of the relevant normalized concepts are shown below.



Because of the universal r-edge in $\mathsf{tree}(D_{\mathrm{ex}}^{\top})$ and $\mathsf{tree}(D'^{\top}_{\mathrm{ex}})$, which is missing in $\mathsf{tree}(C_{\mathrm{ex}})$, no homomorphism exists from $\mathsf{tree}(D_{\mathrm{ex}}^{\top})$ or $\mathsf{tree}(D'^{\top}_{\mathrm{ex}})$ onto $\mathsf{tree}(C_{\mathrm{ex}})$. However, by mapping m''_0 onto n_0 and m''_1 onto n_1, we find a homomorphism φ from $\mathsf{tree}(D''^{\top}_{\mathrm{ex}})$ onto $\mathsf{tree}(C_{\mathrm{ex}})$. The next step is to construct a substitution σ according to the definition in Figure 4.1.1. Since X is no element of $\mathrm{var}(D''_{\mathrm{ex}})$, $\sigma(X) = \top$. Moreover, as Y occurs in m''_0 and m''_1, we have to compute the lcs of $C_{\mathrm{ex}}{\downarrow_{\varphi(m''_0)}}$ and $C_{\mathrm{ex}}{\downarrow_{\varphi(m''_1)}}$. Since $\varphi(m''_0) = n_0$ and $\varphi(m''_1) = n_1$, this means to compute the lcs of C_{ex} and $\exists r.A$. Thus, we obtain $\sigma(Y) = \exists r.A$. In the next step of the algorithm, we find that $\sigma(D) = \exists r.A \sqcap \exists r.(A \sqcap \exists r.A)$ which is subsumed by the input concept C_{ex}. Thus, σ is added to the list $\mathcal{C}$ of solutions.

For the $\top$-pattern D'''_{ex}, the only homomorphism φ from $\mathsf{tree}(D'''^{\top}_{\text{ex}})$ onto $\mathsf{tree}(C_{\text{ex}})$ clearly also maps m'''_0 onto n_0 and m'''_1 onto m_1. However, since $D'''^{\top}_{\text{ex}}$ contains no variables, we immediately obtain the substitution $\sigma' = \{X \mapsto \top, Y \mapsto \top\}$. In this case, the final subsumption test fails, i.e., $C_{\text{ex}} \not\sqsubseteq \sigma'(D)$.

As a result, the set $\{\sigma\} = \{\{X \mapsto \top, Y \mapsto \exists r.A\}\}$ is returned as solution for the matching problem $C_{\text{ex}} \equiv^? D_{\text{ex}}$. □

In the context of matching w.r.t. cyclic $\mathcal{EL}$-TBoxes in Section 4.4.1, the basic principle of the above matching algorithm, i.e., computing matchers by finding homomorphisms between trees and computing the lcs, will be extended to computing matchers by finding certain relations between directed graphs and again computing the lcs.

How the above $\mathcal{ALE}$-matching algorithm can be implemented is the subject of Section 4.5.1.

4.2 Matching in $\mathcal{ALN}$

Besides $\mathcal{ALE}$, matching in $\mathcal{ALN}$ is the second instance of concept matching to introduce. Although $\mathcal{ALN}$-matching problems look basically like their counterparts from $\mathcal{ALE}$, the entirely different characterization of subsumption for $\mathcal{ALN}$-concept descriptions suggests the use of different underlying techniques for matching in $\mathcal{ALN}$. The relevant approach has first been presented in [BKBM99].

In addition to $\mathcal{ALN}$, we also examine matching in two sublanguages of $\mathcal{ALN}$, namely $\mathcal{FL}_\bot$ and $\mathcal{FL}_\neg$. The main motive behind this is that, for matching problems, the computational complexity obtained for $\mathcal{ALN}$ does not automatically carry over to its sublanguages, i.e., the unavailability of some constructs in the sublanguage might make it more difficult to find a solution.

In the following subsection, we introduce a characterization of subsumption of $\mathcal{ALN}$-concept descriptions (and sublanguages) based on role languages. Note that a similar characterization has been used in Section 3.1.2 for the DLs $\mathcal{FL}_0$ and $\mathcal{L}_{\forall\exists}$.

4.2.1 Formal preliminaries

To simplify notation, let $\neg\mathsf{N}_{\text{con}} := \{\neg A \mid A \in \mathsf{N}_{\text{con}}\}$. Furthermore, denote the set of primitive concepts by

$$\mathsf{P} := \mathsf{N}_{\text{con}} \cup \neg\mathsf{N}_{\text{con}} \cup \{\bot\} \cup \mathsf{N}_{\leqslant} \cup \mathsf{N}_{\geqslant}.$$

Our aim is to represent $\mathcal{ALN}$-concept descriptions up to equivalence by vectors of formal languages, one language for each $P \in \mathsf{P}$, and to characterize subsumption accordingly. To this end, we introduce the so-called $\mathcal{FL}_0$-normal form [BKBM99] for $\mathcal{ALN}$-concept descriptions. The name '$\mathcal{FL}_0$-normal form' stems from the fact that such a normal form has first been introduced for $\mathcal{FL}_0$-concept descriptions as 'concept centered normal form' in [BN98].[1]

Definition 4.2.1 ($\mathcal{FL}_0$-normal form)
For every $P \in \mathsf{P}$, let R_P denote an arbitrary finite formal language over N_{role}. Then

$$C := \bigsqcap_{P \in \mathsf{P}} \forall R_P.P$$

is an *$\mathcal{ALN}$-concept description in $\mathcal{FL}_0$-normal form* representing the $\mathcal{ALN}$-concept description

$$\bigsqcap_{P \in \mathsf{P}} \bigsqcap_{n \in \mathbb{N}} \bigsqcap_{r_1 \cdots r_n \in R_P} \forall r_1 \ldots \forall r_n.P.$$

[1] A similar normal form is called *unfolding* in [Neb90].

For every $\mathcal{ALN}$-concept description D, define $\forall\emptyset.D := \top$ and $\forall\{\varepsilon\}.D := D$. □

Recall that every R_P occurring in the $\mathcal{FL}_0$-normal form of C is called a *role language*. Concept *patterns* in $\mathcal{FL}_0$-normal form are defined analogously by adding a role language R_X for every $X \in \mathsf{N}_{\text{var}}$. As shown in [BKBM99], every $\mathcal{ALN}$-concept description can be transformed into $\mathcal{FL}_0$-normal form in polynomial time.

As usual, denote by $\overline{R}$ the complement of any role language R, i.e., $\overline{R} := \mathsf{N}^*_{\text{role}} \setminus R$. In order to refer to the role languages occurring in the $\mathcal{FL}_0$-normal form more easily, the following notation is introduced.

Definition 4.2.2 Let D be an $\mathcal{ALN}$-concept pattern. Then the $\mathcal{FL}_0$-normal form of D is of the form $\bigsqcap_{P\in\mathsf{P}\cup\mathsf{N}_{\text{var}}} R_P.P$. For every $P \in \mathsf{P} \cup \mathsf{N}_{\text{var}}$, define $C|_P := R_P$. □

The motive behind introducing the $\mathcal{FL}_0$-normal form is a formal-language characterization of equivalence introduced in [BKBM99]. For the presentation of this characterization, and also for our solution strategy for matching problems and matching problems under side conditions, we need to introduce some auxiliary operations on formal languages, namely the left and right product of words with formal languages, and the left product of languages.

Definition 4.2.3 (Left and right products)
Let $L, M \subseteq \mathsf{N}^*_{\text{role}}$ and $w \in \mathsf{N}^*_{\text{role}}$. Then the *left product of w and L ($w^{-1} \cdot L$)*, the *right product of w and L ($L \cdot w^{-1}$)*, and the *left product of L and M ($L^{-1} \cdot M$)* are defined as follows.

$$\begin{aligned} w^{-1} \cdot L &:= \{v \in \mathsf{N}^*_{\text{role}} \mid wv \in L\} \\ L \cdot w^{-1} &:= \{v \in \mathsf{N}^*_{\text{role}} \mid vw \in L\} \\ L^{-1} \cdot M &:= \bigcap_{w\in L} w^{-1} \cdot M \end{aligned}$$ □

Note that always $\{w\} \cdot (w^{-1} \cdot L) \subseteq L$ and $(L \cdot w^{-1}) \cdot \{w\} \subseteq L$, which initially motivated the notation using the 'inverse' of w. Similarly, $L^{-1} \cdot M$ is the largest language N with $L \cdot N \subseteq M$. For every language L it trivially holds that $\{\varepsilon\}^{-1} \cdot L = L$ and $\emptyset^{-1} \cdot L = \mathsf{N}^*_{\text{role}}$. In formal language expressions, we define the left and right product to precede union, intersection and concatenation. Hence, $L^{-1} \cdot M \cdot N$ stands for $(L^{-1} \cdot M) \cdot N$, $L \cup M \cdot w^{-1}$ for $L \cup (M \cdot w^{-1})$ and so on. The following simple example illustrates the above operators further.

Example 4.2.4 Let $\mathsf{N}_{\text{role}} := \{r, s, t\}$. Let $L := \{r, rs, t\}$ and $M := \{rsst, rst, s, tst\}$. Then, e.g., $rs^{-1} \cdot L = \{\varepsilon\}$, $rs^{-1} \cdot M = \{st, t\}$, $L \cdot t^{-1} = \{\varepsilon\}$, $M \cdot t^{-1} = \{rs, rss, ts\}$, and $L^{-1} \cdot M = \{st\}$. □.

We are now prepared to introduce the formal-language characterization of equivalence of $\mathcal{ALN}$-concept descriptions introduced in [BKBM99].

Lemma 4.2.5 *(Characterization of equivalence)*
Let C_1 and C_2 be $\mathcal{ALN}$-concept descriptions. Then, $C_1 \equiv C_2$ iff for every $P \in \mathsf{N}_{\text{con}} \cup \neg\mathsf{N}_{\text{con}}$,

for every $(\geqslant n\ r) \in \mathsf{N}_{\geqslant}$*, and for every* $(\leqslant n\ r) \in \mathsf{N}_{\leqslant}$ *it holds that*

$$E_{C_1} = E_{C_2} \tag{$\bot$}$$
$$C_1|_P \cup E_{C_1} = C_2|_P \cup E_{C_2} \tag{P}$$
$$\bigcup_{m \geq n} C_1|_{(\geqslant m\ r)} \cup E_{C_1} = \bigcup_{m \geq n} C_2|_{(\geqslant m\ r)} \cup E_{C_2}$$
$$\bigcup_{m \leq n} C_1|_{(\leqslant m\ r)} \cup E_{C_1} \cdot r^{-1} = \bigcup_{m \leq n} C_2|_{(\leqslant m\ r)} \cup E_{C_2} \cdot r^{-1},$$

where $E_{C_i} = \{w \in \mathsf{N}^*_{\mathrm{role}} \mid C_i \sqsubseteq \forall\{w\}.\bot\}$ *for* $i = 1, 2$.

In [Bra00, BBK01], a so-called 'reduced' normal form has been introduced in order to simplify the characterization of subsumption of $\mathcal{ALN}$-concept descriptions further.

Definition 4.2.6 (Reduced normal form)
An $\mathcal{ALN}$-concept description C in $\mathcal{FL}_0$-normal form is *reduced* iff

- $C|_\bot = R \setminus (R \cdot \mathsf{N}^+_{\mathrm{role}})$, where R is a finite language over $\mathsf{N}_{\mathrm{role}}$ with
$$R \cdot \mathsf{N}^*_{\mathrm{role}} = \{w \in \mathsf{N}^*_{\mathrm{role}} \mid C \sqsubseteq \forall\{w\}.\bot\};$$
- $C|_P = C|_P \setminus (C|_\bot \cdot \mathsf{N}^*_{\mathrm{role}})$ for every $P \in \mathsf{N}_{\mathrm{con}} \cup \neg\mathsf{N}_{\mathrm{con}}$;
- $C|_{(\leqslant n\ r)} = \bigcup_{m \leq n} C|_{(\leqslant n\ r)} \setminus (C|_\bot \cdot \mathsf{N}^*_{\mathrm{role}} \cdot r^{-1})$; and
- $C|_{(\geqslant n\ r)} = \bigcup_{m \geq n} C|_{(\geqslant n\ r)} \setminus (C|_\bot \cdot \mathsf{N}^*_{\mathrm{role}})$. □

In order to present the relevant characterization of subsumption, we first have to introduce a relation on formal languages, the so-called multiset order.

Definition 4.2.7 (Multiset order)
Let $\succ \subseteq \wp(\mathsf{N}^*_{\mathrm{role}}) \times \wp(\mathsf{N}^*_{\mathrm{role}})$ such that, for every $L, M \subseteq \mathsf{N}^*_{\mathrm{role}}$, $L \succ M$ iff there exist finite languages $X, Y \subseteq \mathsf{N}^*_{\mathrm{role}}$ with:

- $\emptyset \neq X \subseteq M$;
- $L = (M \setminus X) \cup Y$; and
- $\forall y \in Y \exists x \in X : x <_{pr} y$. □

Note that $U \succ V$ iff U can be transformed into V by performing the following step one or more times: remove a word u from U and replace it by a finite subset of $\{u\} \cdot \mathsf{N}^+_{\mathrm{role}}$. The following example illustrates this.

Example 4.2.8 Let $\mathsf{N}_{\mathrm{role}} := \{r, s, t\}$. Then, e.g., $\{r, rs, t\} \succ \{rrr, rsr, rstr\}$ because r can be replaced by rrr and rsr, rs can be replaced by $rstr$, and t can be removed without replacement. Moreover, $\{r, rs, t\} \succ \emptyset$ because all words can be removed without replacement. On the other hand, $\{r, rs, t\} \not\succ \{r, s\}$ because s does not correspond to an extension of any of r, rs, or t. □

Equivalence and subsumption between reduced $\mathcal{ALN}$-concept descriptions can now be characterized as follows.

Lemma 4.2.9 *(Characterization of subsumption for reduced normal forms)*
Let C, D be reduced $\mathcal{ALN}$-concept descriptions over N_{con}, N_{role}, $\mathsf{N}_{\leqslant}$, and $\mathsf{N}_{\geqslant}$. Then

1. *$C \equiv D$ iff $C|_P = D|_P$ for every $P \in \mathsf{P}$;*
2. *$C \sqsubset D$ iff either*
   - *$C|_\perp = D|_\perp$, $C|_P \supseteq D|_P$ for all $P \in \mathsf{P} \setminus \{\perp\}$, and $C|_P \neq D|_P$ for at least one $P \in \mathsf{P} \setminus \{\perp\}$; or*
   - *$C|_\perp \succ D|_\perp$, $D|_P \supseteq C|_P \cup C|_\perp \cdot \mathsf{N}^*_{\text{role}}$ for every $P \in \mathsf{P} \setminus (\{\perp\} \cup \mathsf{N}_{\leqslant})$, and $D|_P \supseteq C|_P \cup C|_\perp \cdot \mathsf{N}^*_{\text{role}} \cup C|_\perp \cdot r^{-1} \cdot \mathsf{N}^*_{\text{role}}$ for every $P =: (\leqslant n\, r) \in \mathsf{N}_{\leqslant}$.*

Note that Definitions 4.2.1 and 4.2.6 implicitly also define (reduced) normal forms for the sublanguages $\mathcal{FL}_\perp$ and $\mathcal{FL}_\neg$ of $\mathcal{ALN}$. Moreover, as shown in [Bra00, BBK01], the relevant characterizations of subsumption for $\mathcal{FL}_\perp$ and $\mathcal{FL}_\neg$ correspond to the restriction of Lemma 4.2.5 to the first two equations and of Lemma 4.2.9 to the first two conditions, where $\neg\mathsf{N}_{\text{con}} = \emptyset$ for $\mathcal{FL}_\perp$. It is easy to see for every $\mathcal{FL}_\perp$-concept description C that $E_C = C|_\perp \cdot \Sigma^*$ and, analogously,

$$E_C = \big(C|_\perp \cup \bigcup_{P \in \mathsf{N}_{\text{con}}} C|_P \cap C|_{\neg P}\big) \cdot \Sigma^*$$

for every $\mathcal{FL}_\neg$-concept description C. With this preparation, we introduce the actual $\mathcal{ALN}$-matching algorithm in the following section.

4.2.2 Solving $\mathcal{ALN}$-matching problems

By means of the $\mathcal{FL}_0$-normal form, a matching problem can be viewed as a problem over formal languages. One main result from [BKBM99] shows how the least matcher to a solvable $\mathcal{ALN}$-matching problem can be constructed.

Lemma 4.2.10 *Let $C \equiv^? D$ be an $\mathcal{ALN}$-matching problem. Then, either $C \equiv^? D$ is not solvable or it has a least matcher σ that assigns to each variable X occurring in D the concept description $\sigma(X)$ defined by*

$$\sigma(X) := \forall W^X_\perp.\perp \;\sqcap \prod_{P \in \mathsf{P}} \forall (D|_X^{-1} \cdot W^X_P) \setminus (D|_X^{-1} \cdot E_C).P,$$

where $E_C = \{w \in \mathsf{N}^*_{\text{role}} \mid C \sqsubseteq \forall\{w\}.\perp\}$, $W^X_\perp$ is a role language of polynomial size in C with $W^X_\perp \cdot \mathsf{N}^*_{\text{role}} = D|_X^{-1} \cdot E_C$, and all role languages of the form W^X_P with $P \in \mathsf{P} \setminus \{\perp\}$ are defined as follows.

$$W^X_P := \begin{cases} C|_P \cup E_C & \text{if } P \in \mathsf{N}_{\text{con}} \cup \neg\mathsf{N}_{\text{con}} \\ \bigcup\limits_{m \geq n} C|_{(\geqslant m\, r)} \cup E_C & \text{if } P =: (\geqslant n\, r) \in \mathsf{N}_{\geqslant} \\ \bigcup\limits_{m \leq n} C|_{(\leqslant m\, r)} \cup E_C \cdot r^{-1} & \text{if } P =: (\leqslant n\, r) \in \mathsf{N}_{\leqslant} \end{cases}$$

There are at least two strategies to decide whether the substitution σ defined above solves the matching problem $C \equiv^? D$. Either ascertain the solvability of $C \equiv^? D$ *before* computing σ, or compute σ first and decide the equivalence $C \equiv \sigma(D)$ *afterwards*. In [BKBM99], the former strategy is taken: a system of formal language equations, so-called 'solvability equations', is proposed which is solvable iff $C \equiv^? D$ is. Deciding solvability of these

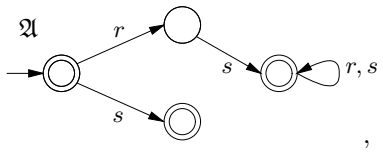
equations, however, necessitates computing exactly those role languages which occur in the $\mathcal{FL}_0$-normal form of $\sigma(X)$ constructed in Lemma 4.2.10.

As the second strategy is computationally equivalent but more easily explained, we deviate from the original in [BKBM99] by computing a candidate solution first and testing for equivalence afterwards. To this end, we utilize the a characterization of equivalence from Lemma 4.2.5.

Informally, the $\mathcal{ALN}$-matching algorithm $\mathsf{match}_{\mathcal{ALN}}$ can now be described as follows. Upon input $C \equiv^? D$, (i) transform C and D into $\mathcal{FL}_0$-normal form, (ii) construct the candidate solution σ defined in Lemma 4.2.10, and (iii) test whether C and $\sigma(D)$ satisfy the formal language equations shown in Lemma 4.2.5. If they do, return the least matcher σ, otherwise return 'fail'. It remains to provide a method by which to solve Steps (ii) and (iii) in polynomial time.

To this end, so-called 'tree-like automata' [BKBM99], can be utilized. Intuitively, these are deterministic finite automata whose structure differs from a tree only in that they either have ordinary leaves or leaves with an r-transition to themselves for every $r \in \mathsf{N}_{\text{role}}$. Consider the following example.

Example 4.2.11 Let $\mathsf{N}_{\text{role}} = \{r, s\}$. Then the role language $\{\varepsilon, s\} \cup \{rs\} \cdot \mathsf{N}^*_{\text{role}}$ can be represented by a tree-like automaton $\mathfrak{A}$ of the form



,

where $\longrightarrow$ marks the initial state and double circles denote final states. □

As shown in [Bra00, BKBM99], tree-like automata have the following properties.

- A tree-like automaton $\mathfrak{A}$ that accepts E_C can be constructed in polynomial time in the size of C. From $\mathfrak{A}$, a language U of polynomial size in C with $E_C = U \cdot \mathsf{N}^*_{\text{role}}$ can be constructed in linear time.
- The operations union, intersection, and complement on treelike automata can be defined in such a way that the size of the resulting automaton does not exceed the maximum of the sizes of the input automata. Moreover, all operations can be performed in linear time.
- If U, V, W are finite languages, then a tree-like automaton accepting $U^{-1} \cdot (V \cup W \cdot \mathsf{N}^*_{\text{role}})$ can be constructed in polynomial time in the size of the input.[2]

As a consequence, tree-like automata can be used to construct the candidate solution σ defined in Lemma 4.2.10 in polynomial time. It remains to show how tree-like automata can be used to test whether σ actually is a solution.

Consider a matching problem $C \equiv^? D$ in $\mathcal{FL}_0$-normal form with a candidate solution σ as defined in Lemma 4.2.10. Instantiating the entire system of equations from Lemma 4.2.5 by C and $\sigma(D)$ is beyond the scope of this section. Nevertheless, as a typical example, consider Equation (P) defined for every $P \in \mathsf{N}_{\text{con}} \cup \neg\mathsf{N}_{\text{con}}$. Inserting the role languages from C and $\sigma(D)$, we obtain the following equation.

$$C|_P \cup E_C = D|_P \cup E_{\sigma(D)} \cup \bigcup_{X \in \mathsf{N}_{\text{var}}} D|_X \cdot \left(D|_X^{-1} \cdot (C|_P \cup E_C)\right) \qquad (*)$$

[2]Note, however, that the relevant paper uses a different notation in that the left product operator is introduced only for words, i.e., $w^{-1} \cdot L$, but not for formal languages, i.e., $L^{-1} \cdot M$.

Assume that Equation ($\bot$) has already been tested, i.e., $E_C = E_{\sigma(D)}$. By definition of the left product, the union over all $X \in \mathsf{N}_{\text{var}}$ on the right-hand side of ($*$) is always a subset of the left-hand side of the equation. Hence, Equation ($*$) holds iff (i) $D|_P \subseteq C|_A \cup E_C$ and (ii) for all $u \in C|_P$ either (iia) $u \in D|_X \cup E_{\sigma(D)}$ or (iib) $u \in D|_X \cdot (D|_X^{-1} \cdot C|_P)$ or (iic) $u \in D|_P \cdot (D|_X^{-1} \cdot E_C)$. Condition (i) can be decided by testing the tree-like automaton of $D|_P \cap \overline{(C|_P \cup E_C)}$ for emptiness. For Condition (iia), merely the word problem w.r.t. the tree-like automaton for $D|_P \cup E_{\sigma(D)}$ must be decided for every $u \in C|_P$. Since there is no concatenation defined for tree-like automata, the remaining Conditions (iib) and (iic) cannot be solved by means of one single treelike automaton. Nevertheless, one can show that $u \in D|_X \cdot (D|_X^{-1} \cdot C|_P)$ iff $\{u\}^{-1} \cdot D|_X \cap D|_X^{-1} \cdot C|_P$ is not empty, which again can be decided by a tree-like automaton in polynomial time. Case (iic) is analogous. The other equations from Lemma 4.2.10 can be decided similarly, see [BKBM99] for details.

This completes our overview of matching in $\mathcal{ALN}$. Note that the above algorithm $\mathsf{match}_{\mathcal{ALN}}$ implicitly also yields matching algorithms for the sublanguages $\mathcal{FL}_\bot$ and $\mathcal{FL}_\neg$: applied to an $\mathcal{FL}_\bot$- or $\mathcal{FL}_\neg$-matching problem $\mathcal{P}$, $\mathsf{match}_{\mathcal{ALN}}$ computes the least matcher to $\mathcal{P}$ in the relevant sublanguage iff $\mathcal{P}$ is solvable or otherwise returns "fail". Especially, $\mathsf{match}_{\mathcal{ALN}}$ coincides with the dedicated $\mathcal{FL}_\bot$- and $\mathcal{FL}_\neg$-matching algorithms presented in [BKBM99] when applied to $\mathcal{FL}_\bot$- or $\mathcal{FL}_\neg$-matching algorithms, respectively.

In Section 4.5.2, we will explain how the theoretical algorithm described above can be implemented. In the following section, we extend the $\mathcal{ALN}$-matching algorithm $\mathsf{match}_{\mathcal{ALN}}$ described above to matching in $\mathcal{ALN}$ under side conditions.

4.3 Matching in $\mathcal{ALN}$ under side conditions

Side conditions are intended to provide a means of restricting the set of admissible solutions to a matching problem further. Formally, we extend $\mathcal{ALN}$-matching problems by side conditions as follows.

Definition 4.3.1 (Matching problems under side conditions)
Let $\mathcal{L} \in \{\mathcal{FL}_\bot, \mathcal{FL}_\neg, \mathcal{ALN}\}$ and let $C \equiv^? D$ be an $\mathcal{L}$-matching problem. An *$\mathcal{L}$-side condition* for a variable $X \in \mathsf{N}_{\text{var}}$ is of the form $X\ \rho^?\ E$, where $\rho \in \{\sqsubseteq, \sqsubset\}$ and E is an $\mathcal{L}$-concept pattern. The side condition is called *subsumption condition* iff $\rho = \sqsubseteq$ and *strict subsumption condition* iff $\rho = \sqsubset$. A substitution σ *satisfies* $X\ \rho^?\ E$ iff $\sigma(X)\ \rho\ \sigma(E)$.

Let $\text{var}(D) =: \{X_1, \ldots, X_\ell\}$ with $|\text{var}(D)| = \ell$. For every $k \in \{1, \ldots, \ell\}$ let $X_k\ \rho_k^?\ E_k$ be an $\mathcal{L}$-side condition with $\text{var}(E_k) \subseteq \{X_1, \ldots, X_{k-1}\}$. Then, $S := \{X_k\ \rho_k^?\ E_K \mid 1 \leq k \leq \ell\})$ is a set of *acyclic* $\mathcal{L}$-side conditions and $(C \equiv^? D, S)$ an *$\mathcal{L}$-matching problem under acyclic side conditions*. A *matcher* to $(C \equiv^? D, S)$ is a matcher of $C \equiv^? D$ satisfying every side condition in S. □

If all side conditions are subsumption conditions, we speak of 'matching under subsumption conditions'. Whenever the underlying DL $\mathcal{L}$ is clear, we may speak of 'side condition' instead of $\mathcal{L}$-side condition and so on.

Matching under side conditions is more complex than matching under subsumption conditions. Firstly, for every $\mathcal{L} \in \{\mathcal{FL}_0, \mathcal{FL}_\bot, \mathcal{FL}_\neg, \mathcal{ALN}\}$, deciding the solvability of $\mathcal{L}$-matching problems modulo equivalence under subsumption conditions is tractable [BBK01] while it has been shown in [BKBM99] by reduction of 3SAT that deciding the solvability of $\mathcal{FL}_0$-matching problems modulo equivalence under acyclic side conditions is NP-hard. The same reduction works for the DLs $\mathcal{FL}_\bot$, $\mathcal{FL}_\neg$, and $\mathcal{ALN}$. Hence, assuming P $\neq$ NP, there is no polynomial time algorithm computing matchers to matching problems under side conditions.

Secondly, as the following example shows, solvable matching problems under side conditions no longer need to have a *least* matcher but rather finitely many *minimal* matchers.

Example 4.3.2 Consider the $\mathcal{FL}_\bot$-matching problem

$$A_1 \sqcap A_2 \equiv^? X_1 \sqcap X_2$$

under the strict subsumption conditions

$$\{X_1 \sqsubset^? \top, X_2 \sqsubset^? X_1\}.$$

The only solutions satisfying both the matching problem and the strict subsumption conditions are $\sigma_1 := \{X_1 \mapsto A_1, X_2 \mapsto A_1 \sqcap A_2\}$ and $\sigma_2 := \{X_1 \mapsto A_2, X_2 \mapsto A_1 \sqcap A_2\}$. Since the assignments for X_2 cannot be chosen more specific, both matchers are minimal. None is a least matcher, however, since the assignments for X_1 are mutually incompatible w.r.t. subsumption. □

As a preliminary step towards matching under acyclic side conditions, we introduce a matching algorithm for subsumption conditions first presented in [Bra00, BBK01]. The algorithm will be utilized in our matching algorithm for acyclic side conditions in Section 4.3.1.

Matching under subsumption conditions

In order to solve $\mathcal{L}$-matching problems under (possibly cyclic) subsumption conditions for $\mathcal{L} \in \{\mathcal{FL}_0, \mathcal{FL}_\neg, \mathcal{ALN}\}$, an algorithm $\mathsf{match}^{\sqsubseteq}_{\mathcal{L}}$ has been proposed in [BBK01] which utilizes the algorithm $\mathsf{match}_{\mathcal{L}}$ for ordinary $\mathcal{L}$-matching problems discussed in Section 4.2.2.

Definition 4.3.3 Let $\mathcal{P} := (C \equiv^? D, S)$ be an $\mathcal{L}$-matching problem under subsumption conditions. For every substitution σ from N_{var} into the set of $\mathcal{ALN}$-concept descriptions, let

$$\mathcal{P}_\sigma := \{C \equiv^? D\} \cup \{\sigma(X) \sqsubseteq^? E \mid X \sqsubseteq^? E \in S\}.$$

Then, the algorithm $\mathsf{match}^{\sqsubseteq}_{\mathcal{L}}$ is defined as follows.

1. $\sigma(X) := \bot$ for all variables X;

2. If $\mathsf{match}_{\mathcal{L}}(\mathcal{P}_\sigma)$ returns "fail", then return "fail";
   else if $\sigma \equiv \mathsf{match}_{\mathcal{L}}(\mathcal{P}_\sigma)$, then return σ;
   else $\sigma := \mathsf{match}_{\mathcal{L}}(\mathcal{P}_\sigma)$; continue with 2. □

For every $\mathcal{L} \in \{\mathcal{FL}_\bot, \mathcal{FL}_\neg, \mathcal{ALN}\}$, it has been shown in [BBK01] that every $\mathcal{L}$-matching problem $\mathcal{P}$ under subsumption conditions is solvable iff $\mathsf{match}^{\sqsubseteq}_{\mathcal{L}}(\mathcal{P})$ does not return "fail". In this case, the least matcher to $\mathcal{P}$ is $\mathsf{match}^{\sqsubseteq}_{\mathcal{L}}(\mathcal{P})$ which can be computed in polynomial time. Note that the $\mathcal{ALN}$-matching algorithm $\mathsf{match}^{\sqsubseteq}_{\mathcal{ALN}}$ coincides with $\mathsf{match}^{\sqsubseteq}_{\mathcal{FL}_\bot}$ when its input is restricted to $\mathcal{FL}_\bot$-matching problems under non-strict subsumption conditions. The same correspondence holds w.r.t. $\mathcal{FL}_\neg$.

Just as $\mathsf{match}_{\mathcal{ALN}}$ is utilized for $\mathsf{match}^{\sqsubseteq}_{\mathcal{ALN}}$, we show in the following section how the algorithm $\mathsf{match}^{\sqsubseteq}_{\mathcal{ALN}}$ can be employed to solve matching problems under acyclic side conditions.

Input: $\mathcal{L}$-matching problem under acyclic side conditions (M, S)
Output: least matcher for (M, S) or "fail".

1. Let $\mathcal{P}_{\sqsubseteq} := (M, S_{\sqsubseteq})$. If $\mathsf{match}_{\mathcal{L}}^{\sqsubseteq}(\mathcal{P}_{\sqsubseteq})$ returns "fail" then return "fail";
2. $k := \ell$; $\sigma := \mathsf{match}_{\mathcal{L}}^{\sqsubseteq}(\mathcal{P})$;
3. If $k = 0$ then return σ;
   If $\sigma(X_k)\ \rho_k\ \sigma(E_k)$ then continue with 5.
4. Guess modification σ' of σ for $X_k \sqsubset^? E_k$; if $\sigma'(E_k) \equiv \sigma(E_k)$ then return "fail";
   $\mathcal{P}' := (\{C \equiv^? D\} \cup \{\sigma'(X_j) \sqsubseteq^? X_j \mid 1 \leq j \leq k-1\}, S_{\sqsubseteq})$;
   If $\mathsf{match}_{\mathcal{L}}^{\sqsubseteq}(\mathcal{P}')$ returns "fail" then return "fail";
   $\sigma := \mathsf{match}_{\mathcal{L}}^{\sqsubseteq}(\mathcal{P}')$
5. $k := k - 1$; continue with 3.

Figure 4.3.1: The algorithm $\mathsf{match}_{\mathcal{L}}^{\sqsubseteq}$

4.3.1 Solving matching problems under acyclic side conditions

In the present section we show how to solve $\mathcal{L}$-matching problems under acyclic side conditions for $\mathcal{L} \in \{\mathcal{FL}_\bot, \mathcal{FL}_\neg, \mathcal{ALN}\}$. As mentioned previously, sublanguages are examined separately because the results obtained for $\mathcal{ALN}$ do not automatically carry over to sublanguages of $\mathcal{ALN}$.

Given that $\mathcal{ALN}$ and its sublanguages provide no existential restrictions, handling cyclic side conditions might appear trivial at first glance: a cyclic relationship of the form $X \sqsubset \forall r.X$ seems to be solvable only by assigning $\bot$ to X. The following example shows that, however, cyclic strict subsumption conditions can have non-trivial solutions.

Example 4.3.4 Let $\mathsf{N}_{\text{con}} := \emptyset$ and $\mathsf{N}_{\text{role}} := \{r, s\}$. Consider the matching problem $\mathcal{M}$ defined by

$$\forall rsrs.\bot \equiv^? \forall rs.X$$

under the side condition

$$X \sqsubset^? \forall rs.X.$$

In this case, the only solution to $\mathcal{M}$ is $\sigma = \{X \mapsto \forall rs.\bot\}$, even though $\sigma' = \{X \mapsto \bot\}$ is a more specific solution of the side condition. □

Similar more complex examples can easily be constructed. In the present work, we restrict our attention to the acyclic case. Our algorithm for matching under acyclic side conditions has been defined for $\mathcal{L} \in \{\mathcal{FL}_\bot, \mathcal{FL}_\neg\}$ in [BBK01]. In the present section, we also discuss the case $\mathcal{L} = \mathcal{ALN}$.

Definition 4.3.5 Let $\mathcal{L} \in \{\mathcal{FL}_\bot, \mathcal{FL}_\neg, \mathcal{ALN}\}$ and let $\mathcal{P} := (\{C \equiv^? D\}, S)$ be an $\mathcal{L}$-matching problem under acyclic side conditions with $\mathrm{var}(D) = \{X_1, \ldots, X_\ell\}$ and $|\mathrm{var}(D)| = \ell$. Denote by $S_{\sqsubseteq}$ the set of subsumption conditions obtained from S by replacing every ρ_i with $\sqsubseteq$. Then, $\mathsf{match}_{\mathcal{L}}^{\sqsubseteq}(\mathcal{P})$ is defined as shown in Figure 4.3.1. □

Applied to input $\mathcal{P}$, the algorithm $\mathsf{match}_{\mathcal{L}}^{\sqsubseteq}$ first calls $\mathsf{match}_{\mathcal{L}}^{\sqsubseteq}(\mathcal{P}_{\sqsubseteq})$. If this yields "fail" then $\mathcal{P}$ is also unsolvable because every solution to $\mathcal{P}$ also solves $\mathcal{P}_{\sqsubseteq}$. Otherwise, the computed

substitution σ solves $C \equiv^? D$ but may still violate some strict subsumption condition. Starting with the last subsumption condition, the algorithm tries to modify σ such that every violated subsumption condition is satisfied.

Assume that the highest index of a violated subsumption condition is k. Since σ solves $X_k \sqsubseteq^? E_k$, $\sigma(X_k) \equiv \sigma(E_k)$. Since $\mathsf{match}_{\mathcal{L}}^{\sqsubseteq}$ always computes the least solution, strict subsumption can only be achieved by making $\sigma(E_k)$ more general. To this end, we (non-deterministically) try to remove one word from one of the role languages representing $\sigma(E_k)$. This is done by modifying the role languages of the values assigned to variables occurring in $\sigma(E_k)$. In order to obtain minimal matchers, our modification changes the original substitution σ as little as possible.

The new substitution σ' obtained thus satisfies $X_k \sqsubset^? E_k$ and preserves all subsumption conditions of variables with index greater or equal k. However, the other subsumption conditions (even the non-strict versions) as well as the original matching problem need no longer be solved by σ'. To correct this effect, $\mathsf{match}_{\mathcal{L}}^{\sqsubseteq}$ is used to compute the least substitution that solves $\mathcal{P}_{\sqsubseteq}$ *and* subsumes σ'. The second condition preserves the validity of the strict subsumption conditions from index k to ℓ.

Clearly, the key point left open in Definition 4.3.5 is to specify exactly how σ is modified to σ'. Since the relevant strategy depends on the choice of $\mathcal{L}$, we discuss modifications in detail for all DLs in question in the following section.

4.3.2 How to guess modifications

In Section 4.2.1, a representation of concept descriptions based on formal languages has been introduced. Based on this representation, we now define modifications as operations on formal languages. In the following, let $\mathcal{P} := (M, S)$ be an $\mathcal{L}$-matching problem ($\mathcal{L} \in \{\mathcal{FL}_\bot, \mathcal{FL}_\neg, \mathcal{ALN}\}$) under acyclic side conditions with $M = \{C \equiv^? D\}$, $\mathrm{var}(D) = \{X_1, \ldots, X_\ell\}$ and $|\mathrm{var}(D)| = \ell$. We begin by introducing modifications in the smallest sublanguage of interest, $\mathcal{FL}_\bot$.

Modifications in $\mathcal{FL}_\bot$

By the following definition, modifications in $\mathcal{FL}_\bot$ are introduced formally. To simplify our notation, denote the set of primitive concepts by $\mathsf{P} := \{\bot\} \cup \mathsf{N}_{\mathrm{con}}$.

Definition 4.3.6 (Modification in $\mathcal{FL}_\bot$)
Let $\mathcal{P}$ be an $\mathcal{FL}_\bot$-matching problem under acyclic subsumption conditions. Let σ be a reduced substitution solving $(M, S_{\sqsubseteq})$. Let k be the highest index with $\rho_k = \sqsubset$ and $\sigma(X_k) \equiv \sigma(E_k)$. A *modification* σ' of σ is defined by executing one of the following steps:

- $\bot$*-modification*
  (Non-deterministically) guess one word $\hat{u} \in \sigma(X_k)|_\bot$. For all $j \in \{1, \ldots, k-1\}$, compute
  $$W_\bot^j := \bigcup_{w \in E_k|_{X_j}} w^{-1} \cdot \{\hat{u}\}$$
  Thus, $W_\bot^j$ contains all suffixes of $\hat{u}$ which yield $\hat{u}$ in the product $E_k|_{X_j} \cdot \sigma(X_j)|_\bot$. Define σ' by specifying the relevant role languages $\sigma'(X_j)|_P$ for all $P \in \mathsf{P}$.
  1. $\sigma'(X_j)|_\bot := (\sigma(X_j)|_\bot \setminus W_\bot^j) \cup (\sigma(X_j)|_\bot \cap W_\bot^j) \cdot \mathsf{N}_{\mathrm{role}}$
  2. For all $P \in \mathsf{N}_{\mathrm{con}}$, define: $\sigma'(X_j)|_P := \sigma(X_j)|_P \cup (\sigma(X_j)|_\bot \cap W_\bot^j)$

- N_{con}-*modification*
  (Non-deterministically) guess one atomic concept $P \in \mathsf{N}_{\text{con}}$. For P, guess one word $\hat{u} \in \sigma(X_k)|_P$. Using $\hat{u}$, for all $j \in \{1, \ldots, k-1\}$ compute $W_P^j := \bigcup_{w \in E_k|_{X_j}} w^{-1} \cdot \{\hat{u}\}$. Then define:

  1. $\sigma'(X_j)|_P := \sigma(X_j)|_P \setminus W_P^j$; and
  2. $\sigma'(X_j)|_Q := \sigma(X_j)|_Q$ for all $Q \in \mathsf{P} \setminus \{P\}$. □

In case of a $\bot$-modification, where w is picked from the role language corresponding to the $\bot$-concept, the removal of some word v from some role language implicitly removes every continuation vv' of v. To avoid this, every word in $\{v\} \cdot \mathsf{N}_{\text{role}}$ is put back for every v removed. In addition, since v is also implicitly removed from role languages corresponding to atomic concepts, it is also transferred to these role languages to ensure that the computed substitution is as specific as possible. Note that $W_\bot^j \neq \emptyset$ implies $\sigma(X_j)|_\bot \succ \sigma'(X_j)|_\bot$.

In an N_{con}-modification, w is picked from a role language corresponding to some atomic concept $P \in \mathsf{N}_{\text{con}}$. In this case, removing certain words from role languages of the variables in E suffices to obtain a minimal modification.

Note that modifications need not be defined for the first subsumption condition which contains no variables. We give an example in order to show that (i) modifications deleting only one word do not always suffice and (ii) matching in Step 4 of the algorithm $\mathsf{match}_{\mathcal{L}}^{\sqsubseteq}$ is necessary. In order to simplify notation, denote value restrictions over singleton sets without parenthesis, e.g., write $\forall rr.P$ instead of $\forall\{rr\}.P$.

Example 4.3.7 Let $\mathsf{N}_{\text{con}} = \{P\}$ and $\mathsf{N}_{\text{role}} = \{r, s\}$. Consider the matching problem

$$\forall\{rrr, rrs, rs, srr\}.\bot \sqcap \forall\{rr, sr\}.P \equiv^? \forall rr.X_1 \sqcap \forall sr.X_2 \sqcap \forall r.X_3 \sqcap \forall r.X_4$$

under the following set of subsumption conditions.

$$\begin{aligned} &\{X_1 \sqsubseteq^? \forall\{r, s\}.\bot, \\ &\ X_3 \sqsubset^? \forall\{rs, s\}.X_1 \sqcap \forall r.X_2, \\ &\ X_4 \sqsubset^? \forall s.\bot \sqcap \forall\{\varepsilon, r\}.X_3\} \end{aligned}$$

Executing algorithm $\mathsf{match}_{\mathcal{FL}_\bot}^{\sqsubseteq}$ yields as initial solution σ in Step 2

$$\begin{aligned} &\{X_1 \mapsto \forall\{r, s\}.\bot \sqcap \forall\{\varepsilon\}.P, \\ &\ X_2 \mapsto \forall r.\bot \sqcap \forall\{\varepsilon\}.P, \\ &\ X_3 \mapsto \forall\{rr, rs, s\}.\bot \sqcap \forall r.P, \\ &\ X_4 \mapsto \forall\{rr, rs, s\}.\bot \sqcap \forall r.P\}. \end{aligned}$$

which violates the third subsumption condition, as the test in Step 3 shows: $\sigma(X_4)$ is equivalent to $\sigma(\forall s.\bot \sqcap \forall\{\varepsilon, r\}.X_3)$. In Step 4, we choose a $\bot$-modification and pick the word rs from the role language $\sigma(X_4)|_\bot = \{rr, rs, s\}$. Hence, $W_\bot^3 = \{rs, s\}$. Thus, rs and s must be changed in $\sigma(X_3)|_\bot$. The modified solution σ' now yields

$$\sigma'(X_3) = \forall\{rr, rsr, rss, sr, ss\}.\bot \sqcap \forall\{r, rs, s\}.P,$$

while the other variables remain unchanged. We find that σ' already solves the matching problem $\mathcal{P}'$ in Step 4, implying that $\mathsf{match}_{\mathcal{FL}_\bot}^{\sqsubseteq}(M')$ yields σ'.

In the second iteration we find in Step 3 that the second subsumption condition is violated, since $\sigma(X_3)$ is equivalent to $\sigma(\forall\{rs, s\}.X_1 \sqcap \forall r.X_2)$. We choose an N_{con}-modification and pick the word rs from the role language $\sigma(X_3)|_P = \{r, rs, s\}$ corresponding to P. This yields $W_A^1 = \{\varepsilon\}$ and $W_P^2 = \{s\}$. Nevertheless, the role language $\sigma(X_2)|_P = \{\varepsilon\}$ does not contain s, while $\sigma(X_1)|_P = \{\varepsilon\}$ contains ε. We therefore have

$$\sigma'(X_1) = \forall\{r, s\}.\bot,$$

while the other variables remain unchanged. Again σ' solves the matching problem $\mathcal{P}'$ in Step 4, so that we have σ' as new substitution σ. In the third iteration, we now find in Step 3 that the first subsumption condition holds, so that the final result is the following.

$$\begin{aligned}\{&X_1 \mapsto \forall\{r, s\}.\bot,\\ &X_2 \mapsto \forall r.\bot \sqcap \forall\{\varepsilon\}.P,\\ &X_3 \mapsto \forall\{rr, rsr, rss, sr, ss\}.\bot \sqcap \forall\{r, rs, s\}.P,\\ &X_4 \mapsto \forall\{rr, rs, s\}.\bot \sqcap \forall r.P\}\end{aligned}$$ □

The above example might give rise to the question whether solving the matching problem $\mathcal{P}'$ in Step 4 of every iteration of the algorithm $\mathsf{match}^{\sqsubseteq}_{\mathcal{FL}_\bot}$ is necessary at all. The following example shows that this step is needed.

Example 4.3.8 We examine the matching problem

$$\forall\{rrr, rrs\}.\bot \sqcap \forall rr.P \equiv^? \forall rr.X_1 \sqcap \forall r.X_2 \sqcap X_3 \sqcap X_4$$

under the following set of subsumption conditions.

$$\begin{aligned}\{&X_3 \sqsubset^? \forall rr.X_1 \sqcap \forall r.X_2,\\ &X_4 \sqsubset^? X_3\}\end{aligned}$$

Executing algorithm $\mathsf{match}^{\sqsubseteq}_{\mathcal{FL}_\bot}$ again computes an initial solution σ in Step 2, yielding:

$$\begin{aligned}\{&X_1 \mapsto \forall\{r, s\}.\bot \sqcap P,\\ &X_2 \mapsto \forall\{rr, rs\}.\bot \sqcap \forall r.P,\\ &X_3 \mapsto \forall\{rrr, rrs\}.\bot \sqcap \forall\{rr\}.P,\\ &X_4 \mapsto \forall\{rrr, rrs\}.\bot \sqcap \forall\{rr\}.P\}.\end{aligned}$$

Clearly, in Step 3 we find the second subsumption condition violated, making it necessary to modify $\sigma(X_3)$. Nevertheless, for the initial solution σ the first subsumption condition is also violated because $\sigma(X_3)$ is equivalent to $\sigma(\forall rr.X_1 \sqcap \forall r.X_2)$. As a consequence, any successful modification will result in a substitution σ' with $\sigma'(X_3) \not\equiv \sigma'(\forall rr.X_1 \sqcap \forall r.X_2)$. Hence, σ' can be no solution to the matching problem $\mathcal{P}'$ in Step 4. □

The above examples may suffice to show how modifications in $\mathcal{FL}_\bot$, and thus the algorithm $\mathsf{match}^{\sqsubseteq}_{\mathcal{FL}_\bot}$, work. In the following, the corresponding definitions for modifications in $\mathcal{FL}_\neg$ and $\mathcal{ALN}$ are introduced. Soundness and completeness for $\mathcal{FL}_\bot$, $\mathcal{FL}_\neg$, and $\mathcal{ALN}$ are proved in Section 4.3.3.

Modifications in $\mathcal{FL}_\neg$

In contrast to $\mathcal{FL}_\bot$, inconsistencies in $\mathcal{FL}_\neg$ can not only be introduced by the $\bot$-concept, but also by interaction between an atomic concept P and its negation $\neg P$. Nevertheless,

since only reduced substitutions occur in the algorithm $\mathsf{match}_{\mathcal{L}}^{\sqsubset}$, such interactions have to be observed only in the second step of $\bot$-modifications: there, the intersection $\sigma(X_j)|_\bot \cap W_\bot^j$ must *not* be added to both $\sigma'(X_j)_P$ and $\sigma'(X_j)_{\neg P}$ for any $P \in \mathsf{N}_{\mathrm{con}}$ because this would render the removal from $\sigma(X_j)|_\bot$ useless. Instead, we add only a non-deterministically chosen subset of $\sigma(X_j)|_\bot \cap W_\bot^j$. Apart from this, it suffices to extend $\mathsf{N}_{\mathrm{con}}$-modifications to negated atomic concepts without further changes. We thus obtain the following definition for modifications in $\mathcal{FL}_\neg$.

Again, for the sake of a simpler notation, denote the set of primitive concepts by $\mathsf{P} := \{\bot\} \cup \mathsf{N}_{\mathrm{con}} \cup \{\neg P \mid P \in \mathsf{N}_{\mathrm{con}}\}$.

Definition 4.3.9 (Modification in $\mathcal{FL}_\neg$)
Let $\mathcal{P}$ be an $\mathcal{FL}_\neg$-matching problem under acyclic subsumption conditions. Let σ be a reduced substitution solving $(M, S_\sqsubseteq)$. Let k be the highest index with $\rho_k = \sqsubset$ and $\sigma(X_k) \equiv \sigma(E_k)$. A *modification* σ' of σ is defined by executing one of the following steps:

- $\bot$-*modification*
  (Non-deterministically) guess one word $\hat{u} \in \sigma(X_k)|_\bot$. For all $j \in \{1, \ldots, k-1\}$, compute
  $$W_\bot^j := \bigcup_{w \in E_k|_{X_j}} w^{-1} \cdot \{\hat{u}\}$$
  Thus, $W_\bot^j$ contains all suffixes of $\hat{u}$ yielding $\hat{u}$ in the product $E_k|_{X_j} \cdot W_\bot^j$. Define σ' by specifying the relevant role languages $\sigma'(X_j)|_P$ for all $P \in \mathsf{P}$.
  1. $\sigma'(X_j)|_\bot := (\sigma(X_j)|_\bot \setminus W_\bot^j) \cup (\sigma(X_j)|_\bot \cap W_\bot^j) \cdot \mathsf{N}_{\mathrm{role}}$
  2. For all $P \in \mathsf{P} \setminus \{\bot\}$, (non-deterministically) choose a subset $\hat{W}_P^j \subseteq \sigma(X_j)|_\bot \cap W_\bot^j$. Then define: $\sigma'(X_j)|_P := \sigma(X_j)|_P \cup \hat{W}^j$
- $\mathsf{N}_{\mathrm{con}}$-*modification*
  (Non-deterministically) guess one (negated) atomic concept $P \in \mathsf{P} \setminus \{\bot\}$. For P, guess one word $\hat{u} \in \sigma(X_k)|_P$. For all $j \in \{1, \ldots, k-1\}$ compute $W_P^j := \bigcup_{w \in E_k|_{X_j}} w^{-1} \cdot \{\hat{u}\}$. Then define:
  1. $\sigma'(X_j)_P := \sigma(X_j)_P \setminus W_P^j$; and
  2. $\sigma'(X_j)_Q := \sigma(X_j)_Q$ for all $Q \in \mathsf{P} \setminus \{\bot, P\}$.

Modifications in $\mathcal{ALN}$

In $\mathcal{ALN}$, an additional source of inconsistencies is introduced by interactions between number restrictions, e.g., $(\leqslant 1\, r) \sqcap (\geqslant 2\, r)$. Similar to the case of $\mathcal{FL}_\neg$, these interactions must be taken into account only in the second step of $\bot$-modifications. Let $\mathsf{P} := \{\bot\} \cup \mathsf{N}_{\mathrm{con}} \cup \{\neg P \mid P \in \mathsf{N}_{\mathrm{con}}\} \cup \mathsf{N}_{\leqslant} \cup \mathsf{N}_{\geqslant}$.

Definition 4.3.10 (Modification in $\mathcal{ALN}$)
Let $\mathcal{P}$ be an $\mathcal{ALN}$-matching problem under acyclic subsumption conditions. Let σ be a reduced substitution solving $(M, S_\sqsubseteq)$. Let k be the highest index with $\rho_k = \sqsubset$ and $\sigma(X_k) \equiv \sigma(E_k)$. A *modification* σ' of σ is defined by executing one of the following steps:

- $\bot$-*modification*
  (Non-deterministically) guess one word $\hat{u} \in \sigma(X_k)|_\bot$. For all $j \in \{1, \ldots, k-1\}$, compute
  $$W_\bot^j := \bigcup_{w \in E_k|_{X_j}} w^{-1} \cdot \{\hat{u}\}$$

Thus, $W^j_\bot$ contains all suffixes of $\hat{u}$ yielding $\hat{u}$ in the product $E_k|_{X_j} \cdot W^j_\bot$. Define σ' by specifying the relevant role languages $\sigma'(X_j)|_H$ for all $H \in \mathsf{P}$.

1. $\sigma'(X_j)|_\bot := (\sigma(X_j)|_\bot \setminus W^j_\bot) \cup (\sigma(X_j)|_\bot \cap W^j_\bot) \cdot \mathsf{N}_{\text{role}}$
2. For all $P \in \mathsf{P} \setminus \{\bot\}$, (non-deterministically) choose a subset $\hat{W}^j \subseteq \sigma(X_j)|_\bot \cap W^j_\bot$. Then define: $\sigma'(X_j)|_P := \sigma(X_j)|_P \cup \hat{W}^j$

- N_{con}-*modification*
(Non-deterministically) guess one (negated) atomic concept or number restriction $P \in \mathsf{P} \setminus \{\bot\}$. For P, guess one word $\hat{u} \in \sigma(X_k)|_P$. For all $j \in \{1, \ldots, k-1\}$ compute $W^j_P := \bigcup_{w \in E_k|_{X_j}} w^{-1} \cdot \{\hat{u}\}$. Then define:

  1. $\sigma'(X_j)_P := \sigma(X_j)_P \setminus W^j_P$; and
  2. $\sigma'(X_j)_Q := \sigma(X_j)_Q$ for all $Q \in \mathsf{P} \setminus \{\bot, P\}$.

In the next section we show that the algorithm $\mathsf{match}^{\sqsubset}_{\mathcal{L}}$ is correct for every $\mathcal{L} \in \{\mathcal{FL}_\bot, \mathcal{FL}_\neg, \mathcal{ALN}\}$. Termination of $\mathsf{match}^{\sqsubset}_{\mathcal{L}}$ is an immediate consequence of the fact that $\mathsf{match}^{\sqsubset}_{\mathcal{L}}$ always terminates and that a finite number of matching problems under non-strict subsumption conditions are solved.

4.3.3 Soundness and completeness

With a formal definition of modifications, we are prepared to prove soundness and completeness of the algorithm $\mathsf{match}^{\sqsubset}_{\mathcal{L}}$. We first address the case $\mathcal{L} = \mathcal{FL}_\bot$.

Soundness and completeness in $\mathcal{FL}_\bot$

We begin by showing that the modification strategy defined in Definition 4.3.6 does produce a strict solution for the relevant subsumption condition $X_k \sqsubset^? E_k$. Moreover, we show that modifications can be constructed so as to remain more specific than a given solution to the relevant matching problem.

Lemma 4.3.11 *(Strictness and minimality of modifications in $\mathcal{FL}_\bot$)*
Let $\mathcal{P}$ be an $\mathcal{FL}_\bot$-matching problem under acyclic subsumption conditions. Let σ be a reduced substitution solving $(M, S_\sqsubset)$ and τ a reduced solution to $\mathcal{P}$ with $\sigma \sqsubset \tau$. Let $k \in \{2, \ldots, \ell\}$ with $\rho_k = \sqsubset$ and $\sigma(X_k) \equiv \sigma(E_k)$. Then there exists a modified substitution σ' solving $(M, S_\sqsubset)$ with

1. *$\sigma \sqsubset \sigma'$;*
2. *$\sigma' \sqsubseteq \tau$ and $\sigma'(X_k) \sqsubset \sigma'(E_k)$.*

Proof. 1. We show that every possible modification yields the desired property.

$\bot$-Modification: for every $j \in \{1, \ldots, \ell\}$, $\sigma(X_j)|_\bot$ is modified by replacing every word $w \in W^j_\bot$ by the continuations $\{w\} \cdot \mathsf{N}_{\text{role}}$. If $W^j_\bot = \emptyset$ then $\sigma'(X_j)|_P = \sigma(X_j)|_P$ for all $P \in \mathsf{P}$, implying $\sigma'(X_j) = \sigma(X_j)$. Otherwise, $\sigma(X_j)|_\bot \succ \sigma'(X_j)|_\bot$. For every $P \in \mathsf{P} \setminus \{\bot\}$, all words added to $\sigma'(X_j)|_P$ are contained in $\sigma(X_j)|_\bot$. Hence,

$$\sigma'(X_j)|_P \subseteq \sigma(X_j)|_P \cup \sigma(X_j)|_\bot \cdot \mathsf{N}^*_{\text{role}}$$

for all $P \in \mathsf{P} \setminus \{\bot\}$, implying $\sigma(X_j) \sqsubset \sigma'(X_j)$ by Definition 4.2.9. By construction, the second case applies to at least one j, implying $\sigma \sqsubset \sigma'$.

N_{con}-Modification: then the only difference between σ and σ' is the deletion of words in one role language which, again by our characterization of subsumption, immediately implies the claim.

2. It suffices to find a guessing strategy for a modification σ' with $\sigma' \sqsubseteq \tau$ and $\sigma'(X_k) \sqsubset \sigma'(E_k)$. To this end, two cases are distinguished.

Case 1: $\sigma(E_k)$ and $\tau(E_k)$ disagree on the $\bot$-concept in the sense that

$$\sigma(X_k)|_\bot \cdot \mathsf{N}^*_{\text{role}} = \sigma(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}} \supset \tau(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}}.$$

Thus, there is some $\hat{u} \in \sigma(X_k)|_\bot$ missing in $\tau(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}}$. This implies $\hat{u} \notin E_k|_\bot$. Construct σ' by a $\bot$-modification choosing $\hat{u}$. By definition, we obtain σ' with

$$\sigma'(X_j)|_\bot = (\sigma(X_j)|_\bot \setminus W^j_\bot) \cup (\sigma(X_j)|_\bot \cap W^j_\bot) \cdot \mathsf{N}_{\text{role}} \qquad (*)$$

for every $j \in \{1, \ldots, k-1\}$, where $W^j_\bot = \bigcup_{w \in E_k|_{X_j}} w^{-1} \cdot \{\hat{u}\}$. By definition of $W^j_\bot$,

$$\sigma'(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}} = \sigma(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}} \setminus \bigcup_{j=1}^{k-1} E_k|_{X_j} \cdot W^j_\bot.$$

The word $\hat{u}$ occurs in $\sigma(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}}$ and, as $\hat{u} \notin E_k|_\bot$, in at least one product $E_k|_{X_j} \cdot W^j_\bot$, implying $\hat{u} \notin \sigma'(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}}$. Hence, we obtain

$$\sigma'(X_k)|_\bot \cdot \mathsf{N}^*_{\text{role}} \supset \sigma'(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}}.$$

We already know $\sigma \sqsubseteq \sigma'$ from (1) and $\sigma'(X_k) = \sigma(X_k)$, implying $\sigma'(X_k) \sqsubseteq \sigma'(E_k)$. Together with the above strict inclusion, $\sigma'(X_k) \sqsubset \sigma'(E_k)$ as required.

It remains to show $\sigma' \sqsubseteq \tau$, i.e., $\sigma'(X_j) \sqsubseteq \tau(X_j)$ for every $j \in \{1, \ldots, k-1\}$. If not $\sigma'(X_j) = \sigma(X_j)$ then, by construction of σ', $\sigma(X_j)|_\bot \succ \sigma'(X_j)|_\bot$. As $\sigma \sqsubset \tau$ and σ disagrees with τ on the $\bot$-concept, also $\sigma(X_j)|_\bot \succ \tau(X_j)|_\bot$. More precisely, every word in $\sigma(X_j)|_\bot \cap W^j_\bot$ can be replaced by some extensions contained in $\tau(X_j)|_\bot$. But since $(\sigma(X_j)|_\bot \cap W^j_\bot) \cdot \mathsf{N}_{\text{con}}$ (i) is the largest set of such extensions and (ii) contains the shortest possible extensions, we find $\sigma'(X_j)|_\bot \succ \tau(X_j)|_\bot$. Since $\sigma(X_j) \sqsubset \tau(X_j)$ and $\sigma(X_j)|_\bot \succ \tau(X_j)|_\bot$ we also know by characterization of subsumption that, for every $P \in \mathsf{N}_{\text{con}}$,

$$\tau(X_j)|_P \subseteq \sigma(X_j)|_P \cup \sigma(X_j)|_\bot \cdot \mathsf{N}^*_{\text{role}}.$$

As a consequence of $(*)$,

$$\sigma'(X_j)|_\bot \cdot \mathsf{N}^*_{\text{role}} = (\sigma(X_j)|_\bot \cdot \mathsf{N}^*_{\text{role}}) \setminus (\sigma(X_j)|_\bot \cap W^j_\bot).$$

Moreover, as always $\sigma'(X_j)|_P = \sigma(X_j)|_P \cup (\sigma(X_j)|_\bot \cap W^j_\bot)$, we may replace σ by σ' without changing the right-hand side, obtaining

$$\tau(X_j)|_P \subseteq \sigma'(X_j)|_P \cup \sigma'(X_j)|_\bot \cdot \mathsf{N}^*_{\text{role}}.$$

Together with $\sigma(X_j)|_\bot \succ \tau(X_j)|_\bot$, this implies $\sigma'(X_j) \sqsubset \tau(X_j)$. As $\sigma'(X_j) = \sigma(X_j)$ for every $j \in \{k, \ldots, \ell\}$, trivially implying $\sigma'(X_j) \sqsubseteq \sigma(\tau)$, we finally have $\sigma' \sqsubseteq \tau$.

Case 2: $\sigma(E_k)$ and $\tau(E_k)$ agree on the $\bot$-concept in the sense that

$$\sigma(X_k)|_\bot \cdot \mathsf{N}^*_{\text{role}} = \sigma(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}} = \tau(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}}.$$

As $\sigma(E_k) \sqsubset \tau(E_k)$, this implies that there is a $P \in \mathsf{N}_{\text{con}}$ and a word $\hat{u} \in \sigma(X_k)|_P$ such that $\hat{u} \notin \tau(E_k)|_P$. Construct σ' by N_{con}-modification choosing P and the above $\hat{u}$. For σ' it holds that

$$\sigma'(E_k)|_P = \sigma(E_k)|_P \setminus \bigcup_{j=1}^{k-1} E_k|_{X_j} \cdot W^j_P.$$

The word $\hat{u}$ occurs in $\sigma(E_k)|_P$ since

$$\sigma(X_k)|_P = \mathrm{pf}(\sigma(E_k)|_P) \subseteq \sigma(E_k)|_P,$$

and occurs in at least one product $E_k|_{X_j} \cdot W_\perp^j$ because otherwise either $\hat{u} \notin \sigma(E_k)|_P$ or $\hat{u} \in E_k|_P$. Thus, $\hat{u} \notin \sigma'(E_k)|_P$. As σ' is obtained from σ only by removing words from role languages $\sigma(X_j)|_P$ for all $j \in \{1, \ldots, k-1\}$, we have $\sigma'(X_k) \sqsubset \sigma'(E_k)$. It remains to show $\sigma' \sqsubseteq \tau$. For every $j \in \{1, \ldots, k-1\}$, we again distinguish two cases: if $\sigma(X_j) \equiv \sigma'(X_j)$ then trivially $\sigma'(X_j) \sqsubseteq \tau(X_j)$ for all j, implying the proposition. Otherwise, $\sigma(X_j)|_\perp = \sigma'(X_j)|_\perp$, since we have no $\perp$-modification, and therefore $\sigma(X_j)|_Q = \sigma'(X_j)|_Q$ for every $Q \in \mathsf{N}_{\mathrm{con}} \backslash \{P\}$ and $\sigma'(X_j)|_P = \sigma(X_j)|_P \backslash W_P^j$ with $\sigma(X_j)|_P \cap W_P^j \neq \emptyset$. For every j, $\tau(X_j)|_P$ contains no word from W_P^j because otherwise $\hat{u} \in \tau(E_k)|_P$. Hence, $\sigma'(X_j)|_P \subseteq \tau(X_j)|_P$ for every j. The fact that $\sigma'(X_j)|_Q$ coincides with $\sigma(X_j)|_Q$ for every $Q \neq P$ and that $\sigma \sqsubset \tau$ implies the proposition $\sigma' \sqsubseteq \tau$. □

We have shown that the modified substitution σ' can be constructed minimal in the sense that no other solution τ at the same time (i) lies between σ and σ' in respect to the strict ordering $\sqsubset$ on substitutions, and (ii) satisfies the kth side condition. This minimality property justifies that no modification makes the left-hand side $\sigma(X_k)$ more specific when dealing with a subsumption condition $X_k \sqsubset^? E_k$.

We are now prepared to prove soundness of the algorithm $\mathsf{match}^{\sqsubseteq}_{\mathcal{FL}_\perp}$. To this end, we need to make sure that subsumption conditions remain valid after modification.

Since more than one step in the computation of $\mathsf{match}^{\sqsubseteq}_{\mathcal{L}}$ is considered, some notation is required in preparation. Denote by σ_{new} the new solution σ of $\mathcal{P}'$ computed in Step 4 of the algorithm. Moreover, denote by σ_0 the initial solution computed in Step 2. In the tth iteration of Steps 3 to 5, $t = 0, 1, \ldots$, denote by σ_t the substitution σ occurring in Steps 3 and 4 and by σ'_t the substitution σ' occurring in Steps 4 and 5. Note that always $\sigma_{\mathrm{new}} = \sigma_{t+1}$. By definition, the solution returned by $\mathsf{match}^{\sqsubseteq}_{\mathcal{FL}_\perp}$ upon input $\mathcal{P}$ is σ_ℓ.

Lemma 4.3.12 *(Soundness)*

1. *For every modification of σ yielding σ', it holds that if $\mathsf{match}^{\sqsubseteq}_{\mathcal{FL}_\perp}(\mathcal{P}')$ succeeds in Step 4 of the algorithm then $\sigma(X_j) \equiv \sigma'(X_j) \equiv \sigma_{\mathrm{new}}(X_j)$ for every $j \in \{k, \ldots, \ell\}$ and $\sigma' \sqsubseteq \sigma_{\mathrm{new}}$;*
2. *for every $t \in \{1, \ldots, \ell\}$, σ_t satisfies Subsumption Conditions $\ell - t + 1$ to ℓ;*
3. *if $\mathsf{match}^{\sqsubseteq}_{\mathcal{FL}_\perp}(\mathcal{P})$ returns the substitution σ then σ solves $\mathcal{P}$.*

Proof. 1. For every $j \in \{k, \ldots, \ell\}$, $\sigma(X_j) = \sigma'(X_j)$ by Definition 4.3.6. As X_k to X_ℓ do not occur in the additional matching problems that distinguish $\mathcal{P}'$ from $\mathcal{P}$, and as σ_{new} is the least solution to $\mathcal{P}'$ and a solution to $\mathcal{P}$, it is easy to see that $\sigma'(X_j) \equiv \sigma_{\mathrm{new}}(X_j)$.

For every $j \in \{1, \ldots, k-1\}$, $\sigma'(X_j) \sqsubseteq \sigma_{\mathrm{new}}(X_j)$ holds because of the additional matching problems modulo subsumption $\{\sigma'(X_j) \sqsubseteq^? X_j | 1 \leq j \leq k-1\}$ in $\mathcal{P}'$. Together with the above, this implies $\sigma' \sqsubseteq \sigma_{\mathrm{new}}$ as required.

2. Let $\ell \geq 1$ since otherwise the proposition trivially holds. We show for every $t \in \{1, \ldots, \ell\}$ that σ_t satisfies Subsumption Conditions $\ell - t + 1$ to ℓ. Proof by induction over t.

   - $t = 1$: The case $\rho_1 = \sqsubseteq$ is trivial. Moreover, if $\sigma_0(X_\ell) \sqsubset \sigma_0(E_k)$ then $\sigma_1 = \sigma_0$, implying the proposition. Otherwise, by Lemma 4.3.11, $\sigma'_0(X_\ell) \sqsubset \sigma'_0(E_\ell)$. As

shown above in (1), this implies $\sigma_1(X_\ell) = \sigma'_0(X_\ell)$. Moreover, by (1), $\sigma'_0 \sqsubseteq \sigma_1$, yielding

$$\sigma_1(X_\ell) = \sigma'_0(X_\ell) \sqsubset \sigma'_0(E_\ell) \sqsubseteq \sigma_1(E_\ell).$$

- $t > 1$: By induction hypothesis, σ_t satisfies Subsumption Conditions $\ell - t + 1$ to ℓ. If $\rho_t = \sqsubset$ then $\sigma_{t+1} = \sigma_t$, implying the proposition. Otherwise, by Lemma 4.3.11, $\sigma'_t(X_{\ell-t}) \sqsubset \sigma'_t(E_{\ell-t})$. By (1), $\sigma_t(X_k) = \sigma'_t(X_k) = \sigma_{t+1}(X_k)$ for all $k > \ell - t$, implying that σ_{t+1} still satisfies Subsumption Conditions $\ell - t + 1$ to ℓ. Moreover, together with the second claim from (1), we obtain $\sigma_{t+1}(X_{\ell-t}) \sqsubset \sigma_{t+1}(E_{\ell-t})$ as seen for the case $(t = 1)$.

3. Immediate consequence of (2).

□

In order to prove completeness, we have to show that $\mathsf{match}^{\sqsubseteq}_{\mathcal{FL}_\bot}(\mathcal{P})$ successfully returns a solution if the input matching problem $\mathcal{P}$ is solvable.

Lemma 4.3.13 *(Completeness)*
Let τ be a reduced solution to $\mathcal{P}$. Then $\mathsf{match}^{\sqsubseteq}_{\mathcal{FL}_\bot}(\mathcal{P})$ returns a substitution σ solving $\mathcal{P}$.

PROOF. If $\mathsf{match}^{\sqsubseteq}_{\mathcal{FL}_\bot}(\mathcal{P})$ succeeds then, by Lemma 4.3.12, σ_ℓ satisfies $\mathcal{P}$. Hence, it suffices to show that every modification step necessary for the computation of σ_ℓ succeeds. If σ_0 solves $\mathcal{P}$ then the proposition holds trivially. Otherwise, since τ also satisfies $(M, S_\sqsubseteq)$ of which σ_0 is the least solution, $\sigma_0 \sqsubseteq \tau$. By Lemma 4.3.11, this guarantees the existence of a modification σ'_0 with $\sigma'_0 \sqsubseteq \tau$. This implies that τ also solves $\mathcal{P}'$, implying that $\sigma_{\text{new}} = \sigma_1$ is defined and, by minimality, $\sigma_1 \sqsubseteq \tau$. It is easy to see that, iterating the above argument, we obtain that the computation of σ_ℓ succeeds and that $\sigma_\ell \sqsubseteq \tau$. □

As a consequence of the previous lemma, every minimal matcher (w.r.t. subsumption of substitutions) is computed by an appropriate run of $\mathsf{match}^{\sqsubseteq}_{\bot}$. This can be readily seen by replacing τ by a minimal matcher to $\mathcal{P}$.

Soundness and completeness in $\mathcal{FL}_\neg$

In $\mathcal{FL}_\neg$-concept descriptions, inconsistencies can additionally be introduced by words occurring in role languages referring to an atomic concept and to its negation. Recall that our notation $C|_\bot$ allows for this by referring to the reduced version of C where all inconsistencies are explicit.

In Lemma 4.3.11 we could prove for every reduced substitution σ that $\sigma \sqsubseteq \sigma'$ for every possible modification σ'. In case of $\mathcal{FL}_\neg$, this is no longer possible, because we depend stronger on the properties of a strict solution τ. In the following lemma we therefore begin by specifying a guessing strategy relative to τ.

Lemma 4.3.14 *(Strictness and minimality of modifications in $\mathcal{FL}_\neg$)*
Let $\mathcal{P} := (M, S)$ be an $\mathcal{FL}_\neg$-matching problem under subsumption conditions over N_{con}, N_{role}, and $\mathsf{N}_{\text{var}} =: \{X_1, \ldots, X_\ell\}$. Let $S = \{X_j \ \rho_j^? \ E_j \mid 1 \leq j \leq \ell\}$. Let σ be a reduced substitution solving $(M, S_\sqsubseteq)$. Let $k \in \{2, \ldots, \ell\}$ with $\rho_k = \sqsubset$ and $\sigma(X_k) \equiv \sigma(E_k)$. Let τ be a reduced solution to M with $\sigma \sqsubset \tau$. Then there exists a modified substitution σ' solving $(M, S_\sqsubseteq)$ with

1. *$\sigma \sqsubset \sigma'$;*

2. $\sigma' \sqsubseteq \tau$ *and* $\sigma'(X_k) \sqsubset \sigma'(E_k)$.

PROOF. 1. We show that there exists a modification in accordance with Definition 4.3.9 such that $\sigma \sqsubseteq \sigma'$ and $\sigma'(X_k) \sqsubset \sigma'(E_k)$. To this end, we present a guessing strategy to find an appropriate modification σ', distinguishing two cases.

Case 1: $\sigma(E_k)$ and $\tau(E_k)$ disagree on the $\bot$-concept, i.e.,

$$\sigma(X_k)|_\bot \cdot \mathsf{N}^*_{\text{role}} = \sigma(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}} \supset \tau(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}}.$$

This case is similar to the one for $\mathcal{FL}_\bot$ because σ is reduced. Hence, again construct σ' by a $\bot$-modification, picking one word $\hat{u} \in \sigma(X_k)|_\bot$ missing in $\tau(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}}$. In contrast to the case of $\mathcal{FL}_\bot$, we additionally choose

$$\hat{W}^j_P := \sigma(X_j)|_\bot \cap W^j_\bot \cap \tau(X_j)|_P$$

for every $j \in \{1, \ldots, k-1\}$ and every $P \in \mathsf{P} \setminus \{\bot\}$. As σ' is still reduced, for every j either $\sigma'(X_j) \equiv \sigma(X_j)$ or both $\sigma(X_j)|_\bot \succ \sigma'(X_j)|_\bot$ and

$$\sigma'(X_j)|_P \subseteq \sigma(X_j)|_P \cup \sigma(X_j)|_\bot \cdot \mathsf{N}^*_{\text{role}}$$

for every $P \in \mathsf{P} \setminus \{\bot\}$. Because of $\hat{u}$, the second case applies at least once, yielding $\sigma \sqsubset \sigma'$ by Definition 4.2.9. Note that the above subset relation holds independently of our choice of the sets $\hat{W}^j_P$. Nevertheless, choosing $\hat{W}^j_P$ wrongly compromises the reducedness of σ'.

Case 2: Analogous to the guessing strategy for modifications in $\mathcal{FL}_\bot$. If $\sigma_t(E_k)$ and $\tau(E_k)$ agree on the $\bot$-languages then

$$\sigma(X_k)|_\bot \cdot \mathsf{N}^*_{\text{role}} = \sigma(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}} = \tau(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}}.$$

Thus, there is some concept $P \in \mathsf{P} \setminus \{\bot\}$ and some word $\hat{u} \in \sigma(X_k)|_P$ such that $\hat{u} \notin \tau(E_k)|_P$. Construct σ' by an N_{con}-modification choosing P and $\hat{u}$. Consequently, $\sigma'(X_j)|_Q = \sigma(X_j)|_Q$ for all $Q \in \mathsf{P} \setminus \{P\}$ and $j \in \{1, \ldots, \ell\}$ and also for $Q = P$ and $j \in \{k, \ldots, \ell\}$. Moreover, $\sigma'(X_j)|_P \subseteq \sigma(X_j)|_P$ for all $j \in \{1, \ldots, k-1\}$ with a strict inclusion for at least one index j. Consequently, by Definition 4.2.9, $\sigma'(X_j) \sqsubseteq \sigma(X_j)$ and a strict subsumption for at least one j, implying $\sigma \sqsubset \sigma'$.

2. We show that using the guessing strategy from (1) suffices for our claim. The proof is analogous to the one for $\mathcal{FL}_\bot$.

Case 1: In case of a $\bot$-modification,

$$\sigma'(X_j)|_\bot = (\sigma(X_j)|_\bot \setminus W^j \bot) \cup (\sigma(X_j)|_\bot \cap W^j_\bot) \cdot \mathsf{N}_{\text{role}}$$

for every $j \in \{1, \ldots, k-1\}$, implying

$$\sigma'(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}} = \sigma(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}} \setminus \bigcup_{j=1}^{k-1} E_k|_{X_j} \cdot W^j_\bot.$$

As $\hat{u} \in \sigma(E_k)|_\bot$ and $\hat{u} \notin \tau(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}}$, $\hat{u} \notin E_k|_\bot$ and there is at least one $j \in \{1, \ldots, k-1\}$ with $\hat{u} \in E_k|_{X_j} \cdot W^j_\bot$. Hence, $\hat{u} \notin \sigma'(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}}$, implying

$$\sigma'(X_k)|_\bot \cdot \mathsf{N}^*_{\text{role}} \supset \sigma'(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}}.$$

Together with $\sigma \sqsubset \sigma'$ from (1), this implies $\sigma'(X_k) \sqsubset \sigma'(E_k)$. It remains to show $\sigma' \sqsubseteq \tau$. As (i) the definition of $\sigma'(X_j)|_\bot$ for $\bot$-modifications in $\mathcal{FL}_\neg$ is identical to the one for $\mathcal{FL}_\bot$, and (ii) the characterization of subsumption for $\mathcal{FL}_\neg$ is analogous to the one for $\mathcal{FL}_\bot$, we

analogously obtain for every $j \in \{1, \ldots, k-1\}$ that either $\sigma'(X_j) \equiv \sigma(X_j)$, trivially implying $\sigma'(X_j) \sqsubseteq \tau(X_j)$, or both $\sigma'(X_j)|_\perp \succ \tau(X_j)|_\perp$ and, for every $P \in \mathsf{P} \setminus \{\perp\}$,

$$\tau(X_j)|_P \subseteq \sigma(X_j)|_P \cup \sigma(X_j)|_\perp \cdot \mathsf{N}^*_{\text{role}}. \qquad (*)$$

Although again

$$\sigma'(X_j)|_\perp \cdot \mathsf{N}^*_{\text{role}} = (\sigma(X_j)|_\perp \cdot \mathsf{N}^*_{\text{role}}) \setminus (\sigma(X_j)|_\perp \cap W^j_\perp),$$

the set $\sigma'(X_j)|_P$ in comparison to the case for $\mathcal{FL}_\perp$ is smaller. Therefore, however, exchanging σ by σ' still preserves the subset relation from $(*)$, yielding

$$\tau(X_j)|_P \subseteq \sigma'(X_j)|_P \cup \sigma'(X_j)|_\perp \cdot \mathsf{N}^*_{\text{role}},$$

for every $P \in \mathsf{P} \setminus \{\perp\}$, yielding $\sigma'(X_j) \sqsubseteq \tau(X_j)$.
Case 2: analogous to the case of $\mathcal{FL}_\perp$. □

Lemma 4.3.12 only depends on the facts (i) that the variables in $\{X_k, \ldots, X_\ell\}$ remain unchanged in the modification of $\sigma_t(E_k)$, (ii) that $\mathsf{match}^{\sqsubseteq}_{\mathcal{FL}_\perp}$ computes least matchers w.r.t. the ordering $\sqsubseteq$ on substitutions, and (iii) that modifications are successful for a solvable matching problem. These facts also hold for $\mathcal{FL}_\neg$, as we have already seen. Consequently, the proof of soundness of the algorithm $\mathsf{match}^{\sqsubseteq}_{\mathcal{FL}_\neg}$ is analogous to Lemma 4.3.12.

The same analogy holds for the proof of completeness. Lemma 4.3.13 only relies on the minimality of modifications, i.e., Lemma 4.3.11, on Part (2) of Lemma 4.3.12 and on the fact that $\mathsf{match}^{\sqsubseteq}_{\mathcal{FL}_\perp}$ computes least solutions. Therefore, completeness for $\mathsf{match}^{\sqsubseteq}_{\mathcal{FL}_\neg}$ can be shown analogously.

It remains to show soundness and completeness for $\mathcal{L} = \mathcal{ALN}$.

Soundness and completeness in $\mathcal{ALN}$

Similar to the previous cases, we show some basic properties of modifications in $\mathcal{ALN}$ which suffice to prove soundness and completeness in the way seen for the smaller DLs.

Lemma 4.3.15 *(Strictness and minimality of modifications in $\mathcal{ALN}$)*
Let $\mathcal{P} := (M, S)$ be an $\mathcal{ALN}$-matching problem under subsumption conditions over N_{con}, N_{role}, and $\mathsf{N}_{\text{var}} =: \{X_1, \ldots, X_\ell\}$. Let $S = \{X_j \ \rho^?_j \ E_j \mid 1 \le j \le \ell\}$. Let σ be a reduced substitution solving $(M, S_\sqsubseteq)$. Let $k \in \{2, \ldots, \ell\}$ with $\rho_k = \sqsubset$ and $\sigma(X_k) \equiv \sigma(E_k)$. Let τ be a reduced solution to M with $\sigma \sqsubset \tau$. Then there exists a modified substitution σ' solving $(M, S_\sqsubseteq)$ with

1. *$\sigma \sqsubset \sigma'$;*
2. *$\sigma' \sqsubseteq \tau$ and $\sigma'(X_k) \sqsubset \sigma'(E_k)$.*

Proof. 1. Case 1: if σ and τ disagree on the $\perp$-concept in the sense introduced in Lemma 4.3.11 then define σ' by a $\perp$-modification choosing some word $\hat{u} \in \sigma(X_k)|_\perp \setminus \tau(E_k)|_\perp$ and $\hat{W}^j_P := \sigma(X_j) \cap W^j_\perp \cap \tau(X_j)|_P$ for every $j \in \{1, \ldots, k-1\}$ and every $P \in \mathsf{P} \setminus \{\perp\}$. For every j, this implies either $\sigma'(X_j) \equiv \sigma(X_j)$ or both $\sigma(X_j)|_\perp \succ \sigma'(X_j)|_\perp$ and

$$\sigma'(X_j)|_P \subseteq \sigma(X_j)|_P \cup \sigma(X_j)|_\perp \cdot \mathsf{N}^*_{\text{role}}$$

for every $P \in \mathsf{P} \setminus \{\perp\}$. As $\hat{u} \in \sigma(E_k)|_\perp$, the second case applies to at least one index j, implying $\sigma \sqsubset \sigma'$ by Definition 4.2.9.

Case 2: if σ and τ agree on the $\bot$-concept then define σ' by am N_{con}-modification choosing some $P \in \mathsf{P} \setminus \{\bot\}$ and some word $\hat{u} \in \sigma(X_k)|_P$. Hence, analogous to Lemma 4.3.11, $\sigma'(X_j)|_Q = \sigma(X_j)|_Q$ for all $Q \in \mathsf{P} \setminus \{P\}$ and all $j \in \{1, \dots, \ell\}$ and also for $Q = P$ and all $j \in \{k, \dots, \ell\}$. For all $j \in \{1, \dots, k-1\}$, $\sigma'(X_j)|_P \subseteq \sigma(X_j)|_P$ with a strict inclusion for at least one index j. Hence, by Definition 4.2.9, $\sigma \sqsubset \sigma'$.

2. We show that the guessing strategy from (1) suffices for our claim. Case 1: in case of a $\bot$-modification, $\hat{u} \in \sigma(E_k)|_\bot \setminus \tau(E_k)|_\bot$ and $\hat{u} \notin E_k|_\bot$, implying $\hat{u} \in E_k|_{X_k} \cdot W^j_\bot$ for at least one j, implying

$$\sigma'(X_k)|_\bot \cdot \mathsf{N}^*_{\text{role}} \supset \sigma'(E_k)|_\bot \cdot \mathsf{N}^*_{\text{role}}.$$

Again, as $\sigma'(X_k) = \sigma(X_k)$ and $\sigma \sqsubset \sigma'$, implying $\sigma'(X_k) \sqsubseteq \sigma'(E_k)$, this suffices to prove the strict subsumption $\sigma'(X_k) \sqsubset \sigma'(E_k)$.

The proof for $\sigma' \sqsubseteq \tau$ is analogous to Lemma 4.3.11. By construction of σ', for every $j \in \{1, \dots, \ell\}$ either $\sigma'(X_j) = \sigma(X_j)$ or $\sigma(X_j)|_\bot \succ \sigma'(X_j)|_\bot$. As $\sigma \sqsubset \tau$ and as σ and τ disagree on the $\bot$-concept, also $\sigma(X_j)|_\bot \succ \tau(X_j)|_\bot$, implying $\sigma'(X_j)|_\bot \succ \tau(X_j)|_\bot$. With $\sigma(X_j) \sqsubset \tau(X_j)$ we also know by Lemma 4.2.9 that

$$\tau(X_j)|_P \subseteq \sigma(X_j)|_P \cup \sigma(X_j)|_\bot \cdot \mathsf{N}^*_{\text{role}}$$

for every $P \in \mathsf{P} \setminus (\{\bot\} \cup \mathsf{N}_{\leqslant})$ and

$$\tau(X_j)|_P \subseteq \sigma(X_j)|_P \cup \sigma(X_j)|_\bot \cdot \mathsf{N}^*_{\text{role}} \cup \sigma(X_j)|_\bot \cdot r^{-1}$$

for every $P =: (\leqslant n\ r) \in \mathsf{N}_{\leqslant}$. In both cases, every word in $(\sigma(X_j)|_\bot \cdot \mathsf{N}^*_{\text{role}}) \setminus (\sigma'(X_j)|_\bot \cdot \mathsf{N}^*_{\text{role}}$ is either in $\sigma'(X_j)|_P$ or, by choice of W^j_P, not contained in $\tau(X_j)|_P$. Hence, the above inclusions hold when replacing σ by σ', implying by Definition 4.2.9 that $\sigma'(X_j) \sqsubseteq \tau(X_j)$ for every $j \in \{1, \dots, k-1\}$. Hence, $\sigma' \sqsubseteq \tau$.

Case 2: analogous to the case of $\mathcal{FL}_\bot$. □

As seen in Lemmas 4.3.12 and 4.3.13 for the case of $\mathcal{FL}_\bot$, soundness and completeness can be shown using only the facts that (i) $\mathsf{match}^{\sqsubseteq}_{\mathcal{L}}$ computes least matchers, (ii) the variables $\{X_k, \dots, X_\ell\}$ remain unchanged in a modification step for the kth subsumption condition, and (iii) modifications are successful for solvable matching problems. Consequently, soundness and completeness can be shown analogously for $\mathcal{ALN}$.

The successful computation paths of $\mathsf{match}^{\sqsubseteq}_{\mathcal{L}}(\mathcal{P})$ yield all minimal matchers of $\mathcal{P}$ while the length of each computation path is polynomially bounded. Because matching under acyclic side conditions in $\mathcal{FL}_\bot$, $\mathcal{FL}_\neg$, $\mathcal{ALN}$ is known to be NP-hard [BKBM99], we obtain the following theorem.

Theorem 4.3.16 *For every $\mathcal{L} \in \{\mathcal{FL}_\bot, \mathcal{FL}_\neg, \mathcal{ALN}\}$, deciding the solvability of $\mathcal{L}$-matching problems under acyclic side conditions is an NP-complete problem. The least solution to an $\mathcal{L}$-matching problem under acyclic side conditions can be computed by a non-deterministic polynomial time algorithm.*

4.4 Matching in $\mathcal{EL}$ w.r.t. hybrid TBoxes

In the present section we show how to solve matching problems w.r.t. a background terminology, in our case, cyclic or hybrid $\mathcal{EL}$-TBoxes. As shown in Section 3.3, hybrid $\mathcal{EL}$-TBoxes can in polynomial time be reduced to ordinary (cyclic) $\mathcal{EL}$-TBoxes interpreted with gfp-semantics. Hence, for matching in $\mathcal{EL}$ w.r.t. hybrid TBoxes it suffices to devise a matching algorithm w.r.t. cyclic $\mathcal{EL}$-TBoxes.

4.4.1 Matching in $\mathcal{EL}$ w.r.t. cyclic TBoxes

For the definition of concept matching problems at the beginning of this chapter, we have extended concept descriptions by variables, yielding concept patterns. As our aim in the present section is to define matching problems w.r.t. cyclic $\mathcal{EL}$-TBoxes, our first step is to extend these by variables likewise. To this end, the following definition introduces pattern TBoxes in which the right-hand side of a definition may be a concept pattern.

Definition 4.4.1 (Pattern TBox)
An *$\mathcal{EL}$-pattern TBox* $\mathcal{T}$ is a finite set of definitions of the form $A \equiv C$, where $A \in \mathsf{N}_{\text{def}}$ and C is a concept pattern over N_{prim}, N_{def}, N_{role}, and N_{var}. A is called *defined* in $\mathcal{T}$ and may occur on the right hand-side of no other definition in $\mathcal{T}$. Denote by $\mathsf{N}_{\text{var}}^{\mathcal{T}}$ the set of all variables occurring in $\mathcal{T}$. □

Note that variables do not occur on left-hand sides of definitions. Denote by $\mathsf{N}_{\text{var}}^{\mathcal{T}}(A)$ the set of variables in $\mathcal{T}$ 'reachable' from A.

Definition 4.4.2 (Matching problem)
Let $\mathcal{T}$ be an $\mathcal{EL}$-pattern TBox with $A, B \in \mathsf{N}_{\text{def}}^{\mathcal{T}}$. Moreover, let $\mathsf{N}_{\text{var}}^{\mathcal{T}}(A) = \emptyset$. Then $A \equiv^{?}_{\text{gfp},\mathcal{T}} B$ is an *$\mathcal{EL}$-matching problem modulo equivalence w.r.t. $\mathcal{T}$ with gfp-semantics.* □

Throughout this section, we shall refer to '$\mathcal{EL}$-matching problem modulo equivalence with gfp-semantics' by '$\mathcal{EL}$-matching problem'. In order to define solutions to matching problems appropriately, some preparation is necessary. The following definition introduces conservative extensions for pattern TBoxes.

Definition 4.4.3 (Conservative extension)
Let $\mathcal{T}_1$ be an $\mathcal{EL}$-pattern TBox over N_{prim}, N_{def}, N_{role}. and N_{var}. Then an $\mathcal{EL}$-pattern TBox $\mathcal{T}_2$ is a *conservative extension* of $\mathcal{T}_1$ iff $\mathsf{N}_{\text{prim}}^{\mathcal{T}_2} = \mathsf{N}_{\text{prim}}^{\mathcal{T}_1}$, $\mathsf{N}_{\text{role}}^{\mathcal{T}_2} = \mathsf{N}_{\text{role}}^{\mathcal{T}_1}$, $\mathsf{N}_{\text{var}}^{\mathcal{T}_1} \supseteq \mathsf{N}_{\text{var}}^{\mathcal{T}_2}$, and $\mathcal{T}_1 \subseteq \mathcal{T}_2$. □

Note that the above definition coincides on ordinary TBoxes with the definition of conservative extensions from [Baa03a]. Moreover, since $\mathcal{T}_2$ is a pattern TBox, $\mathsf{N}_{\text{def}}^{\mathcal{T}_1} \cap \mathsf{N}_{\text{def}}^{\mathcal{T}_2 \setminus \mathcal{T}_1} = \emptyset$. In contrast to concept matching, we do not use substitutions to instantiate variables. Instead, we simply extend our TBoxes by definitions for the occurring variables. This is accomplished by means of so-called *instantiations*.

Definition 4.4.4 (Instantiation)
Let $\mathcal{T}_1$ be an $\mathcal{EL}$-pattern TBox over N_{prim}, N_{def}, N_{role}, and N_{var}. Let $\mathcal{T}_2$ be a conservative extension of $\mathcal{T}_1$. For every $X \in \mathsf{N}_{\text{var}}^{\mathcal{T}_1}$, let D_X be a concept pattern over N_{prim}, N_{def}, N_{role}, and $\mathsf{N}_{\text{var}}^{\mathcal{T}_1}$. Then

$$\mathcal{T}_3 := \mathcal{T}_2 \cup \{X \equiv D_X \mid X \in \mathsf{N}_{\text{var}}^{\mathcal{T}_1}\}$$

is an *instantiation* of $\mathcal{T}_1$. □

Intuitively, an instantiation turns variables into defined concepts, and thus turns a pattern TBox into an ordinary TBox. Using these notions, it is particularly simple to define solutions to matching problems.

Definition 4.4.5 (Matcher)
Let $A \equiv^{?}_{\text{gfp},\mathcal{T}} B$ be an $\mathcal{EL}$-matching problem and let $\mathcal{T}'$ be an instantiation of $\mathcal{T}$. Then $\mathcal{T}'$ is a *matcher of* $A \equiv^{?}_{\text{gfp},\mathcal{T}} B$ iff $A \equiv_{\text{gfp},\mathcal{T}'} B$. □

Hence, a matcher to $A \equiv^{?}_{\mathrm{gfp},\mathcal{T}} B$ extends the pattern TBox $\mathcal{T}$ by definitions for all variables reachable from B such that A and B become equivalent. Clearly, we may restrict ourselves to matching problems w.r.t. names because it holds for every concept description C and every concept pattern D defined over a pattern TBox $\mathcal{T}$ that the matching problem $C \equiv^{?}_{\mathrm{gfp},\mathcal{T}} D$ can be simulated by $A \equiv^{?}_{\mathrm{gfp},\mathcal{T}\cup\{A\equiv C, B\equiv D\}} B$ with A, B fresh defined names. Before turning to solving matching problems w.r.t. cyclic $\mathcal{EL}$-TBoxes as defined above, some preparation is necessary.

4.4.2 Formal preliminaries

Recall that subsumption for cyclic $\mathcal{EL}$-TBoxes with gfp-semantics has already been characterized in Section 3.3.1 by means of simulation relations on description graphs. In Section 4.1.2, in the context of concept matching in $\mathcal{ALE}$, a superlanguage of $\mathcal{EL}$, we have seen that the lcs plays an important role in the definition of the relevant matching algorithm. In this sense it is not surprising that our first step towards matching w.r.t. cyclic $\mathcal{EL}$-TBoxes is to introduce the lcs for cyclic $\mathcal{EL}$-TBoxes. The relevant definitions are due to [Baa03a].

Definition 4.4.6 (Gfp-lcs)
Let $\mathcal{T}_1$ be a cyclic $\mathcal{EL}$-TBox and let $A, B \in \mathsf{N}^{\mathcal{T}_1}_{\mathrm{def}}$. Let $\mathcal{T}_2$ be a conservative extension of $\mathcal{T}_1$ with $E \in \mathsf{N}^{\mathcal{T}_2}_{\mathrm{def}} \setminus \mathsf{N}^{\mathcal{T}_1}_{\mathrm{def}}$. Then E is the *least common subsumer of A and B in $\mathcal{T}_1$ w.r.t. gfp-semantics (gfp-lcs)* iff the following conditions hold:

1. $A \sqsubseteq_{\mathrm{gfp},\mathcal{T}_2} E$ and $B \sqsubseteq_{\mathrm{gfp},\mathcal{T}_2} E$;
2. if $\mathcal{T}_3$ is a conservative extension of $\mathcal{T}_2$ and F a defined concept in $\mathcal{T}_3$ such that $A \sqsubseteq_{\mathrm{gfp},\mathcal{T}_3} F$ and $B \sqsubseteq_{\mathrm{gfp},\mathcal{T}_3} F$ then $E \sqsubseteq_{\mathrm{gfp},\mathcal{T}_3} F$ □

In order to be able to actually compute the lcs, the product of description graphs is introduced.

Definition 4.4.7 (Graph product)
Let $\mathcal{G}_i := (V_i, E_i, L_i)$, $i = 1, 2$ be two description graphs. Their *product* is the description graph $\mathcal{G}_1 \times \mathcal{G}_2 := (V, E, L)$ with

- $V := V_1 \times V_2$;
- $E := \{((v_1, v_2), r, (v_1', v_2')) \mid \forall i \in \{1, 2\} \colon (v_i, r, v_i') \in E_i\}$; and
- $L(v_1, v_2) := L_1(v_1) \cap L_2(v_2)$.

For a description graph $G = (V, E, L)$ and $n \in \mathbb{N} \setminus \{0\}$, the n-ary graph product is inductively defined as follows.

$$\begin{aligned} \mathcal{G}^1 &:= \mathcal{G}; \\ \mathcal{G}^{n+1} &:= \mathcal{G}^n \times \mathcal{G} \end{aligned}$$ □

Due to associativity and commutativity of the Cartesian product we may denote the vertices of $\mathcal{G}^n$ by n-ary tuples $(v_1, \ldots, v_n)$ instead of $(\ldots(v_1, v_2), \ldots, v_n)$. In order to transform product graphs back to TBoxes, we define TBoxes induced by description graphs.

Definition 4.4.8 (TBox of $\mathcal{G}$)
Let $\mathcal{G} := (V, E, L)$ be a description graph. Then the *TBox of* $\mathcal{G}$ is defined by

$$\mathsf{tbox}(\mathcal{G}) := \{A \equiv \bigsqcap_{P \in L(A)} P \sqcap \bigsqcap_{(A,r,B) \in E} \exists r.B \mid A \in V\}$$ □

Two of the main results from [Baa03a] prove that the gfp-lcs can in fact be computed in polynomial time by means of the graph product.

Lemma 4.4.9 *Let $\mathcal{T}$ be a cyclic $\mathcal{EL}$-TBox and $A, B \in \mathsf{N}_{\mathrm{def}}^{\mathcal{T}}$. Then (A, B) w.r.t. $\mathcal{T} \cup \mathsf{tbox}(\mathcal{G}_{\mathcal{T}} \times \mathcal{G}_{\mathcal{T}})$ is the gfp-lcs of A and B w.r.t. $\mathcal{T}$.*

Theorem 4.4.10 *Let $\mathcal{T}$ be a cyclic $\mathcal{EL}$-TBox and $A, B \in \mathsf{N}_{\mathrm{def}}^{\mathcal{T}}$. Then the gfp-lcs of A and B w.r.t. $\mathcal{T}$ always exists, and it can be computed in polynomial time.*

With this preparation, we are ready to introduce our matching algorithm for cyclic $\mathcal{EL}$-TBoxes in the following section.

4.4.3 Solving matching problems w.r.t. cyclic $\mathcal{EL}$-TBoxes

By treating variables as primitive concepts, pattern TBoxes can, syntactically, be regarded as ordinary TBoxes. This allows us to define normalized pattern TBoxes analogously to normalized cyclic TBoxes, and to transform pattern TBoxes into description graphs and vice versa.

Definition 4.4.11 (n-ary product)
Let $\mathcal{T}$ be an $\mathcal{EL}$-pattern TBox and let $n \in \mathbb{N} \setminus \{0\}$. The *$n$-ary product* $\mathcal{T}^n$ of $\mathcal{T}$ is defined as $\mathcal{T}^n := \mathsf{tbox}(\mathcal{G}_{\mathcal{T}}^n)$. □

In order to extend the notion of simulation relations to graphs of pattern TBoxes, variables are simply ignored:

Definition 4.4.12 (Simulation relation)
Let $\mathcal{T}$ be a normalized $\mathcal{EL}$-pattern TBox. Then Z is a *simulation relation* on $\mathcal{G}_{\mathcal{T}}$ iff $Z \colon \mathcal{G}_{\mathcal{T}'} \rightleftarrows \mathcal{G}_{\mathcal{T}'}$, where $\mathcal{T}' = \mathcal{T}[X/\top \mid X \in \mathsf{N}_{\mathrm{var}}^{\mathcal{T}}]$. □

Note that in respect to the treatment of variables, the above definition is analogous to Definition 4.1.3 for homomorphisms on description trees. We can now define our matching algorithm w.r.t. cyclic $\mathcal{EL}$-TBoxes as follows.

Definition 4.4.13 (match)
Let $\mathcal{T}$ be a normalized $\mathcal{EL}$-pattern TBox and let $A \equiv^?_{\mathrm{gfp},\mathcal{T}} B$ be an $\mathcal{EL}$-matching problem. For every simulation relation $Z \colon \mathcal{G}_{\mathcal{T}} \rightleftarrows \mathcal{G}_{\mathcal{T}}$ and for every $X \in \mathsf{N}_{\mathrm{var}}^{\mathcal{T}}$, define

$$Z(X) := \{A' \in \mathsf{N}_{\mathrm{def}} \mid \exists B' \in \mathsf{N}_{\mathrm{def}} \colon (B', A') \in Z \wedge X \in L_{\mathcal{T}}(B')\}.$$

Then, $\mathsf{match}(A \equiv^?_{\mathrm{gfp},\mathcal{T}} B)$ is defined as shown in Figure 4.4.1. □

Upon input $A \equiv^?_{\mathrm{gfp},\mathcal{T}} B$, our matching algorithm match returns all instantiations $\mathcal{T}_Z$ for which, firstly, Z is a simulation relation on $\mathcal{G}_{\mathcal{T}}$ with $(B, A) \in Z$; and secondly, A subsumes B w.r.t. $\mathcal{T}_Z$ interpreted with gfp-semantics.

For a given Z, $\mathcal{T}_Z$ is defined as an instantiation of a conservative extension of $\mathcal{T}$. We discuss the conservative extension first and the additional definitions for variables afterwards. For

Input: matching problem $\mathcal{P} := A \equiv^{?}_{\mathrm{gfp},\mathcal{T}} B$ with normalized $\mathcal{EL}$-pattern TBox $\mathcal{T}$
Output: set of matchers of $\mathcal{P}$

Return $\{\mathcal{T}_Z \mid Z\colon \mathcal{G}_\mathcal{T} \rightrightarrows \mathcal{G}_\mathcal{T} \wedge (B,A) \in Z \wedge A \sqsupseteq_{\mathrm{gfp},\mathcal{T}_Z} B\}$,
where, for every $Z\colon \mathcal{G}_\mathcal{T} \rightrightarrows \mathcal{G}_\mathcal{T}$, $\mathcal{T}_Z$ is defined by:

$$\begin{aligned}\mathcal{T}_Z := \mathcal{T} \cup \bigcup_{i \in \{|Z(X)| \mid X \in \mathsf{N}^\mathcal{T}_{\mathrm{var}}(B)\} \setminus \{1\}} & (\mathcal{T}[X/\top \mid X \in \mathsf{N}^\mathcal{T}_{\mathrm{var}}])^i \\ \cup \{X \equiv (A_1,\dots,A_n) \mid{}& X \in \mathsf{N}^\mathcal{T}_{\mathrm{var}}(B) \\ & \wedge Z(X) = \{A_1,\dots,A_n\} \wedge |Z(X)| = n\} \\ \cup \{X \equiv \top \mid{}& X \notin \mathsf{N}^\mathcal{T}_{\mathrm{var}}(B)\}.\end{aligned}$$

Figure 4.4.1: The algorithm match for cyclic $\mathcal{EL}$-TBoxes

every variable $X \in \mathsf{N}^\mathcal{T}_{\mathrm{var}}$, $\mathcal{T}$ is extended by the $|Z(X)|$-ary graph product of $\mathcal{T}$. For every X, the set $Z(X)$ contains all 'destination' vertices onto which vertices in $\mathcal{G}_\mathcal{T}$ labeled by X are mapped. Hence, whenever Z maps vertices labeled by X onto n different vertices then $\mathcal{T}$ is extended by the n-ary graph product of $\mathcal{T}$. More precisely, the graph product is computed after removing variables from $\mathcal{T}$. Note that this removal is only done for convenience to simplify the notation in our proofs and not necessary for correctness or completeness of the algorithm.

As a result, the relevant conservative extension of $\mathcal{T}$ for every X contains a definition of the lcs over all destination vertices of vertices labeled by X: if $Z(X)$ contains n pairwise distinct destination vertices $\{A_1,\dots,A_n\}$ then, by Lemma 4.4.9, the relevant lcs is the vertex $(A_1,\dots,A_n)$ in the n-ary product of $\mathcal{T}$.

As the second line of the definition of $\mathcal{T}_Z$ shows, X is finally assigned the lcs over all destinations of X: $X \equiv (A_1,\dots,A_n)$. Note that the condition $|Z(X)| = n$ only ensures pairwise distinctness of the vertices $A_1,\dots,A_n$. Without this condition, X might be assigned to vertices not existing in the relevant extension. Note also that variables unreachable from B are assigned $\top$.

Some remarks on the analogy between the $\mathcal{ALE}$-concept matching algorithm from Definition 4.1.7 and the above might help to understand our matching algorithm further.

- The notion of $\top$-patterns used in the definition of $\mathsf{match}_{\mathcal{ALE}}$ does not occur here because it is only relevant in the presence of value restrictions.

- While $\mathsf{match}_{\mathcal{ALE}}$ considers homomorphisms between description trees, the above algorithm generalizes this to simulation relations between description graphs. Especially, for an $\mathcal{EL}$-concept description C and an $\mathcal{EL}$-concept pattern D, every homomorphism from $\mathsf{tree}(D)$ onto $\mathsf{tree}(C)$ can be viewed as a simulation relation Z on $\mathcal{G}_{\{A\equiv C, B\equiv D\}}$ with $(B,A) \in Z$, where A, B are fresh concept names. Conversely, every simulation relation Z on $\mathcal{G}_{\{A\equiv C, B\equiv D\}}$ with $(B,A) \in Z$ and fresh names A, B induces a positive finite number of homomorphisms from $\mathsf{tree}(D)$ onto $\mathsf{tree}(C)$.

- For every variable X occurring in the respective matching problems, both algorithms compute the lcs over the concepts induced by all 'destinations' of X w.r.t. the relevant homomorphism or simulation relation, respectively. While $\mathsf{match}_{\mathcal{ALE}}$ uses the lcs explicitly, it is implicit in the graph product used in the above definition.

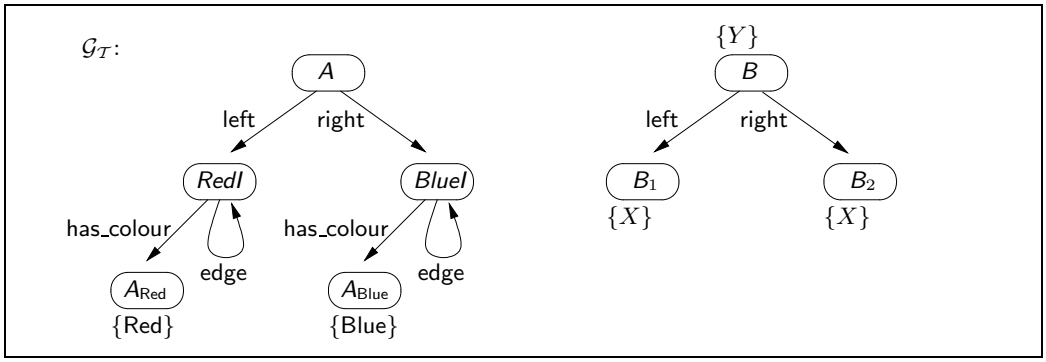


Figure 4.4.2: Example description graph of an $\mathcal{EL}$-pattern TBox

- The algorithm $\mathsf{match}_{\mathcal{ALE}}$ returns a set of substitutions while the above algorithm returns a set of instantiations. However, as we will show in Lemma 4.4.19, the latter generalizes the former in the sense that every $\mathcal{EL}$-matching problem w.r.t. the empty TBox can be solved by means of our algorithm for matching w.r.t. cyclic $\mathcal{EL}$-TBoxes.

In order to get an impression how the above matching algorithm works, consider the following simple example.

Example 4.4.14 Recall Example 2.2.1, where nodes lying on an infinite chain have been defined. We extend this example by introducing infinite chains of coloured nodes. Let $\mathsf{N}_{\text{prim}} := \{\mathsf{Red}, \mathsf{Blue}\}$ and $\mathsf{N}_{\text{role}} := \{\mathsf{edge}, \mathsf{colour}, \mathsf{left}, \mathsf{right}\}$. In our example pattern TBox $\mathcal{T}$, we define the concept RedI of infinite chains of red nodes and the concept BlueI of infinite chains of blue nodes as follows.

$$\begin{aligned} \mathit{RedI} &\equiv \exists \mathsf{edge}.\mathit{RedI} \sqcap \exists \mathsf{has\_colour}.\mathsf{Red} \\ \mathit{BlueI} &\equiv \exists \mathsf{edge}.\mathit{BlueI} \sqcap \exists \mathsf{has\_colour}.\mathsf{Blue} \\ A &\equiv \exists \mathsf{left}.\mathit{RedI} \sqcap \exists \mathsf{right}.\mathit{BlueI} \\ B &\equiv Y \sqcap \exists \mathsf{left}.X \sqcap \exists \mathsf{right}.X \end{aligned}$$

Moreover, $\mathcal{T}$ contains a defined concept A with an existential restriction over left to an infinite red path and an existential restriction over right to an infinite blue path. For the sake of our example we have added a concept pattern B with a variable Y on the top-level, and existential restrictions over right and left, both with another variable X. Figure 4.4.2 shows the description graph of $\mathcal{T}$ after normalization. For every vertex with a non-empty label set, the corresponding label set is denoted above or underneath the vertex. Our goal is to solve the matching problem $A \equiv^{?}_{\text{gfp},\mathcal{T}} B$.

According to the definition of match, we have to consider every simulation relation Z on $\mathcal{G}_\mathcal{T}$ with $(B, A) \in Z$. In our case, it is easy see that only two such Z exist, one of which being

$$Z = \{(B, A), (B_1, \mathit{RedI}), (B_2, \mathit{BlueI})\}.$$

The only other simulation relation containing (B, A) swaps the destination vertices of B_1 and B_2, which need not be considered separately. Hence, the next step is to compute $\mathcal{T}_Z$ for the above Z. Clearly, $\mathsf{N}^{\mathcal{T}}_{\text{var}}(B) = \{X, Y\}$ and, by definition, $Z(X) = \{\mathit{RedI}, \mathit{BlueI}\}$ and $Z(Y) = \{A\}$, so that obviously $|Z(X)| = 2$ and $|Z(Y)| = 1$. Thus, we can already write down our *candidate* solution $\mathcal{T}_Z$ as

$$\mathcal{T}_Z := \mathcal{T} \cup (\mathcal{T}[X/\top, Y/\top])^2 \cup \{X \equiv (\mathit{RedI}, \mathit{BlueI}), Y \equiv A\}.$$

The definitions of X and Y in the instantiation show that the product TBox is only interesting insofar as it defines $(\mathit{RedI}, \mathit{BlueI})$, implying that we need not compute the entire product TBox for our example but only the subgraph induced by this one interesting vertex. To simplify our notation, let $\mathcal{T}' := \mathcal{T}[X/\top, Y/\top]$. It is easy to check that in $\mathcal{G}^2_{\mathcal{T}'}$, the vertex $(\mathit{RedI}, \mathit{BlueI})$ is connected via an edge labeled edge to itself and has an $\mathsf{has\_colour}$-edge to the vertex $(A_{\mathsf{Red}}, A_{\mathsf{Blue}})$. The latter has no outgoing edges and an empty label set because the vertices A_{Red} and A_{Blue} have no outgoing edges and disjoint label sets in $\mathcal{G}_{\mathcal{T}}$. Hence, the only definitions of $\mathcal{T}'^2$ relevant for $\mathcal{T}_Z$ are

$$\begin{aligned}(\mathit{RedI}, \mathit{BlueI}) &\equiv \exists \mathsf{edge}.(\mathit{RedI}, \mathit{BlueI}) \sqcap \exists \mathsf{has\_colour}.(A_{\mathsf{Red}}, A_{\mathsf{Blue}})\\ (A_{\mathsf{Red}}, A_{\mathsf{Blue}}) &\equiv \top,\end{aligned}$$

so that we can write $\mathcal{T}_Z$ as the extension of $\mathcal{T}$ by the following four definitions.

$$\begin{aligned}(\mathit{RedI}, \mathit{BlueI}) &\equiv \exists \mathsf{edge}.(\mathit{RedI}, \mathit{BlueI}) \sqcap \exists \mathsf{has\_colour}.(A_{\mathsf{Red}}, A_{\mathsf{Blue}})\\ (A_{\mathsf{Red}}, A_{\mathsf{Blue}}) &\equiv \top\\ X &\equiv (\mathit{RedI}, \mathit{BlueI})\\ Y &\equiv A\end{aligned}$$

Observe that the assignment to X, i.e., the lcs of RedI and BlueI, cannot be expressed without introducing a new definition for nodes of *some* colour on infinite paths. It remains to test whether $\mathcal{T}_Z$ actually solves the matching problem.

By definition of match, $\mathcal{T}_Z$ is a solution to the matching problem iff $A \sqsupseteq_{\mathrm{gfp},\mathcal{T}_Z} B$. In order to check this subsumption, we have to normalize $\mathcal{T}_Z$ in the sense of Definition 3.3.3. The only definitions violating the normal form are those of B, X, and Y because defined concepts occur outside of existential quantifiers—note that after instantiation of Y, Y is a defined concept. Hence, we replace all defined concepts by their corresponding definitions, obtaining

$$\begin{aligned}B &\equiv \exists \mathsf{left}.\mathit{RedI} \sqcap \exists \mathsf{right}.\mathit{BlueI} \sqcap \exists \mathsf{left}.X \sqcap \exists \mathsf{right}.X\\ X &\equiv \exists \mathsf{edge}.(\mathit{RedI}, \mathit{BlueI}) \sqcap \exists \mathsf{has\_colour}.(A_{\mathsf{Red}}, A_{\mathsf{Blue}})\\ Y &\equiv \exists \mathsf{left}.\mathit{RedI} \sqcap \exists \mathsf{right}.\mathit{BlueI},\end{aligned}$$

while the rest of $\mathcal{T}_Z$ remains unchanged. As B syntactically contains the definition of A, it is easy to check that there exists a simulation relation W on description graph of the normalized version of $\mathcal{T}_Z$ with $(A, B) \in W$. Hence, one solution returned by $\mathsf{match}(A \equiv^?_{\mathrm{gfp},\mathcal{T}} B)$ is $\mathcal{T}_Z$. Note that our algorithm computes $\mathcal{T}_Z$ twice because of the above mentioned other simulation relation swapping the destinations of B_1 and B_2. However, as both results are identical and as the solution of $\mathsf{match}(A \equiv^?_{\mathrm{gfp},\mathcal{T}} B)$ is defined as a set, this set contains only $\mathcal{T}_Z$.

To be quite exact, the our algorithm computes a syntactically larger solution because the entire TBox $\mathcal{T}'^2$ is part of the actual $\mathcal{T}_Z$. For the sake of simplicity, we have restricted our attention to the relevant part of $\mathcal{T}'^2$. □

It remains to show that the above algorithm in general correctly solves $\mathcal{EL}$-matching problems w.r.t. cyclic $\mathcal{EL}$-TBoxes.

4.4.4 Soundness and completeness

The following lemma proves soundness of our algorithm match, i.e., every solution returned by the algorithm solves the input matching problem.

Lemma 4.4.15 *(Soundness)*
Let $A \equiv^?_{\mathrm{gfp},\mathcal{T}} B$ be an $\mathcal{EL}$-matching problem with normalized $\mathcal{T}$. Let $\mathcal{T}_Z$ be one instantiation in $\mathsf{match}(A \equiv^?_{\mathrm{gfp},\mathcal{T}} B)$. Then $A \equiv_{\mathrm{gfp},\mathcal{T}_Z} B$.

Proof. By definition of match, $A \sqsupseteq_{\mathrm{gfp},\mathcal{T}_Z} B$. To show the reverse direction, it suffices to construct a simulation relation $Y \colon \mathcal{G}_{\mathcal{T}_Z} \rightleftharpoons \mathcal{G}_{\mathcal{T}_Z}$ with $(B, A) \in Y$.

As description graphs are defined only for normalized TBoxes, we begin by normalizing $\mathcal{T}_Z$. As $\mathcal{T}_Z$ 'instantiates' every variable X in $\mathcal{T}$ by a definition for X, every definition from $\mathcal{T}$ containing X on the right-hand side is not normalized any more. Moreover, every definition of the form $X \equiv (A_1, \ldots, A_n)$ is not normalized. Hence, it suffices to modify the right-hand sides of $\mathcal{T}_Z$ as follows. Firstly, in every definition from $\mathcal{T}$ replace every occurrence of a variable X by the definition of X, and secondly, in every definition of the form $X \equiv (A_1, \ldots, A_n)$, replace $(A_1, \ldots, A_n)$ by its definition. Formally, we can define the normalized version $\mathcal{T}'_Z$ of $\mathcal{T}_Z$ by

$$\begin{aligned}\mathcal{T}'_Z := {} & \mathcal{T}[X/\mathsf{deft}_{\mathcal{T}_Z}(X) \mid X \in \mathsf{N}^{\mathcal{T}}_{\mathrm{var}}] \\ & \cup \bigcup_{i \in \{|Z(X)| \mid X \in \mathsf{N}^{\mathcal{T}}_{\mathrm{var}}(B)\} \setminus \{1\}} (\mathcal{T}[X/\top \mid X \in \mathsf{N}^{\mathcal{T}}_{\mathrm{var}}])^i \\ & \cup \{X \equiv \mathsf{deft}_{\mathcal{T}_Z}((A_1, \ldots, A_n)) \mid X \equiv (A_1, \ldots, A_n) \in \mathcal{T}_Z \\ & \qquad \wedge\, \mathsf{deft}_{\mathcal{T}_Z}((A_1, \ldots, A_n)) \neq \top\} \\ & \cup \{X \equiv \top \mid X \equiv \top \in \mathcal{T}_Z\}.\end{aligned}$$

Define a binary relation Y on $\mathcal{G}_{\mathcal{T}'_Z}$ as follows.

$$Y := Z \cup \{((A_1, \ldots, A_n), A_i) \mid A_1, \ldots, A_n \in \mathsf{N}^{\mathcal{T}}_{\mathrm{def}} \wedge \mathcal{T}^n \subseteq \mathcal{T}_Z \wedge 1 \leq i \leq n\}$$

We show that Y is a simulation relation on $\mathcal{G}_{\mathcal{T}'_Z}$, i.e., that Conditions S1 and S2 in Definition 3.3.7 are satisfied.

(S1) Consider an arbitrary pair $(D, C) \in Y$. We show $L_{\mathcal{T}'_Z}(D) \subseteq L_{\mathcal{T}'_Z}(C)$.

- If $(D, C) \in Z$ then $L_{\mathcal{T}_Z}(D) \setminus \mathsf{N}^{\mathcal{T}}_{\mathrm{var}}(D) \subseteq L_{\mathcal{T}_Z}(C) \subseteq L_{\mathcal{T}'_Z}(C)$. If $L_{\mathcal{T}_Z}(D)$ contains only variables X with $X \equiv \top \in \mathcal{T}_Z$ then $L_{\mathcal{T}'_Z}(D) = L_{\mathcal{T}_Z}(D)$, implying $L_{\mathcal{T}'_Z}(D) \subseteq L_{\mathcal{T}'_Z}(C)$. Otherwise, $X \equiv (A_1, \ldots, A_n) \in \mathcal{T}_Z$ for some variable $X \in L_{\mathcal{T}_Z}(D)$ and appropriate names $A_1, \ldots, A_n$. In this case, $L_{\mathcal{T}'_Z}(D)$ instead of X additionally contains the primitive concepts in $L_{\mathcal{T}_Z}((A_1, \ldots, A_n))$. But $(D, C) \in Z$ by Definition 4.4.13 implies $C \in \{A_1, \ldots, A_n\}$, implying $L_{\mathcal{T}_Z}((A_1, \ldots, A_n)) \subseteq L_{\mathcal{T}_Z}(C)$ by definition of the product graph. Hence, the primitive names in $L_{\mathcal{T}'_Z}(D)$ replacing X are contained in $L_{\mathcal{T}_Z}(C) \subseteq L_{\mathcal{T}'_Z}(C)$ as required.
- If $(D, C) \in Y \setminus Z$ then either $(D, C) = ((A_1, \ldots, A_n), A_i)$ for some $1 \leq i \leq n$ or $(D, C) = (X, (A_1, \ldots, A_n))$ with $X \equiv (A_1, \ldots, A_n) \in \mathcal{T}_Z$. In the first case, $L_{\mathcal{T}'_Z}((A_1, \ldots, A_n)) = L_{\mathcal{T}_Z}((A_1, \ldots, A_n))$ by definition of $\mathcal{T}'_Z$, $L_{\mathcal{T}_Z}((A_1, \ldots, A_n)) \subseteq L_{\mathcal{T}_Z}(A_i)$ by definition of the product graph, and $L_{\mathcal{T}_Z}(A_i) \subseteq L_{\mathcal{T}'_Z}(A_i)$. In the second case, we immediately have $\mathsf{def}_{\mathcal{T}'_Z}(X) = \mathsf{def}_{\mathcal{T}'_Z}((A_1, \ldots, A_n))$, implying the proposition.

(S2) Consider an arbitrary edge $(D, r, D') \in E_{\mathcal{T}'_Z}$ with $(D, C) \in Y$. It suffices to find an edge $(C, r, C') \in E_{\mathcal{T}'_Z}$ with $(D', C') \in Y$.

- If $(D, C) \in Z$ and $(D, r, D') \in E_{\mathcal{T}}$ then $Z \colon \mathcal{G}_{\mathcal{T}} \rightleftharpoons \mathcal{G}_{\mathcal{T}}$ guarantees that there exists a defined concept $C' \in \mathsf{N}^{\mathcal{T}}_{\mathrm{def}}$ with the desired properties. Since $\mathsf{N}^{\mathcal{T}}_{\mathrm{def}} \subseteq \mathsf{N}^{\mathcal{T}_Z}_{\mathrm{def}}$, C' also

occurs in $\mathsf{N}_{\text{def}}^{\mathcal{T}_Z}$. Moreover, as shown above, $(D', C') \in Z$ implies $(D', C') \in Y$, and $(C, r, C') \in E_{\mathcal{T}}$ implies $(C, r, C') \in E_{\mathcal{T}_Z'}$ because $\mathcal{G}_{\mathcal{T}_Z}$ preserves all edges from $\mathcal{G}_{\mathcal{T}}$.

- If $(D, C) \in Z$ and $(D, r, D') \notin E_{\mathcal{T}_Z}$ then (D, r, D') corresponds to an existential restriction added to $\mathsf{def}_{\mathcal{T}_Z'}(D)$ by substituting some variable X with $\mathsf{def}_{\mathcal{T}_Z}(X)$ in the definition of D in $\mathcal{T}_Z'$. But then, by definition of $\mathcal{T}_Z$, there are $A_1, \ldots, A_n \in \mathsf{N}_{\text{def}}^{\mathcal{T}}$ and some $1 \leq i \leq n$ such that $X \equiv (A_1, \ldots, A_n) \in \mathcal{T}_Z$ and $C = A_i$. By definition of the product graph, firstly, there is an r-edge $(C, r, C') \in E_{\mathcal{T}}$, and secondly, D' is of the form $(B_1, \ldots, B_n)$ with $C' = B_i$. Hence, $(D', C') \in Y$ by definition of Y.
- If $(D, C) \in Z$ and (D, C) is of the form $((A_1, \ldots, A_n), A_i)$ then (D, r, D') is of the form $((A_1, \ldots, A_n), r, (B_1, \ldots, B_n))$. By definition of the product graph, $(A_i, r, B_i) \in E_{\mathcal{T}}$. By definition of Y, $((B_1, \ldots, B_n), B_i) \in Y$ as required. □

Before proving completeness of match, we first introduce an appropriate notion of completeness for matching problems w.r.t. cyclic $\mathcal{EL}$-TBoxes. For concept matching problems, Definition 4.0.5 introduced the notion of s-completeness for this purpose. Following the same idea, i.e., that more specific matchers are more 'interesting' because they contain more information about the input matching problem, we introduce a subsumption relation for matchers

Definition 4.4.16 Let $\mathcal{P} := A \equiv^?_{\text{gfp},\mathcal{T}} B$ be an $\mathcal{EL}$-matching problem and let $\mathcal{T}_1$, $\mathcal{T}_2$ be instantiations of $\mathcal{T}$. Let $\mathcal{T}_1'$ and $\mathcal{T}_2'$ be renamed instances of $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively, such that $\mathsf{N}_{\text{def}}^{\mathcal{T}_1'} \cap \mathsf{N}_{\text{def}}^{\mathcal{T}_2'} = \emptyset$ and $\mathsf{N}_{\text{var}}^{\mathcal{T}_1'} \cap \mathsf{N}_{\text{var}}^{\mathcal{T}_2'} = \emptyset$. Then

- $\mathcal{T}_1$ *is more specific than* $\mathcal{T}_2$ *(*$\mathcal{T}_1 \sqsubseteq_{\text{s}} \mathcal{T}_2$*)* iff for all $X \in \mathsf{N}_{\text{var}}^{\mathcal{T}}$ it holds that $X^{\mathcal{T}_1'} \sqsubseteq_{\text{gfp},\mathcal{T}_1' \cup \mathcal{T}_2'} X^{\mathcal{T}_2'}$, where $X^{\mathcal{T}_i'}$ is the renamed instance of X in $\mathcal{T}_i'$ for $i = 1, 2$.
- $\mathcal{T}_1$ *is equivalent to* $\mathcal{T}_2$ *(*$\mathcal{T}_1 \equiv_{\text{s}} \mathcal{T}_2$*)* iff $\mathcal{T}_1 \sqsubseteq_{\text{s}} \mathcal{T}_2$ and $\mathcal{T}_2 \sqsubseteq_{\text{s}} \mathcal{T}_1$.

Let $\mathcal{M}$ be a matcher of $\mathcal{P}$. Then $\mathcal{M}$ is called *minimal w.r.t.* $\mathcal{P}$ iff $\mathcal{N} \sqsubseteq_{\text{s}} \mathcal{M}$ implies $\mathcal{N} \equiv_{\text{s}} \mathcal{M}$ for all matchers $\mathcal{N}$ of $\mathcal{P}$. A set of matchers $\mathcal{S}$ to $\mathcal{P}$ is called *s-complete* iff $\mathcal{S}$ contains all minimal matchers w.r.t. $\mathcal{P}$. Moreover, M is *the least matcher* of $\mathcal{P}$ iff $\{\mathcal{M}\}$ is s-complete. □

Note that the least matcher of a matching problem is unique up to s-equivalence. Clearly, but for the different comparison relation, the above definition is analogous to Definition 4.0.5. The following example shows that the least matcher need not always exist even for matching problems w.r.t. acyclic $\mathcal{EL}$-TBoxes.

Example 4.4.17 Let $\mathsf{N}_{\text{prim}} := \{P, Q\}$ and $\mathsf{N}_{\text{role}} := \{r\}$. Moreover, let $\mathsf{N}_{\text{def}} = \{A, B\}$ and $\mathsf{N}_{\text{var}} := \{X, Y\}$. Define a pattern TBox $\mathcal{T}$ by $\mathcal{T} := \{A \equiv \exists r.P \sqcap \exists r.Q, B \equiv \exists r.X \sqcap \exists r.Y\}$. Then, $\mathcal{M}_1 := \mathcal{T} \cup \{X \equiv P, Y \equiv Q\}$ and $\mathcal{M}_2 := \mathcal{T} \cup \{X \equiv Q, Y \equiv P\}$ are both minimal matchers of $A \equiv^?_{\text{gfp},\mathcal{T}} B$. A least matcher does not exist. □

We now prove that the algorithm match is complete in the sense of Definition 4.4.16.

Theorem 4.4.18 *(s-Completeness)*
Let $\mathcal{P} := (A \equiv^?_{\text{gfp},\mathcal{T}} B)$ *be an* $\mathcal{EL}$*-matching problem with normalized* $\mathcal{T}$*. Then,* match$(\mathcal{P})$ *is s-complete.*

Proof. Let $\mathcal{M}$ be a matcher of $\mathcal{P}$. W.l.o.g. let $\mathcal{M}$ be normalized. It suffices to show that $\mathcal{T}_Z \sqsubseteq_{\text{s}} \mathcal{M}$ for some matcher $\mathcal{T}_Z \in \mathsf{match}(\mathcal{P})$. By definition, $\mathcal{M}$ is an instantiation of

$\mathcal{T}$, implying that $\mathcal{M}$ extends a conservative extension $\mathcal{T}'$ of $\mathcal{T}$ by a definition for every variable $X \in \mathsf{N}_{\text{var}}^{\mathcal{T}}$. Moreover, $A \equiv_{\text{gfp},\mathcal{M}} B$, implying a simulation relation $Z_{\mathcal{M}} \colon \mathcal{G}_{\mathcal{M}} \rightrightarrows \mathcal{G}_{\mathcal{M}}$ with $(B, A) \in Z_{\mathcal{M}}$.

Starting from $Z_{\mathcal{M}}$ we iteratively construct a simulation relation Z such that $\mathcal{T}_Z$, as defined in Definition 4.4.13, has the desired properties. Intuitively, Z is a restriction of $Z_{\mathcal{M}}$ to $\mathcal{G}_{\mathcal{T}}$ that has as few elements as possible while still containing (B, A).

- $Z_0 := \{(B, A)\}$
- If $(B_1, r, B_2) \in \mathcal{G}_{\mathcal{T}}$ and $(B_1, A_1) \in Z_i$ and $(B_2, A_2) \notin Z_i$ for all $A_2 \in V_{\mathcal{T}}$ with $(A_1, r, A_2) \in \mathcal{G}_{\mathcal{T}}$ then arbitrarily choose an edge $(A_1, r, A_2) \in \mathcal{G}_{\mathcal{T}}$ such that $(B_2, A_2) \in Z_{\mathcal{M}}$. Then define $Z_{i+1} := Z_i \cup \{(B_2, A_2)\}$; and
- let $Z := Z_n$, where $n \in \mathbb{N}$ is the least index with $Z_n = Z_{n+1}$.

The fact that an edge (A_1, r, A_2) with the desired properties always exists in the above iteration is a consequence of $(B, A) \in Z_{\mathcal{M}}$ and $\mathsf{N}_{\text{var}}^{\mathcal{T}}(A) = \emptyset$. The latter implies that $\mathcal{G}_{\mathcal{T}_{\mathcal{M}}}(A)$ and $\mathcal{G}_{\mathcal{T}}(A)$ are equal. Moreover, $Z \colon \mathcal{G}_{\mathcal{T}} \rightrightarrows \mathcal{G}_{\mathcal{T}}$. S1 holds because all pairs added to Z are elements of $Z_{\mathcal{M}}$ and S2 obviously holds by construction.

We first show $\mathcal{T}_Z \sqsubseteq_{\text{s}} \mathcal{M}$. Assume w.l.o.g. that all defined names $A \in \mathsf{N}_{\text{def}}$ (and variables $X \in \mathsf{N}_{\text{var}}$) are renamed to $A^{\mathcal{T}_Z}$ ($X^{\mathcal{T}_Z}$) in $\mathcal{T}_Z$ and to $A^{\mathcal{M}}$ ($X^{\mathcal{M}}$) in $\mathcal{M}$. It suffices to show $X^{\mathcal{T}_Z} \sqsubseteq_{\text{gfp},\mathcal{T}_Z \cup \mathcal{M}} X^{\mathcal{M}}$ for every $X \in \mathsf{N}_{\text{var}}^{\mathcal{T}}$. By definition of $\mathcal{T}_Z$ it holds that $A_i^{\mathcal{T}_Z} \sqsubseteq_{\text{gfp},\mathcal{T}_Z} (A_1^{\mathcal{T}_Z}, \ldots, A_n^{\mathcal{T}_Z})$ for every $X \equiv (A_1^{\mathcal{T}_Z}, \ldots, A_n^{\mathcal{T}_Z}) \in \mathcal{T}_Z$ and every $1 \leq i \leq n$. As $\mathcal{T}_Z \cup \mathcal{M}$ is a conservative extension of $\mathcal{T}_Z$, this subsumption also holds w.r.t. the union $\mathcal{T}_Z \cup \mathcal{M}$. Moreover, since $Z \subseteq Z_{\mathcal{M}}$, $A_i^{\mathcal{M}} \sqsubseteq_{\text{gfp},\mathcal{M}} M^{\mathcal{M}}$, where $X^{\mathcal{T}_Z} \equiv (A_1^{\mathcal{T}_Z}, \ldots, A_n^{\mathcal{T}_Z}) \in \mathcal{T}_Z$ and $X^{\mathcal{M}} \equiv M^{\mathcal{M}} \in \mathcal{M}$, implying $A_i^{\mathcal{M}} \sqsubseteq_{\text{gfp},\mathcal{T}_Z \cup \mathcal{M}} M^{\mathcal{M}}$. By definition of the lcs, we obtain $(A_1^{\mathcal{T}_Z}, \ldots, A_n^{\mathcal{T}_Z})^{\mathcal{M}} \sqsubseteq_{\text{gfp},\mathcal{T}_Z \cup \mathcal{M}} M^{\mathcal{M}}$, implying $X^{\mathcal{T}_Z} \sqsubseteq_{\text{gfp},\mathcal{T}_Z \cup \mathcal{M}} X^{\mathcal{M}}$.

It remains to show that $\mathcal{T}_Z$ is a matcher. By construction, $A \sqsubseteq_{\text{gfp},\mathcal{T}_Z} B$, so that only the reverse direction must be shown. As $\mathcal{G}_{\mathcal{T}_Z}(A^{\mathcal{T}_Z})$ and $\mathcal{G}_{\mathcal{M}}(A^{\mathcal{M}})$ are equal up to the renaming $\cdot^{\mathcal{T}_Z}/\cdot^{\mathcal{M}}$, clearly (i) $A^{\mathcal{T}_Z} \equiv_{\text{gfp},\mathcal{T}_Z \cup \mathcal{M}} A^{\mathcal{M}}$, and (ii) $A^{\mathcal{M}} \equiv_{\text{gfp},\mathcal{T}_Z \cup \mathcal{M}} B^{\mathcal{M}}$ because $\mathcal{M}$ is a matcher. As shown above, $X^{\mathcal{T}_Z} \sqsubseteq_{\text{gfp},\mathcal{T}_Z \cup \mathcal{M}} X^{\mathcal{M}}$ for every $X \in \mathsf{N}_{\text{var}}^{\mathcal{T}}$. Hence, it is easy to show (iii) $B^{\mathcal{T}_Z} \sqsubseteq_{\text{gfp},\mathcal{T}_Z \cup \mathcal{M}} B^{\mathcal{M}}$ because the difference between $\mathcal{G}_{\mathcal{T}_Z \cup \mathcal{M}}(B^{\mathcal{T}_Z})$ and $\mathcal{G}_{\mathcal{T}_Z \cup \mathcal{M}}(B^{\mathcal{M}})$ corresponds to the different definitions for every $X^{\mathcal{T}_Z}$ and $X^{\mathcal{M}}$. Combining the three subsumptions yields $A^{\mathcal{T}_Z} \sqsupseteq_{\text{gfp},\mathcal{T}_Z \cup \mathcal{M}} B^{\mathcal{T}_Z}$, implying $A \sqsupseteq_{\text{gfp},\mathcal{T}_Z} B$. □

Hence, our matching algorithm for cyclic $\mathcal{EL}$-TBoxes with gfp-semantics is sound and s-complete. As matching w.r.t. cyclic $\mathcal{EL}$-TBoxes generalizes matching in $\mathcal{EL}$ w.r.t. the empty TBox, a natural question is whether match can also be used to solve matching problems w.r.t. the empty TBox. The following lemma shows that this holds and that, in particular, all minimal solutions are found.

Lemma 4.4.19 *Let $C \equiv^? D$ be an $\mathcal{EL}$-matching problem. Let $\mathcal{T} := \{A \equiv C, B \equiv D\}$ with $A, B \notin \mathsf{N}_{\text{prim}}$. Then, σ is a minimal solution to $C \equiv^? D$ iff there exists a minimal matcher $\mathcal{M} \in \mathsf{match}(A \equiv^?_{\text{gfp},\mathcal{T}} B)$ with $X \equiv_{\text{gfp},\mathcal{M}} \sigma(X)$ for every $X \in \mathsf{N}_{\text{var}}(D)$.*

Proof. ($\Rightarrow$) If σ solves $C \equiv^? D$ then $C \equiv \sigma(D)$, implying $A \equiv_{\{A \equiv C, B \equiv \sigma(D)\}} B$ with A, B fresh defined concept names. Hence, $A \equiv_{\mathcal{T}'} B$ with

$$\mathcal{T}' = \{A \equiv C, B \equiv D\} \cup \{X \equiv \sigma(X) \mid X \in \mathsf{N}_{\text{var}}(D)\} \cup \{Y \equiv \top \mid Y \notin \mathsf{N}_{\text{var}}(D)\}.$$

As $\mathcal{T}'$ is acyclic, descriptive and gfp-semantics coincide, yielding $A \equiv_{\text{gfp},\mathcal{T}'} B$, so that $\mathcal{T}'$ is an instantiation of a (trivial) conservative extension of $\mathcal{T}$ solving the matching problem

$A \equiv^?_{\mathrm{gfp},\mathcal{T}} B$. Clearly, as $\mathcal{T}$ is acyclic, $\mathcal{T}'$ is a minimal solution because otherwise σ would be no minimal solution to $C \equiv^? D$. Due to the s-completeness of the algorithm match shown in Theorem 4.4.18, there exists a matcher $\mathcal{M} \in \mathsf{match}(A \equiv^?_{\mathrm{gfp},\mathcal{T}} B)$ with the required properties.

($\Leftarrow$) By definition, $\mathcal{M}$ is an instantiation of a conservative extension $\mathcal{T}_2$ of $\mathcal{T}$ with $A \equiv_{\mathrm{gfp},\mathcal{M}} B$. As $\mathcal{T}$ is acyclic it is easy to see from the definition of match that $\mathcal{T}_2$ and $\mathcal{M}$ are acyclic as well. Hence, $\mathcal{M} \setminus \mathcal{T}_2$, i.e., the part of $\mathcal{M}$ instantiating variables, can be expanded, producing an instantiation $\mathcal{M}'$ in which no defined concept occurs on the right-hand side of any definition of the form $X \equiv C_X$ with $X \in \mathsf{N}_{\mathrm{var}}$. But then

$$\mathcal{T}' := \mathcal{T} \cup \{X \equiv C_X \mid X \in \mathsf{N}_{\mathrm{var}}(D)\}$$

also solves $A \equiv^?_{\mathrm{gfp},\mathcal{T}} B$. As descriptive and gfp-semantics coincide on acyclic TBoxes, $A \equiv_{\mathcal{T}'} B$. Hence, the substitution σ defined by $\sigma(X) = C_X$ for all $X \in \mathsf{N}_{\mathrm{var}}(D)$ solves $C \sqsubseteq^? D$. Again, σ is minimal because otherwise we could construct a solution to $A \equiv^?_{\mathrm{gfp},\mathcal{T}} B$ more specific than $\mathcal{M}$. □

Consequently, our matching algorithm for cyclic $\mathcal{EL}$-TBoxes with greatest-fixedpoint semantics generalizes the $\mathcal{EL}$-matching algorithm w.r.t. the empty TBox presented in [BK00a]. This immediately implies several complexity lower bounds: Firstly, deciding the solvability of matching problems modulo equivalence w.r.t. cyclic $\mathcal{EL}$-TBoxes is NP-hard. Secondly, the minimal matchers to matching problems w.r.t. cyclic $\mathcal{EL}$-TBoxes can be of exponential size in the input TBox. Moreover, the number of minimal matchers can be also be exponential in the input TBox. Any algorithm solving matching problems w.r.t. cyclic $\mathcal{EL}$-TBoxes is therefore necessarily worst-case exponential.

Note also that deciding the solvability of matching problems modulo *subsumption* w.r.t. cyclic $\mathcal{EL}$-TBoxes can be trivially reduced to subsumption w.r.t. cyclic $\mathcal{EL}$-TBoxes by replacing all occurring variables by $\top$.

It remains to examine the computational complexity of our algorithm. For a given matching problem $A \equiv^?_{\mathrm{gfp},\mathcal{T}} B$, the number of simulation relations $Z \colon \mathcal{G}_\mathcal{T} \rightleftharpoons \mathcal{G}_\mathcal{T}$ with $(B, A) \in Z$ can be exponentially large in the input. For every fixed Z, it is easy to see that every instantiation $\mathcal{T}_Z$, as defined in Definition 4.4.13, can be computed in exponential time in the input TBox. Hence, we obtain the following complexity result.

Theorem 4.4.20 *Deciding the solvability of matching problems modulo subsumption w.r.t. cyclic $\mathcal{EL}$-TBoxes with gfp-semantics is tractable. Deciding the solvability of matching problems modulo equivalence w.r.t. cyclic $\mathcal{EL}$-TBoxes with gfp-semantics is NP-hard. The solutions of a matching problem w.r.t. cyclic $\mathcal{EL}$-TBoxes with gfp-semantics can be exponential in number and can be of exponential size in the input matching problem. They can be computed by a deterministic exponential-time algorithm.*

It is open whether the solvability of matching problems modulo equivalence w.r.t. cyclic $\mathcal{EL}$-TBoxes with gfp-semantics is in NP. It might be interesting to note that matching w.r.t. cyclic TBoxes becomes simpler under certain conditions. As shown in [Baa03a], computing the lcs w.r.t. cyclic $\mathcal{EL}$-TBoxes is polynomial for the binary lcs, i.e., if the lcs of only two concepts has to be computed. For matching problems in which every variable occurs at most a constant number of times, this implies that every instantiation $\mathcal{T}_Z$ can be computed in polynomial time.

In preparation for matching w.r.t. hybrid $\mathcal{EL}$-TBoxes, the following subsection shows how to reduce the lcs w.r.t. hybrid $\mathcal{EL}$-TBoxes to cyclic $\mathcal{EL}$-TBoxes with gfp-semantics.

4.4.5 The least-common subsumer w.r.t. hybrid $\mathcal{EL}$-TBoxes

Our aim is to extend the lcs w.r.t. cyclic $\mathcal{EL}$-TBoxes introduced in Definition 4.4.6 to hybrid $\mathcal{EL}$-TBoxes. To this end, the notion of conservative extensions of $\mathcal{EL}$-TBoxes from Definition 4.4.3 has to be extended from cyclic to hybrid TBoxes.

Definition 4.4.21 (Conservative extension)
Let $(\mathcal{F}, \mathcal{T}_1)$ be a hybrid $\mathcal{EL}$-Box. Then $(\mathcal{F}, \mathcal{T}_2)$ is a *conservative extension* of $(\mathcal{F}, \mathcal{T}_1)$ iff $\mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$ in the sense of Definition 4.4.3. □

Hence, a conservative extension of $(\mathcal{F}, \mathcal{T})$ is obtained by fixing $\mathcal{F}$ and extending $\mathcal{T}$ in the usual way. We can now define the lcs w.r.t. hybrid TBoxes analogously to the case of cyclic TBoxes as a conservative extension satisfying certain conditions.

Definition 4.4.22 (Hybrid lcs)
Let $(\mathcal{F}, \mathcal{T}_1)$ be a hybrid TBox and $A, B \in \mathsf{N}_{\mathrm{def}}^{\mathcal{T}_1}$. Let $(\mathcal{F}, \mathcal{T}_2)$ be a conservative extension of $(\mathcal{F}, \mathcal{T}_1)$ with $C \in \mathsf{N}_{\mathrm{def}}^{\mathcal{T}_2}$. Then, C in $(\mathcal{F}, \mathcal{T}_2)$ is the *hybrid least-common subsumer (lcs) of* A, B *in* $(\mathcal{F}, \mathcal{T}_1)$ iff the following conditions hold.

1. $A \sqsubseteq_{\mathrm{gfp},\mathcal{F},\mathcal{T}_2} C$ and $B \sqsubseteq_{\mathrm{gfp},\mathcal{F},\mathcal{T}_2} C$; and
2. If $(\mathcal{F}, \mathcal{T}_3)$ is a conservative extension of $(\mathcal{F}, \mathcal{T}_2)$ and $D \in \mathsf{N}_{\mathrm{def}}^{\mathcal{T}_3}$ such that $A \sqsubseteq_{\mathrm{gfp},\mathcal{F},\mathcal{T}_3} D$ and $B \sqsubseteq_{\mathrm{gfp},\mathcal{F},\mathcal{T}_3} D$ then $C \sqsubseteq_{\mathrm{gfp},\mathcal{F},\mathcal{T}_3} D$. □

In order to compute the lcs w.r.t. hybrid $\mathcal{EL}$-TBoxes, we utilize the reduction from hybrid to cyclic TBoxes defined in Definition 3.3.14 and the usual gfp-lcs algorithm for cyclic $\mathcal{EL}$-TBoxes from Section 4.4.2t. The following lemma shows that this strategy in fact yields the correct result w.r.t. hybrid TBoxes.

Lemma 4.4.23 *Let $(\mathcal{F}, \mathcal{T})$ be a normalized hybrid TBox and $A, B \in \mathsf{N}_{\mathrm{def}}^{\mathcal{T}}$. Then, (A, B) in $(\mathcal{F}, f(\mathcal{T}) \cup f(\mathcal{T})^2)$ is the hybrid lcs of A and B in $(\mathcal{F}, \mathcal{T})$.*

Proof. Let $\mathcal{U} := f(\mathcal{T}) \cup f(\mathcal{T})^2$. We have to show that (A, B) in $(\mathcal{F}, \mathcal{U})$ satisfies the two conditions from Definition 4.4.22.

(1) We show $A \sqsubseteq_{\mathrm{gfp},\mathcal{F},\mathcal{U}} (A, B)$. By Theorem 3.3.18, this holds iff $A \sqsubseteq_{\mathrm{gfp},f(\mathcal{U})} (A, B)$ which holds iff there exists a simulation relation $Y \colon f(\mathcal{U}) \rightleftharpoons f(\mathcal{U})$ with $((A, B), A) \in Y$. We construct such a Y. By definition of the gfp-lcs, we already know $A \sqsubseteq_{\mathrm{gfp},\mathcal{U}} (A, B)$, implying a simulation relation $Y_1 \colon \mathcal{U} \rightleftharpoons \mathcal{U}$ with $((A, B), A) \in Y_1$.

Claim 1. There exists a simulation relation Z such that $Z \colon \mathcal{G}_{f(\mathcal{U})} \rightleftharpoons \mathcal{G}_{\mathcal{U}}$, $Z^{-1} : \mathcal{G}_{\mathcal{U}} \rightleftharpoons \mathcal{G}_{f(\mathcal{U})}$, and the identical relation is a subrelation of Z.

By Lemma 3.3.9, $Y := Z \circ Y_1 \circ Z^{-1}$ is a simulation relation on $f(\mathcal{U})$. Moreover, $((A, B), A) \in Y$ because $((A, B), (A, B)) \in Z$ and $(A, A) \in Z^{-1}$. Hence, (A, B) in $f(\mathcal{U})$ subsumes A, and by symmetry also subsumes B.

Proof of Claim 1. As $(\mathcal{F}, \mathcal{T})$ is normalized, $V := V_{f(\mathcal{U})} = V_{\mathcal{U}} = \mathsf{N}_{\mathrm{def}}^{\mathcal{T}} \cup (\mathsf{N}_{\mathrm{def}}^{\mathcal{T}})^2$. In order to distinguish between vertices from $\mathcal{G}_{f(\mathcal{U})}$ and vertices from $\mathcal{G}_{\mathcal{U}}$, denote every vertex $v \in \mathcal{G}_{\mathcal{U}}$ by v'. We show that

$$Z := \{(v, v') \mid v \in \mathsf{N}_{\mathrm{def}}^{\mathcal{T}} \cup (\mathsf{N}_{\mathrm{def}}^{\mathcal{T}})^2\} \cup \{(v, (v, v)') \mid v \in \mathsf{N}_{\mathrm{def}}^{\mathcal{T}}\}$$

has the desired properties. Consider Z^{-1} first. By construction, $L_{\mathcal{U}}(v') \subseteq L_{f(\mathcal{U})}(v)$ for every $v \in V$ and $E_{\mathcal{U}} \subseteq E_{f(\mathcal{U})}$, implying that the identical subrelation of Z^{-1} satisfies Conditions (S1) and (S2) for simulation relations. For every pair in Z^{-1} of the form

$((v,v)',v)$, (S1) holds by definition of the graph product because always $L_{\mathcal{U}}((v,v)') = L_{\mathcal{U}}(v')$. (S2) holds for every pair $((v,v)',v)$ because every edge $((v,v)'\,r\,(w,w)') \in E_{\mathcal{U}}$ corresponds to the edge $(v'\,r\,w') \in E_{\mathcal{U}}$ which also occurs in $E_{f(\mathcal{U})}$ (named $(v\,r\,w)$) with $((w,w)',w) \in Z^{-1}$.

Now consider Z. In order to show that (S1) holds, we show that $L_{f(\mathcal{U})}(v) \subseteq L_{\mathcal{U}}(v')$ for every v. This suffices for Z because $L_{f(\mathcal{U})}(v) \supseteq L_{\mathcal{U}}(v')$ by construction and $L_{f(\mathcal{U})}(v,v) = L_{f(\mathcal{U})}(v)$ for every $v \in V_{f(\mathcal{U})}$. We distinguish between two cases for v.

If $v = A$ for some $A \in \mathsf{N}_{\mathrm{def}}^{\mathcal{T}}$ then $L_{f(\mathcal{U})}(v) = \mathsf{def}_{f(\mathcal{U})}(A) \cap \mathsf{N}_{\mathrm{prim}}$. As the $\mathcal{F}$-completion only adds descriptive consequences from $\mathcal{F}$ and as no defined name from $\mathsf{N}_{\mathrm{def}}^{\mathcal{T}}$ occurs anywhere in $f(\mathcal{T})^2$, it is easy to see that $\mathsf{def}_{f(\mathcal{U})}(A)$ equals $\mathsf{def}_{f(f(\mathcal{T}))}(A)$. As descriptive consequences of $\mathcal{F}$ and $f(\mathcal{T})$ are by definition already descriptive consequences of $\mathcal{F}$ and $\mathcal{T}$, $\mathsf{def}_{f(f(\mathcal{T}))}(A)$ equals $\mathsf{def}_{f(\mathcal{T})}(A)$, implying $L_{f(\mathcal{U})}(v) = L_{\mathcal{U}}(v')$.

If $v = (A,B)$ for some $(A,B) \in \mathsf{N}_{\mathrm{def}}^{\mathcal{T}}$ then $L_{f(\mathcal{U})}(v) = \mathsf{def}_{f(\mathcal{U})}(A,B) \cap \mathsf{N}_{\mathrm{prim}}$. Consider some $P \in \mathsf{def}_{f(\mathcal{U})}(A,B) \cap \mathsf{N}_{\mathrm{prim}}$. If $P \in L_{\mathcal{T}^2}$ then obviously also $P \in L_{\mathcal{U}}(v')$. Otherwise, P is a descriptive consequence of (A,B) w.r.t. $\mathcal{F} \cup f(\mathcal{T})^2$. But then, by definition of the lcs, P is also a consequence of both A and B in $\mathcal{F} \cup f(\mathcal{T})$, and thus a descriptive consequence of A and B in $\mathcal{F} \cup \mathcal{T}$ because $f(\mathcal{T})$ already contains all descriptive consequences of $\mathcal{F} \cup f(\mathcal{T})$. Thus, P occurs in $L_{f(\mathcal{T})}(A)$ and in $L_{f(\mathcal{T})}(B)$, and therefore also in $L_{f(\mathcal{T})^2}(A,B) \subseteq L_{\mathcal{U}}((A,B)')$, as required.

It remains to show that (S2) holds for Z. Consider some $(v\,r\,w) \in E_{f(\mathcal{U})}$ and some $u' \in V_{\mathcal{U}}$ with $(v,u') \in Z$. We again distinguish two cases for v.

If $v = A$ for some $A \in \mathsf{N}_{\mathrm{def}}^{\mathcal{T}}$ then $u' = A'$ or $u' = (A,A)'$. In the first case, it is easy to see that $(A'\,r\,w') \in E_{\mathcal{U}}$ because $\mathsf{def}_{f(\mathcal{U})}(A) = \mathsf{def}_{\mathcal{U}}(A)$, as argued above, and $(w,w') \in Z$ by construction. If $u' = (A,A)'$ then $((A,A)'\,r\,(w,w)') \in E_{\mathcal{U}}$ by definition of the graph product and because $(A'\,r\,w') \in E_{\mathcal{U}}$, as argued above. Moreover, $(w,(w,w)') \in Z$ by construction.

If $v = (A,B)$ for some $(A,B) \in \mathsf{N}_{\mathrm{def}}^{\mathcal{T}^2}$ and $(v,u) \in Z$ then $u' = (A,B)'$ and either $w \in \mathsf{N}_{\mathrm{def}}^{\mathcal{T}}$ or $w \in \mathsf{N}_{\mathrm{def}}^{\mathcal{T}^2}$. If $w \in \mathsf{N}_{\mathrm{def}}^{\mathcal{T}^2}$ then $(v\,r\,w) \in E_{\mathcal{T}^2} \subseteq E_{\mathcal{U}}$ with $(w,w) \in Z$ as required. Otherwise, $w = C$ for some $C \in \mathsf{N}_{\mathrm{def}}^{\mathcal{T}}$, implying that $C = A_P$ for some $P \in \mathsf{N}_{\mathrm{prim}}$ and that the edge $((A,B)\,r\,A_P)$ corresponds to the existential restriction $\exists r.A_P$ added to the definition of (A,B) by f. Hence, $\exists r.P$ is a descriptive consequence of (A,B) w.r.t. $\mathcal{F} \cup \mathcal{U}$, implying that $\exists r.P$ is a descriptive consequence of both A and B w.r.t. $\mathcal{F} \cup f(\mathcal{T})$ and thus also a consequence of A and B w.r.t. $\mathcal{F} \cup \mathcal{T}$. Consequently, $\exists r.A_P$ occurs in $\mathsf{def}_{f(\mathcal{U})}(A)$ and $\mathsf{def}_{f(\mathcal{U})}(B)$. Hence, $\exists r.(A_P,A_P) \in \mathsf{def}_{f(\mathcal{U})^2}(A,B)$, implying $((A,B)\,r\,(A_P,A_P)') \in E_{\mathcal{U}}$ and $(A_P,(A_P,A_P)') \in Z$ as required.

It remains to show that (A,B) in $\mathcal{U}$ is the least subsumer of A and B.

(2) Consider a conservative extension $(\mathcal{F},\mathcal{U} \cup \mathcal{V})$ of $(\mathcal{F},\mathcal{U})$ and some $D \in \mathsf{N}_{\mathrm{def}}^{\mathcal{V}}$ such that $A \sqsubseteq_{\mathrm{gfp},\mathcal{F},\mathcal{U}\cup\mathcal{V}} D$ and $B \sqsubseteq_{\mathrm{gfp},\mathcal{F},\mathcal{U}\cup\mathcal{V}} D$. By Theorem 3.3.18, this implies $A \sqsubseteq_{\mathrm{gfp},f(\mathcal{U}\cup\mathcal{V})} D$ and analogous for B.

Claim 2. There exists a modification $\mathcal{V}'$ of $\mathcal{V}$ extending the definitions in $\mathcal{V}$ by additional conjuncts and a simulation relation Z such that $Z\colon \mathcal{G}_{f(\mathcal{U}\cup\mathcal{V})} \rightleftharpoons \mathcal{G}_{\mathcal{U}\cup\mathcal{V}'}$, $Z^{-1} : \mathcal{G}_{\mathcal{U}\cup\mathcal{V}'} \rightleftharpoons \mathcal{G}_{f(\mathcal{U}\cup\mathcal{V}')}$, and the identical relation is a subrelation of Z.

As a consequence, D in $f(U \cup \mathcal{V})$ is equivalent to D in $U \cup \mathcal{V}'$, a conservative extension of $\mathcal{U}$. But as (A,B) is the gfp-lcs of A and B in U, this by definition implies $D \sqsubseteq_{\mathrm{gfp},f(\mathcal{U}\cup\mathcal{V})} (A,B)$, which by Theorem 3.3.18 implies $D \sqsubseteq_{\mathrm{gfp},\mathcal{F},\mathcal{U}\cup\mathcal{V}} (A,B)$, as required.

Proof of Claim 2. By definition, no defined names from $\mathcal{V}$ occur in $\mathcal{U}$, implying that the $\mathcal{F}$-completion of definitions in $\mathcal{U}$ does not depend on $\mathcal{V}$. Hence, $f(\mathcal{U} \cup \mathcal{V})$ equals $f(\mathcal{U}) \cup \mathcal{V}'$, where

$$\mathcal{V}' = \{A \equiv C \in f(\mathcal{U} \cup \mathcal{V}) \mid A \in \mathsf{N}_{\mathrm{def}}^{\mathcal{V}}\},$$

i.e., $\mathcal{V}'$ is the subset of $f(\mathcal{U} \cup \mathcal{V})$ containing the definitions of concept names from $\mathcal{V}$. By Claim 1, there exists a simulation relation Z_1 between $f(\mathcal{U})$ and $\mathcal{U}$ with the desired properties. Define $Z := Z_1 \cup \{(A, A') \mid A \in \mathsf{N}_{\text{def}}^{\mathcal{V}}\}$. As $f(\mathcal{U} \cup \mathcal{V}) = f(\mathcal{U}) \uplus \mathcal{V}'$, it immediately follows that Z satisfies Claim 2. □

The above Lemma proves that the hybrid lcs of two concepts A, B defined in a hybrid TBox $(\mathcal{F}, \mathcal{T})$ can be obtained by computing the gfp-lcs of A and B w.r.t. the $\mathcal{F}$-completion $f(\mathcal{T})$ of $\mathcal{T}$. As the lcs of arbitrary arity can be reduced to the binary lcs, the above results immediately carry over to the n-ary lcs. As the reduction from hybrid to cyclic $\mathcal{EL}$-TBoxes can be computed in polynomial time, see Corollary 3.3.19, and as the lcs algorithm for cyclic $\mathcal{EL}$-TBoxes with gfp-semantics has already been studied [Baa03a], we obtain an lcs algorithm for hybrid $\mathcal{EL}$-TBoxes with the following properties.

Corollary 4.4.24 *Let $(\mathcal{F}, \mathcal{T})$ be a normalized hybrid $\mathcal{EL}$-TBox and $A, B \in \mathsf{N}_{\text{def}}^{\mathcal{T}}$. Then the lcs of A and B always exists and can be computed in polynomial time in the size of $(\mathcal{F}, \mathcal{T})$. The lcs of arbitrary arity w.r.t. hybrid $\mathcal{EL}$-TBoxes can be computed in exponential time in the size of the input and is of exponential size in the size of the input in the worst-case.*

To complete the picture of non-standard inferences w.r.t. hybrid $\mathcal{EL}$-TBoxes, we extend the most-specific concept defined for cyclic $\mathcal{EL}$-TBoxes with gfp-semantics in [Baa03a] to hybrid $\mathcal{EL}$-TBoxes. The underlying idea is the same as seen above for the lcs, i.e., the usual notion of conservative extensions is replaced by hybrid conservative extensions and gfp-subsumption is replaced by hybrid subsumption.

Definition 4.4.25 (Hybrid msc)
Let $(\mathcal{F}, \mathcal{T}_1)$ be a hybrid TBox and $\mathcal{A}$ an $\mathcal{EL}$-ABox containing the individual name a. Let $(\mathcal{F}, \mathcal{T}_2)$ be a conservative extension of $(\mathcal{F}, \mathcal{T}_1)$ with $C \in \mathsf{N}_{\text{def}}^{\mathcal{T}_2}$. Then, C in $(\mathcal{F}, \mathcal{T}_2)$ is the *most-specific concept (msc) of a in $(\mathcal{F}, \mathcal{T}_1)$ and A* iff the following conditions hold.

1. a is an instance of C w.r.t. $(\mathcal{F}, \mathcal{T}_2)$ and $\mathcal{A}$; and
2. If $(\mathcal{F}, \mathcal{T}_3)$ is a conservative extension of $(\mathcal{F}, \mathcal{T}_2)$ and $D \in \mathsf{N}_{\text{def}}^{\mathcal{T}_3}$ such that a is an instance of D w.r.t. $(\mathcal{F}, \mathcal{T}_3)$ and $\mathcal{A}$ then $C \sqsubseteq_{\text{gfp},\mathcal{F},\mathcal{T}_3} D$. □

It can be shown similarly to Lemma 4.4.23 that the hybrid msc of an individual a w.r.t. $(\mathcal{F}, \mathcal{T})$ and $\mathcal{A}$ is obtained by computing the gfp-msc of a w.r.t. the $\mathcal{F}$-completion $f(\mathcal{T})$ of $\mathcal{T}$. Hence, we obtain the following corollary for the msc.

Corollary 4.4.26 *Let $(\mathcal{F}, \mathcal{T})$ be a normalized hybrid $\mathcal{EL}$-TBox, $\mathcal{A}$ be an $\mathcal{EL}$-ABox, and $a \in \mathsf{N}_{\text{nom}}^{\mathcal{A}}$ an individual name. Then the msc of a w.r.t. $(\mathcal{F}, \mathcal{T})$ and $\mathcal{A}$ always exists and can be computed in polynomial time.*

In the following section, we close our theoretical examination of non-standard inferences by showing how matching problems w.r.t. hybrid $\mathcal{EL}$-TBoxes can be solved.

4.4.6 Solving matching problems w.r.t. hybrid $\mathcal{EL}$-TBoxes

It has been shown in Section 4.4.4 how matching problems w.r.t. cyclic $\mathcal{EL}$-TBoxes with gfp-semantics can be solved, the main ingredient being the gfp-lcs w.r.t. cyclic $\mathcal{EL}$-TBoxes with gfp-semantics from [Baa03a]. As the latter has been extended from cyclic to hybrid TBoxes in the previous subsection, it seems natural to use the same approach in order to solve matching problems w.r.t. hybrid $\mathcal{EL}$-TBoxes. In order to define a matching algorithm for hybrid TBoxes, we first have to extend our notion of a pattern TBox to hybrid TBoxes.

Definition 4.4.27 (Hybrid pattern TBox)
A hybrid *$\mathcal{EL}$-pattern TBox* $\mathcal{T}$ is a pair $(\mathcal{F}, \mathcal{T})$ of a general $\mathcal{EL}$-TBox $\mathcal{F}$ defined over N_{prim} and N_{role}, and an $\mathcal{EL}$-pattern TBox defined over N_{def}, N_{prim}, and N_{role}. □

Hence, hybrid pattern TBoxes extend ordinary pattern TBoxes in the obvious way. Conservative extensions and instantiations of hybrid pattern TBoxes are defined analogous to their cyclic counterparts, i.e., they affect only $\mathcal{T}$ and leave $\mathcal{F}$ unchanged. We can now immediately extend the notion of matching problems to hybrid pattern TBoxes.

Definition 4.4.28 (Matching problem)
Let $(\mathcal{F}, \mathcal{T})$ be a hybrid $\mathcal{EL}$-pattern TBox with $A, B \in \mathsf{N}_{\text{def}}^{\mathcal{T}}$. Moreover, let $\mathsf{N}_{\text{var}}^{\mathcal{T}}(A) = \emptyset$. Then $A \equiv^{?}_{\text{gfp},\mathcal{F},\mathcal{T}} B$ is a *hybrid $\mathcal{EL}$-matching problem modulo equivalence w.r.t.* $(\mathcal{F}, \mathcal{T})$. □

Note that, despite the restriction of A to defined concept names from $\mathcal{T}$, concept patterns can also be matched against concept names defined in $\mathcal{F}$. For instance, in order to match a concept pattern B defined in $\mathcal{T}$ against some $P \in \mathsf{N}_{\text{con}}^{\mathcal{T}}$ from $\mathcal{F}$, it suffices to extend $\mathcal{T}$ by a definition of the form $A_P \equiv P$, with A_P a fresh concept name, and solve the matching problem $A_P \equiv^{?}_{\text{gfp},\mathcal{F},\mathcal{T}} B$. Clearly, one can also define concept patterns using only names from $\mathcal{F}$.

Solutions to hybrid $\mathcal{EL}$-matching problems can now defined analogous to matchers for matching problems w.r.t. cyclic TBoxes.

Definition 4.4.29 (Matcher)
Let $A \equiv^{?}_{\text{gfp},\mathcal{F},\mathcal{T}} B$ be a hybrid $\mathcal{EL}$-matching problem and let $(\mathcal{F}, \mathcal{T}')$ be an instantiation of $(\mathcal{F}, \mathcal{T})$. Then $(\mathcal{F}, \mathcal{T}')$ is a *matcher of* $A \equiv^{?}_{\text{gfp},\mathcal{F},\mathcal{T}} B$ iff $A \equiv_{\text{gfp},\mathcal{F},\mathcal{T}'} B$. □

Using the reduction from Section 3.3.2, we can now define a matching algorithm for hybrid TBoxes as follows.

Definition 4.4.30 ($\mathsf{match}_{\text{hy}}$)
Let $(\mathcal{F}, \mathcal{T})$ be a normalized hybrid $\mathcal{EL}$-TBox and let $A \sqsubseteq^{?}_{\text{gfp},\mathcal{F},\mathcal{T}} B$ be a hybrid $\mathcal{EL}$-matching problem. Then define

$$\mathsf{match}_{\text{hy}}(A \sqsubseteq_{\text{gfp},\mathcal{F},\mathcal{T}} B) := \{(\mathcal{F}, (\mathcal{T}' \setminus f(\mathcal{T})) \cup \mathcal{T}) \mid \mathcal{T}' \in \mathsf{match}(A \sqsubseteq_{\text{gfp},f(\mathcal{T})} B)\}. \quad \square$$

In the above definition, $f(\mathcal{T})$ denotes the $\mathcal{F}$-completion of $\mathcal{T}$ from Definition 3.3.14 and match the matching algorithm for cyclic $\mathcal{EL}$-TBoxes from in Definition 4.4.13. Hence, the algorithm $\mathsf{match}_{\text{hy}}$ proceeds in three main steps. Firstly, the input hybrid pattern TBox $(\mathcal{F}, \mathcal{T})$ is translated into an equivalent[3] cyclic pattern TBox $f(\mathcal{T})$. Secondly, for the translated matching problem $A \sqsubseteq_{\text{gfp},f(\mathcal{T})} B$, the algorithm match computes all minimal solutions and returns them in the form of instantiations $\mathcal{T}'$ of $f(\mathcal{T})$. Thirdly, the solution is returned as a set of instantiations of *hybrid* pattern TBoxes. How exactly these hybrid instantiations are defined deserves a closer look.

As every instantiation $\mathcal{T}'$ returned by the algorithm match is a conservative extension of $f(\mathcal{T})$ and not $\mathcal{T}$, $\mathcal{T}'$ already completely specifies a solution to the initial hybrid matching problem. Or, in other words, $\mathcal{F}$ becomes redundant. As we are interested in *hybrid* instantiations of $(\mathcal{F}, \mathcal{T})$, and not of $(\mathcal{F}, f(\mathcal{T}))$, we modify every $\mathcal{T}'$ by removing $f(\mathcal{T})$ and replacing it by the original TBox $\mathcal{T}$, i.e., compute $(\mathcal{T}' \setminus f(\mathcal{T})) \cup \mathcal{T}$. This modification preserves equivalence as a direct consequence of the correctness of the $\mathcal{F}$-completion shown in Section 3.3.2. Using the result on the hybrid lcs from Lemma 4.4.23, one can show the following corollary.

[3] Treating variables as atomic concepts.

Corollary 4.4.31 *Let $(\mathcal{F},\mathcal{T})$ be a normalized hybrid $\mathcal{EL}$-TBox and let $A \sqsubseteq_{\mathrm{gfp},\mathcal{F},\mathcal{T}} B$ be an $\mathcal{EL}$-matching problem w.r.t. $(\mathcal{F},\mathcal{T})$. Then, $\mathsf{match}_{\mathrm{hy}}(A \sqsubseteq_{\mathrm{gfp},\mathcal{F},\mathcal{T}} B)$ computes an s-complete set of matchers to $A \sqsubseteq_{\mathrm{gfp},\mathcal{F},\mathcal{T}} B$.*

The complexity results obtained in the previous section together with the fact that $f(\mathcal{T})$ can be computed in polynomial time in the size of $(\mathcal{F},\mathcal{T})$ immediately imply the following complexity results.

Corollary 4.4.32 *Deciding the solvability of matching problems modulo subsumption w.r.t. hybrid $\mathcal{EL}$-TBoxes is tractable. Deciding the solvability of matching problems modulo equivalence w.r.t. hybrid $\mathcal{EL}$-TBoxes is NP-hard.*

The solutions to a matching problem w.r.t. hybrid $\mathcal{EL}$-TBoxes can be exponential in number and of exponential size in the input matching problem. They can be computed by a deterministic exponential-time algorithm.

It is open whether the solvability of matching problems modulo equivalence w.r.t. hybrid $\mathcal{EL}$-TBoxes is in NP. Note that that additional rewriting might be desirable in order to present the solutions of $\mathsf{match}_{\mathrm{hy}}$ more succinctly because $\mathcal{T}'$ can contain the n-ary product of $f(\mathcal{T})$ which might contain information already implied by $\mathcal{F}$. Moreover, it might be interesting to apply the above approach to matching problems defined over general TBoxes if only the decision problem is of interest.

In the remainder of this section, we briefly describe how to utilize the $\mathcal{F}$-completion to reduce other non-standard inferences from hybrid to cyclic TBoxes.

In the case of the lcs, consider a hybrid $\mathcal{EL}$-TBox $(\mathcal{F},\mathcal{T})$. We want to compute the lcs of defined concepts $A_1,\ldots,A_n \in \mathsf{N}^{\mathcal{T}}_{\mathrm{def}}$. Again, note that auxiliary definitions can be added to $\mathcal{T}$ in order to directly use concepts from $\mathcal{F}$. We know by Lemma 4.4.9 that the gfp-lcs w.r.t. cyclic $\mathcal{EL}$-TBoxes is simply computed by the graph product. Hence, the lcs of $A_1,\ldots,A_n \in \mathsf{N}^{\mathcal{T}}_{\mathrm{def}}$ w.r.t. $(\mathcal{F},\mathcal{T})$ corresponds to the concept $(A_1,\ldots,A_n)$ defined in the n-ary product $f(\mathcal{T})^n$. Therefore, it suffices to extend $(\mathcal{F},\mathcal{T})$ to $(\mathcal{F},\mathcal{T}\cup f(\mathcal{T})^n)$ and return $(A_1,\ldots,A_n)$. Note that the foundation $\mathcal{F}$ is not used here, for which reason some kind of rewriting (using $\mathcal{F}$) might be desirable.

The case of the msc is similar. Given a hybrid $\mathcal{EL}$-TBox $(\mathcal{F},\mathcal{T})$, an ABox $\mathcal{A}$ and some individual $a \in \mathsf{N}^{\mathcal{A}}_{\mathrm{nom}}$, we compute the msc of a w.r.t. $\mathcal{A}$ and $(\mathcal{F},\mathcal{T})$ as the msc of a w.r.t. $\mathcal{A}$ and $f(\mathcal{T})$. The results for the msc from [Baa03a] guarantee that the correct concept is computed. Moreover, as the msc can be computed in polynomial time in the cyclic case, we immediately obtain the following corollary.

Corollary 4.4.33 *The lcs w.r.t. a hybrid $\mathcal{EL}$-TBox always exists, it can be of exponential size in the size of the input TBox, and it can be computed in deterministic exponential time. The binary lcs w.r.t. a hybrid $\mathcal{EL}$-TBox can be computed in polynomial time. The msc w.r.t. a hybrid $\mathcal{EL}$-TBox and an ABox $\mathcal{A}$ always exists and it can be computed in deterministic polynomial time.*

4.5 Implementations

In order to show the practicability of the matching algorithms presented in Sections 4.1 and 4.5.2, prototype implementations for both algorithms have been developed and tested. Our implementation of matching in $\mathcal{ALE}$ is presented in Section 4.5.1 while in Section 4.5.2 deals with the implementation of matching in $\mathcal{ALN}$. The fact that the DLs $\mathcal{ALE}$ and $\mathcal{ALN}$ share the common sublanguage $\mathcal{FL}_\neg$ provides an opportunity to examine at the end of

Section 4.5.2 how the, as we shall see, quite different architectures of the algorithms perform in comparison.

4.5.1 Matching in $\mathcal{ALE}$

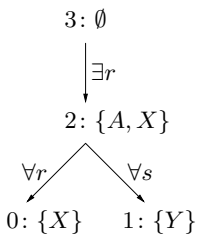
Recall the $\mathcal{ALE}$-matching algorithm from Figure 4.1.1. Clearly, in order to implement $\mathsf{match}_{\mathcal{ALE}}$, three major subtasks must be solved:

1. generate all $\top$-patterns D' of the input pattern D;
2. find all homomorphisms φ from $\mathsf{tree}(D')$ onto $\mathsf{tree}(C)$; and
3. for every variable X, compute the lcs of all subconcepts $C{\downarrow_{\varphi(m)}}$, where X occurs at position m in $\mathsf{tree}(D'^{\top})$.

The first task only refers to the input concept pattern and requires only simple syntactical manipulations. Even the computation of the $\top$-normal form $D'^{\top}$ of a $\top$-pattern D' can be done easily in polynomial time. As (even optimized) implementations of the lcs algorithm for $\mathcal{ALE}$ already exist [BT02a], the third task is simple as soon as D' and φ are determined. The final subsumption test $C \sqsupseteq \sigma(D)$ can also be carried out by a standard reasoner, such as FaCT [Hor98] or Racer [HM01b].

The crucial task is the second one. Constructing homomorphisms between two description trees in the usual top-down way (similar to the lcs algorithms) may lead to subproblems being solved several times over, resulting in an exponential worst-case complexity. To overcome this problem, we choose a dynamic-programming strategy and compose homomorphisms in a bottom-up fashion, thereby storing and re-using sets of admissible destination nodes for every source node. Hence, only polynomially many subproblems are solved for the computation of one homomorphism. The dynamic-programming approach, however, suggests a more sophisticated data structure for the representation of description trees. It proved expedient not to choose an algebraic data structure (as used for the lcs implementations), but to represent a description tree by a certain set of arrays that allows to retrieve 'interesting' aspects important for the computation of homomorphisms more quickly. The following example illustrates this representation.

Example 4.5.1 Consider the $\mathcal{ALE}$-concept pattern $D := \exists r.(A \sqcap X \sqcap \forall r.X \sqcap \forall s.Y)$. The representation of D as a description tree is shown below on the left, with nodes labeled in depth-first order from 0 to 3. In the implementation, D would be represented by nine arrays shown below on the right. For every node, its depth in the description tree, label sets, successors and predecessor are represented by the first six arrays. The next two arrays, valueNode and existNode, are indexed by role name and depth, while varNode is indexed by variable name.



depthA	[0:2 1:2 2:1 3:0]
labels	[0:∅ 1:∅ 2:{A} 3:∅]
variables	[0:{X} 1:{Y} 2:{X} 3:∅]
valueSucc	[0:[r:∅ s:∅] 1:[r:∅ s:∅] 2:[r:{0} s:{1}] 3:[r:∅ s:∅]]
existSucc	[0:[r:∅ s:∅] 1:[r:∅ s:∅] 2:[r:∅ s:∅] 3:[r:{2} s:∅]]
pred	[0:(2, ∀, r) 1:(2, ∀, s) 2:(3, ∃, r)]
valueNode	[r:[1:∅ 2:{0}] s:[1:∅ 2:{1}]]
existNode	[r:[1:{2} 2:∅] s:[1:∅ 2:∅]]
varNode	[X:{0, 2} Y:{1}]

Input: description trees $\mathcal{G}_s =: (N_s, E_s, n_{s0}, L_s)$, $\mathcal{G}_d =: (N_d, E_d, n_{d0}, L_d)$, $n_s \in N_s, n_d \in N_d$
Output: n_d admissible destination of n_s? True iff:

- $\bot \in L_d(n_d)$ and
  - $\mathsf{depth}(n_s) > \mathsf{depth}(n_d)$ or
  - $\mathsf{depth}(n_s) = \mathsf{depth}(n_d)$ and either $n_s = n_{s0}$ and $n_d = n_{d0}$ or both nodes are successors w.r.t. the same quantor and role name
- $\bot \notin L_d(n_d)$
  - $\mathsf{depth}(n_s) = \mathsf{depth}(n_d)$
  - for every successor n'_s of n_s there exists at least one successor n'_d of n_d as admissible destination for n'_s.

Figure 4.5.1: Test for admissible destination nodes

Without going into detail one can see in, e.g., valueNode that there is only one $\forall$-successor w.r.t. role r on depth 2, namely node 0. □

In our implementation, homomorphisms are composed in two steps. In the actual bottom-up computation, a set of admissible destination nodes is computed for every node of the source description tree. These sets are then used to compute the actual homomorphisms. The crucial part in the first step is, for a given source node, to determine whether some destination node is admissible. This part is shown in further detail in Figure 4.5.1. The idea is to test for stricter conditions than the 'local' part of Definition 4.1.1 in order to detect non-admissible destination nodes sooner. For instance, according to Definition 4.1.1, a leaf labeled with $\bot$ is always an admissible destination node. However, if its depth exceeds that of the source node then every mapping containing this pair violates Condition 4 of Definition 4.1.3 at some node on the path from the root to the source node. Note that in case $\bot \notin L_d(n_d)$ a recursive call for n'_s is only performed if this node does not already occur in the relevant arrays. Note also that no backtracking is necessary because of the dynamic programming strategy.

In comparison to the theoretical algorithm, the implemented one contains three optimizations worth mentioning.

- *Preprocessing*: the input concept pattern and concept description are simplified, producing smaller description trees.
- *Necessary conditions*: let $\top(D)$ and $\bot(D)$ denote the concept obtained from the pattern D by replacing all variables in D by $\top$ and $\bot$, respectively. If $C \not\sqsubseteq \top(D)$ or $\bot(D) \not\sqsubseteq C$ then the matching problem $C \equiv^? D$ has no solution.
- $\top$*-patterns*: to generate a top-pattern D' of D is only promising when replacing variables by $\top$ leads to a removal of some subterm in the $\top$-normal form $D'^{\top}$ and hence to a removal of edges in the relevant description tree $\mathsf{tree}(D'^{\top})$. Moreover, if one $\top$-pattern $D'^{\top}$ does admit of a homomorphism then every generalization of D' also does, only yielding solutions potentially not minimal w.r.t. $\sqsubseteq$.

The following section shows some performance tests for the implemented algorithms with the optimizations discussed above.

Benchmarks

An obvious approach to benchmarking our implementation of $\mathcal{ALE}$-matching is to use randomly generated matching problems. Nevertheless, if C and D are generated independently of each other then it is unlikely that a matcher for $C \equiv^? D$ exists. In particular, due to the second optimization (necessary conditions) such matching problems might be solved without even invoking the actual matching algorithm.

Therefore, we randomly generate a concept C first and then construct a concept pattern D from C by randomly replacing subconcepts of C by variables. Matching problems obtained in this way are not necessarily solvable because of multiple occurrences of variables. As a simple example, consider $C := \exists r.A \sqcap \exists s.B$ and $D := \exists r.X \sqcap \exists s.X$. The matching problem $C \equiv^? D$ has no solution.

Note that assuming the concept pattern D to be smaller than C seems justified especially when viewing matching as querying over KBs. Note also that benchmarks computed thus do not make full use of the above optimization techniques, as the necessary-conditions optimization is never relevant for the problems generated in the above way.

Our implementation uses Common LISP as underlying implementation language. The benchmarks have been measured on a standard PC with one 1.7 GHz Intel Pentium-4 processor and 512 MB of memory. In a first experiment, a total of 1,200 matching problems (in 10 groups, using different parameters for the random generation) have been examined. Taking overall averages, the concept description C had an average size of 518 with a maximum of 992, and the concept pattern D had size 185 with a maximum of 772. The matching algorithm on average took 1.2 seconds to solve the problem, the observed maximum was 58.2 seconds.

More details on the behaviour of the $\mathcal{ALE}$-matching algorithm can be found in Section 4.5.2, where in a second experiment, $\mathsf{match}_{\mathcal{ALE}}$ is applied to $\mathcal{FL}_\neg$-matching problems.

4.5.2 Matching in $\mathcal{ALN}$

In order to implement the $\mathcal{ALN}$-matching algorithm introduced in Section 4.2.2, appropriate data structures for the representation of concept descriptions, concept patterns, and tree-like automata are necessary.

As the algorithm is defined w.r.t. the role languages of the $\mathcal{FL}_0$-normal form of its input, it seems expedient to begin by translating the input matching problem into an array of sets of lists over symbols, the symbols representing the alphabet N_{role}. Our data structure for tree-like automata resembles the inductive representation of trees: a vector whose elements are either atomic objects or again vectors. In our case, we only additionally have to discriminate non-final from final nodes and ordinary leaves from those accepting $\mathsf{N}^*_{\text{role}}$. In order to decide word-problems more quickly, vectors representing non-leaf nodes are implemented as arrays instead of lists.

The overall strategy of the implementation corresponds to the steps described in Section 4.2.2:

1. represent the input role languages as tree-like automata;
2. compute a candidate solution σ by operations on tree-like automata; and
3. verify the system of formal language equations from Lemma 4.2.5 in the way described above in Section 4.2.2.

As implementation language, we chose Common LISP because it proved well-suited to realize our representation of tree-like automata. Moreover a LISP implementation makes our

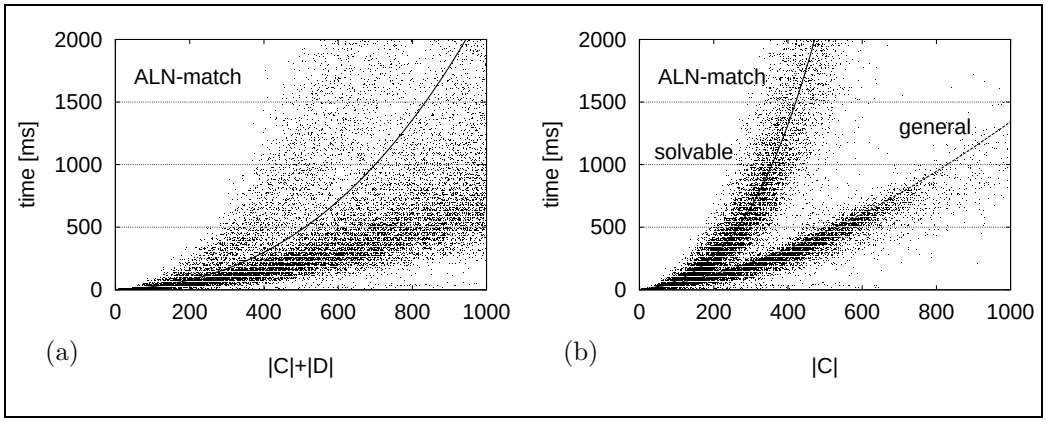


Figure 4.5.2: Benchmarks for matching in $\mathcal{ALN}$

algorithm compatible to the DL system SONIC [Tur05, TK04] which provides an interface to the knowledge editor PROTÉGÉ [GMF+03], see also Section 1.1. This may help to make our algorithm available, and usable, by domain experts using DL systems.

Benchmarks

In order to test the performance of our implementation on a sufficiently large set of data, we again had to resort to randomly generated matching problems. For the same reason as described above, random $\mathcal{ALN}$-matching problems were generated in a two stage process, first randomly generating a concept C and then constructing a concept pattern D from C by randomly replacing sub-concepts of C by variables.

The generated random matching problems were influenced by a vector of probabilities controlling the depth and width of the resulting concept C as well as the frequency of the different constructors available in $\mathcal{ALN}$ and the variables in D. Our benchmarks comprise a total of about 22,000 matching problems in 220 groups, each of which was generated with a unique probability vector. Moreover, we have generated another 12,000 matching problems which, though random, were constructed to be always solvable. The maximum problem size, i.e., the sum of the sizes of C and D, was limited by 1,000. The benchmarks were measured on a standard PC with one 1.7 GHz Pentium-4 processor and 512 MB of memory. Computing overall averages, the algorithm takes 0.8 seconds to solve a matching problem of size 528 with D being two thirds the size of C.

Figure 4.5.2 gives a more detailed account of our findings. Diagram (a) shows the result of our benchmarks as a scatterplot together with a fitting function computed by the least-squares method. One dot in the diagram represents one matching problem $C \equiv^? D$. In the diagram, the horizontal position of every dot represents the sum of the sizes C and D while the vertical position represents the time in milliseconds necessary to solve the problem.

The fitting function in Figure 4.5.2(a) not only matches the overall average fairly well, but also shows the general trend of the expected computation time for larger problems. A problem of size, e.g., 800 increases the computation time to about 1.5 seconds. Nevertheless, the 'darker' cluster below the fitting function indicates that the majority of the problems are solved in less than one second.

Astonished by the strong dispersion of the scatterplot in Figure 4.5.2(a), we have rearranged the plot so that the horizontal position of every dot representing a matching

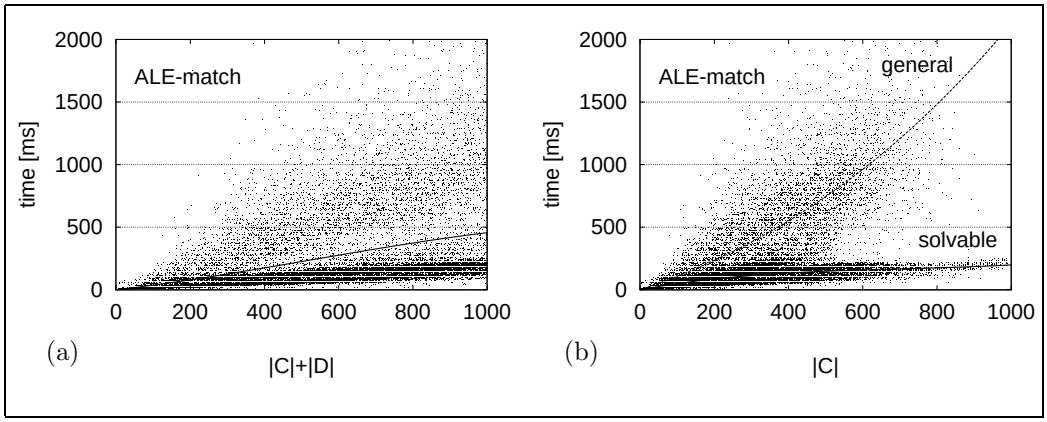


Figure 4.5.3: Benchmarks for the $\mathcal{ALE}$-matching algorithm in $\mathcal{FL}_\neg$

problem $C \equiv^? D$ is determined by the size of C alone, thus ignoring the size of D. This rearrangement produces the scatterplot in Figure 4.5.2(b).

Comparing diagrams (a) and (b), the first immediate observation is that the size of C influences the computation time much stronger than the size of D—although on average the size of D is two thirds the size of C. Moreover, we observe one cluster of simpler matching problems and another cluster of 'hard' ones, where a problem of size 400 on average already seems to take 1.5 seconds to solve. Analysis of our data revealed that the 'hard' cases comprise exactly those problems which, though random, were designed to be solvable.

As we do not have the data to verify these findings by matching problems from realistic applications, we cannot rule out that the above findings are specific to randomly generated matching problems. Nevertheless, it seems expedient to aim future optimizations of the $\mathcal{ALN}$-matching algorithm at improving the computation time for solvable matching problems.

A comparison to the $\mathcal{ALE}$-matching algorithm

The fact that a matching algorithm for the DL $\mathcal{ALE}$ has also been implemented offers the unique opportunity to compare the $\mathcal{ALN}$- to the $\mathcal{ALE}$-matching algorithm head to head on $\mathcal{FL}_\neg$-matching problems, the largest DL in the intersection of $\mathcal{ALN}$ and $\mathcal{ALE}$. This comparison might be interesting for two reasons. Firstly, both algorithms take a totally different approach to solving a matching problem $C \equiv^? D$. While the $\mathcal{ALN}$-algorithm solves a system of formal language equations, the $\mathcal{ALE}$-algorithm tries to construct homomorphisms from the description tree of D into that of C. Secondly, the $\mathcal{ALN}$-algorithm exploits the fact that $\mathcal{FL}_\neg$-matching problems have at most one solution while the $\mathcal{ALE}$-algorithm might look for several ones.

For our comparison, we have generated a set of 34,000 $\mathcal{FL}_\neg$-matching problems in the way described above. The results for the $\mathcal{ALN}$-algorithm are very similar to the ones discussed above. On average, a problem of size 539 was solved in 1.2 seconds by the $\mathcal{ALN}$-algorithm, compared to just 0.25 seconds by the algorithm for $\mathcal{ALE}$. The resulting scatterplots for the $\mathcal{ALN}$-algorithm applied to $\mathcal{FL}_\neg$-matching problems are not shown here because they were almost identical to Figure 4.5.2. The scatterplots for the $\mathcal{ALE}$-algorithm are shown in Figure 4.5.3.

The plot in Figure 4.5.3(a) shows that the majority of matching problems is solved in less

than 0.25 seconds with relatively fewer cases strongly deviating upwards. Moreover, the fitting function indicates that even a problem of size 1,000 is usually solved in about 0.5 seconds.

The discrimination by ordinary matching problems and those designed to be solvable, see Figure 4.5.3(b), shows that our findings from the $\mathcal{ALN}$-algorithm are exactly *reversed*. The $\mathcal{ALE}$-algorithm apparently had no difficulty with solvable matching problems while the 'hard' cases comprise those problems of which many have no solution.

The above findings suggests to direct further optimization efforts of our $\mathcal{ALN}$-matching algorithm not to identifying unsolvable problems earlier, but rather to enhancing the computation of solutions of solvable matching problems. In the case of our $\mathcal{ALE}$-matching algorithm, this seems exactly reversed.

The overall performance of both matching algorithm suggests a good run-time behaviour: both algorithms on average solve matching problems of size up to 500 in less than one second. The direct comparison, however, surprisingly shows that the $\mathcal{ALE}$-matching algorithm performs much better—although at least theoretically, the polynomial $\mathcal{ALN}$-matching algorithm might be expected to outperform the exponential space $\mathcal{ALE}$-matching algorithm.

CHAPTER 5

APPROXIMATION

Informally, concept approximation means the following: given a concept C defined in a source DL $\mathcal{L}_s$, find a concept D in a destination DL $\mathcal{L}_d$ such that D subsumes C, and D is the most specific concept in $\mathcal{L}_d$ with this property. More precisely, the above describes *upper* approximation. In our case, the source DL is $\mathcal{ALC}$ and the destination DL $\mathcal{ALE}$. The relevant approximation algorithm is presented in Section 5.1. A method to speed up the computation of $\mathcal{ALE}$-approximations of $\mathcal{ALC}$-concept descriptions with certain properties is presented in Section 5.1.3. Finally, in Section 5.2, we briefly discuss approximation from $\mathcal{ALCN}$, the extension of $\mathcal{ALC}$ by number restrictions, to $\mathcal{ALEN}$, the corresponding extension of $\mathcal{ALE}$. An in-depth examination of $\mathcal{ALCN}$-$\mathcal{ALEN}$-approximation is beyond the scope of the present work.

5.1 Approximation from $\mathcal{ALC}$ to $\mathcal{ALE}$

Formally, the $\mathcal{ALE}$-approximation of an $\mathcal{ALC}$-concept description is defined as follows.

Definition 5.1.1 ($\mathcal{ALE}$-approximation)
Let C be an $\mathcal{ALC}$-concept description. An $\mathcal{ALE}$-concept description D is an *$\mathcal{ALE}$-approximation* of C iff

1. $C \sqsubseteq D$; and
2. $D \sqsubseteq E$ for every $\mathcal{ALE}$-concept description E with $C \sqsubseteq E$. □

Computing $\mathcal{ALE}$-approximations means to eliminate disjunctions from $\mathcal{ALC}$-concept descriptions appropriately. In the simple case of a disjunction $C_1 \sqcup C_2$ of two $\mathcal{ALE}$-concepts, the most specific $\mathcal{ALE}$-concept description subsuming $C_1 \sqcup C_2$ is just the lcs of C_1 and C_2. Hence, the disjunction is approximated by the lcs of the disjuncts.

It seems tempting to generalize this approach to approximating every $\mathcal{ALC}$-concept description C by substituting every occurring disjunction with the lcs of the relevant disjuncts. Hence, define $\mathsf{approx}^{\mathrm{triv}}(C)$ as follows: if $C \equiv \bot$, $C \equiv \top$, or $C \in \mathsf{P}$ then $\mathsf{approx}^{\mathrm{triv}}(C) := C$. Otherwise,

$$\begin{aligned}
\mathsf{approx}^{\mathrm{triv}}(C_1 \sqcap \cdots \sqcap C_n) &:= \mathsf{approx}^{\mathrm{triv}}(C_1) \sqcap \cdots \sqcap \mathsf{approx}^{\mathrm{triv}}(C_n) \\
\mathsf{approx}^{\mathrm{triv}}(C_1 \sqcup \cdots \sqcup C_n) &:= \mathsf{lcs}\{\mathsf{approx}^{\mathrm{triv}}(C_1), \ldots, \mathsf{approx}^{\mathrm{triv}}(C_n)\} \\
\mathsf{approx}^{\mathrm{triv}}(Qr.C) &:= Qr.\mathsf{approx}^{\mathrm{triv}}(C) \qquad \text{for all } Q \in \{\exists, \forall\}
\end{aligned}$$

This naive approach, however, does not always compute the most specific $\mathcal{ALE}$-concept description subsuming C, as the following example illustrates.

Example 5.1.2 For atomic concepts A and B, consider $C_{\text{ex}} := (\forall r.B \sqcup (\exists r.B \sqcap \forall r.A)) \sqcap \exists r.A$.

$$\begin{aligned}\mathsf{approx}^{\text{triv}}(C_{\text{ex}}) &\equiv \mathsf{lcs}\{\forall r.B, \exists r.B \sqcap \forall r.A\} \sqcap \exists r.A \\ &\equiv \forall r.\top \sqcap \exists r.A \\ &\equiv \exists r.A\end{aligned}$$

Clearly, $C_{\text{ex}} \sqsubseteq \exists r.(A \sqcap B) \sqsubset \exists r.A \equiv \mathsf{approx}^{\text{triv}}(C_{\text{ex}})$. Thus, the algorithm $\mathsf{approx}^{\text{triv}}$ does not find an optimal solution. □

Another straightforward approach to approximation seems to compute a set of copies of the original $\mathcal{ALC}$-concept description C, replacing every disjunction by only one disjunct. The least common subsumer of these modified copies might be the approximation. The pseudo-approximation $\mathsf{approx}^{\text{split}}(C)$ would thus be defined as $\mathsf{approx}^{\text{split}}(C) := \mathsf{lcs}(\mathsf{split}(C))$, where $\mathsf{split}(C)$ is inductively defined as follows:

$$\begin{aligned}\mathsf{split}(C) &:= \{C\} \quad \text{, if } C \in \mathsf{P}(C) \cup \{\bot, \top\} \\ \mathsf{split}(C_1 \sqcap \cdots \sqcap C_n) &:= \{D_1 \sqcap \cdots \sqcap D_n \mid D_i \in \mathsf{split}(C_i), 1 \leq i \leq n\} \\ \mathsf{split}(C_1 \sqcup \cdots \sqcup C_n) &:= \mathsf{split}(C_1) \cup \cdots \cup \mathsf{split}(C_n) \\ \mathsf{split}(Qr.C) &:= \{Qr.D \mid D \in \mathsf{split}(C)\} \qquad \text{for all } Q \in \{\exists, \forall\}\end{aligned}$$

The above algorithm works correctly for Example 5.1.2: split transforms the input concept description into a set consisting of $\forall r.A \sqcap \exists r.B$ and $\exists r.A \sqcap \forall r.B \sqcap \exists r.B$ the lcs of which yields $\exists r.(A \sqcap B)$. Nevertheless, other examples show that $\mathsf{approx}^{\text{split}}$ is incorrect.

Example 5.1.3 For atomic concepts A and B, let $D_{\mathsf{ex}} := \exists r.A \sqcap \exists r.B \sqcap \forall r.(\neg A \sqcup \neg B)$. Applying the algorithm $\mathsf{approx}^{\text{split}}$ to C yields the following result.

$$\begin{aligned}\mathsf{approx}^{\text{split}}(D_{\mathsf{ex}}) &\equiv \mathsf{lcs}(\mathsf{split}(D_{\mathsf{ex}})) \\ &\equiv \mathsf{lcs}\{\exists r.A \sqcap \exists r.B \sqcap \forall r.\neg A, \exists r.A \sqcap \exists r.B \sqcap \forall r.\neg B\} \\ &\equiv \exists r.\mathsf{lcs}\{A \sqcap \neg A, A \sqcap \neg B\} \ \sqcap \ \exists r.\mathsf{lcs}\{A \sqcap \neg A, B \sqcap \neg B\} \ \sqcap \ \ldots \\ &\equiv \exists r.(A \sqcap \neg B) \ \sqcap \ \exists r.\bot \ \sqcap \ \ldots \\ &\equiv \bot\end{aligned}$$

Thus, the returned concept does not even subsume the input. □

The above examples suggest that, contrary to first glance, eliminating disjunctions from $\mathcal{ALC}$-concepts appropriately is not trivial. The following sections will show that an $\mathcal{ALE}$-approximation algorithm consistent with Definition 5.1.1 still relies the lcs inference, but additionally relies on a normal form for $\mathcal{ALC}$-concept descriptions introduced next.

5.1.1 Formal preliminaries

In the present section, we are concerned with a normal form for $\mathcal{ALC}$-concept descriptions. This normal is used for a structural characterization of subsumption for the asymmetric case of subsumption between an $\mathcal{ALC}$-concept description and an $\mathcal{ALE}$-concept description. With these preliminaries, our algorithm to compute upper approximations of $\mathcal{ALC}$-concept descriptions is introduced formally in Section 5.1.2.

For the remainder of this chapter, let $\mathsf{N}_{\text{role}} = \{r\}$. All results can easily be generalized to the case of finitely many roles. To simplify our notation, denote by P the set of (negated) atomic concepts, i.e., $\mathsf{P} := \mathsf{N}_{\text{con}} \cup \{\neg A \mid A \in \mathsf{N}_{\text{con}}\}$. For every concept description D, denote by $\mathsf{P}(D)$ the subset of atomic concepts in P occurring on the toplevel of D. To simplify our notation, denote by $\mathsf{Ex}(D)$ the set of all existential restrictions and by $\mathsf{val}(D)$ the conjunction over all value restrictions occurring in D. For example, $\mathsf{Ex}(P \sqcap \exists r.Q \sqcap \exists r.\top) = \{P, \top\}$ and $\mathsf{val}(\forall r.P \sqcap \forall r.(P \sqcap Q)) = P \sqcap P \sqcap Q$.

Normal forms

In [Küs01], a so-called 'concept-oriented' normal form for $\mathcal{ALE}$-concepts has been introduced. The idea behind this normal form is to modify $\mathcal{ALE}$-concept descriptions in such a way that at most one value restriction occurs on top-level and inside every existential restriction.

Definition 5.1.4 ($\mathcal{ALE}$-normal form)
Let D be an $\mathcal{ALE}$-concept description. D is in *$\mathcal{ALE}$-normal form*, iff $D = \bot$, $D = \top$, or D is of the form

$$D = \bigsqcap_{A \in \mathsf{P}(D)} A \sqcap \bigsqcap_{C' \in \mathsf{Ex}(D)} \exists r.C' \sqcap \forall r.\mathsf{val}(D)$$

where $\mathsf{val}(D)$ and every concept in $\mathsf{Ex}(D)$ again are in $\mathcal{ALE}$-normal form. □

Note that $\mathsf{P}(D)$ also contains negated atomic concepts. For instance, the $\mathcal{ALE}$-normal form of $A \sqcap \forall r.A \sqcap \exists r.B \sqcap \forall r.\neg B$ yields $A \sqcap \exists r.B \sqcap \forall r.(A \sqcap \neg B)$. Transforming an arbitrary $\mathcal{ALE}$-concept description into $\mathcal{ALE}$-normal form does not increase its size. We additionally define a so-called *propagated* $\mathcal{ALE}$-normal form that makes inconsistencies explicit and propagates value restrictions to existential restrictions.

Definition 5.1.5 (Propagated $\mathcal{ALE}$-normal form)
Let D be an $\mathcal{ALE}$-concept description. Then D is in *propagated $\mathcal{ALE}$-normal form*, iff none of the following normalization rules can be applied at any position in C.

$$\begin{aligned}
P \sqcap \neg P &\longrightarrow \bot \quad \text{, where } P \in \mathsf{N}_{\text{con}} \\
E \sqcap \bot &\longrightarrow \bot \\
\exists r.\bot &\longrightarrow \bot \\
\forall r.\top &\longrightarrow \top \\
E \sqcap \top &\longrightarrow E \\
\forall r.C \sqcap \forall r.D &\longrightarrow \forall r.(C \sqcap D) \\
\exists r.C \sqcap \forall r.D &\longrightarrow \exists r.(C \sqcap D) \sqcap \forall r.D
\end{aligned}$$

□

Observe that the actual propagation in the above sense is done by the last transformation rule. The propagated $\mathcal{ALE}$-normal form is a specialization of the $\mathcal{ALE}$-normal form for which the last but one transformation rule suffices. Transforming a concept description into propagated $\mathcal{ALE}$-normal form, however, can result in exponentially larger concept descriptions because the last transformation rule duplicates the subconcept D.

In order to extend the (ordinary) $\mathcal{ALE}$-normal form to $\mathcal{ALC}$, we require $\mathcal{ALC}$-concept descriptions to be in negation normal form, and additionally enforce that the bottom concept is represented uniquely and that every disjunction is in disjunctive normal form.

Definition 5.1.6 ($\mathcal{ALC}$-normal form)
Let C be an $\mathcal{ALC}$-concept description. C is in *$\mathcal{ALC}$-normal form*, iff $C = \bot$, $C = \top$, or C is of the form

$$C = C_1 \sqcup \cdots \sqcup C_n,$$
$$C_i = \bigsqcap_{A \in \mathsf{P}(C_i)} A \sqcap \bigsqcap_{C' \in \mathsf{Ex}(C_i)} \exists r.C' \sqcap \forall r.\mathsf{val}(C_i)$$

for all $i \in \{1, \ldots, n\}$, where (i) $C_i \not\equiv \bot$ for all i and (ii) $\mathsf{val}(D)$ and every concept in $\mathsf{Ex}(D)$ again are in $\mathcal{ALC}$-normal form. □

It is easy to see that every $\mathcal{ALC}$-concept description can be transformed into $\mathcal{ALC}$-normal form. Consider the following simple example.

Example 5.1.7 For atomic concepts $P, Q \in \mathsf{N}_{\mathrm{con}}$, let $C := \neg\forall r.(P \sqcap Q) \sqcap (Q \sqcup \exists r.\neg P)$. The negation normal form of C yields $\exists r.(\neg P \sqcup \neg Q) \sqcap (Q \sqcup \exists r.\neg P)$. By distributing conjuncts over the disjunction, we obtain $(\exists r.(\neg P \sqcup \neg Q) \sqcap Q) \sqcup (\exists r.(\neg P \sqcup \neg Q) \sqcap \exists r.\neg P)$, the $\mathcal{ALC}$-normal form of C. □

Note that the $\mathcal{ALC}$-normal form of a concept C can be exponentially larger than C even if C contains only atomic concepts. For instance, computing the disjunctive normal form of $(P_1 \sqcup Q_1) \sqcap \cdots \sqcap (P_n \sqcup Q_n)$ produces a concept description of exponential size in n.

Characterization of subsumption

The normal forms introduced in the previous section will now be utilized for a structural characterization of subsumption between an $\mathcal{ALC}$-concept description C and an $\mathcal{ALE}$-concept description D.

Theorem 5.1.8 *Let C be an $\mathcal{ALC}$-concept description in $\mathcal{ALC}$-normal form and D an $\mathcal{ALE}$-concept description in propagated $\mathcal{ALE}$-normal form. Then, $C \sqsubseteq D$ iff*

1. *$C \equiv \bot$ or $D \equiv \top$, or*
2. *$C = C_1 \sqcup \cdots \sqcup C_n$ and for every $i \in \{1, \ldots, n\}$ it holds that*
   - *$\mathsf{P}(D) \subseteq \mathsf{P}(C_i)$;*
   - *$\forall D' \in \mathsf{Ex}(D)\ \exists C' \in \mathsf{Ex}(C_i)\colon C' \sqcap \mathsf{val}(C_i) \sqsubseteq D'$; and*
   - *$\mathsf{val}(C_i) \sqsubseteq \mathsf{val}(D)$.*

Proof. ($\Rightarrow$) Proof by contraposition. Assume $\bot \sqsubset C \sqsubseteq D \sqsubset \top$.

- Assume $\mathsf{P}(D) \not\subseteq \mathsf{P}(C_i)$ for some $i \in \{1, \ldots, n\}$. Then there exists some $P \in \mathsf{P}(D) \setminus \mathsf{P}(C_i)$. By definition of the $\mathcal{ALC}$-normal form, C_i is consistent, implying the existence of a canonical interpretation $\mathcal{I}$ of C_i with $d_{C_i} \in C_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$. By definition, $d_{C_i} \notin P^{\mathcal{I}}$ since $P \notin \mathsf{P}(C_i)$. Thus, $d_{C_i} \notin D^{\mathcal{I}}$ and therefore $C \not\sqsubseteq D$, in contradiction to the assumption.

- Assume for some $D' \in \mathsf{Ex}(D)$ and some $i \in \{1, \ldots, n\}$ that $C' \sqcap \mathsf{val}(C_i) \not\sqsubseteq D'$ for all $C' \in \mathsf{Ex}(C_i)$. Since C_i is consistent, every $C' \in \mathsf{Ex}(C_i)$ has a tree model $\mathcal{I}_{C'}$ with $d_{C'} \in (C' \sqcap \mathsf{val}(C_i))^{\mathcal{I}_{C'}} \setminus (D')^{\mathcal{I}_{C'}}$ for some $d_{C'} \in \Delta^{\mathcal{I}_{C'}}$. W.l.o.g., we may assume

disjoint domains, i.e., $\Delta^{\mathcal{I}_{C'}} \cap \Delta^{\mathcal{I}_{C''}} = \emptyset$ for distinct $C', C'' \in \mathsf{Ex}(C_i)$. Construct a new model $\mathcal{I}$ with

$$\Delta^{\mathcal{I}} := \{d\} \uplus \bigcup_{C' \in \mathsf{Ex}(C_i)} \Delta^{\mathcal{I}_{C'}}$$
$$P^{\mathcal{I}} := \{d \mid P \in \mathsf{P}(C_i)\} \cup \bigcup_{C' \in \mathsf{Ex}(C_i)} P^{\mathcal{I}_{C'}} \quad \text{for every } P \in \mathsf{P}$$
$$r^{\mathcal{I}} := \{(d, d_{C'}) \mid C' \in \mathsf{Ex}(C_i)\} \cup \bigcup_{C' \in \mathsf{Ex}(C_i)} r^{\mathcal{I}_{C'}}.$$

It is easy to see that $d \in C^{\mathcal{I}}$. On the other hand $d \notin D^{\mathcal{I}}$ because for every C', $d_{C'} \notin D'^{\mathcal{I}_{C'}}$, implying $d \notin D^{\mathcal{I}}$, in contradiction to the assumption.

- Assume $\mathsf{val}(C_i) \not\sqsubseteq \mathsf{val}(D)$ for some $i \in \{1, \ldots, n\}$. Then $\mathsf{val}(C_i)$ has a tree model $\mathcal{I}_{\text{val}}$ such that $d_{\text{val}} \in \mathsf{val}(C_i)^{\mathcal{I}_{\text{val}}} \setminus \mathsf{val}(D)^{\mathcal{I}_{\text{val}}}$ for some $d_{\text{val}} \in \Delta^{\mathcal{I}_{\text{val}}}$. We can now extend the model $\mathcal{I}$ introduced for the previous case by adding d_{val} as an r-successor of d. Again, assume $\Delta^{\mathcal{I}} \cap \Delta^{\mathcal{I}_{\text{val}}} = \emptyset$. Then, define $\mathcal{I}'$ by

$$\Delta^{\mathcal{I}'} := \Delta^{\mathcal{I}} \cup \Delta^{\mathcal{I}_{\text{val}}}$$
$$P^{\mathcal{I}'} := P^{\mathcal{I}} \cup P^{\mathcal{I}_{\text{val}}} \quad \text{for every } P \in \mathsf{P}$$
$$r^{\mathcal{I}'} := \{(d, d_{\text{val}})\} \cup r^{\mathcal{I}} \cup r^{\mathcal{I}_{\text{val}}}.$$

As a result, $d \in (C_i^{\mathcal{I}'})$ for all $i \in \{1, \ldots, n\}$ and thus $d \in C^{\mathcal{I}'}$ but on the other hand $d \notin D^{\mathcal{I}'}$.

($\Leftarrow$) 1. Trivial. 2. Let $i \in \{1, \ldots, n\}$. It is sufficient to show that $C_i \sqsubseteq D$. Let $x \in C_i^{\mathcal{I}}$ for any interpretation $\mathcal{I}$ of C_i. Show: $x \in D^{\mathcal{I}}$.

- By assumption, $x \in P^{\mathcal{I}}$ for every $P \in \mathsf{P}(C_i)$. Since, $\mathsf{P}(D) \subseteq \mathsf{P}(C_i)$, $x \in P^{\mathcal{I}}$ for every $P \in \mathsf{P}(D)$.
- Consider an arbitrary $D' \in \mathsf{Ex}(D)$. By assumption, there is some $C' \in \mathsf{Ex}(C_i)$ with $C' \sqcap \mathsf{val}(C_i) \sqsubseteq D'$. Since $x \in (\exists r.C' \sqcap \forall r.\mathsf{val}(C_i))^{\mathcal{I}}$, $x \in (\exists r.D')^{\mathcal{I}}$.
- As $\mathsf{val}(C_i) \sqsubseteq \mathsf{val}(D)$ and $x \in (\mathsf{val}(C_i))^{\mathcal{I}}$, $x \in (\mathsf{val}(D))^{\mathcal{I}}$.

This implies our claim since

$$D^{\mathcal{I}} = \bigcap_{A \in \mathsf{P}(D)} A^{\mathcal{I}} \cap \bigcap_{D' \in \mathsf{Ex}(D)} (\exists r.D')^{\mathcal{I}} \cap (\mathsf{val}(D))^{\mathcal{I}}. \qquad \square$$

We are now prepared to introduce our approximation algorithm formally.

5.1.2 Approximating $\mathcal{ALC}$-concepts in $\mathcal{ALE}$

A definition of the upper approximation is already given by Definition 5.1.1. We now propose a corresponding algorithm to actually *compute* the upper approximation of a given $\mathcal{ALC}$-concept description.

Definition 5.1.9 (approx)
Let C be an $\mathcal{ALC}$-concept description. Then $\mathsf{approx}(C)$ is defined as shown in Figure 5.1.1. $\square$

Input: $\mathcal{ALC}$-concept description C
Output: $\mathcal{ALE}$-approximation of C

1. If $C \equiv \bot$ or $C \equiv \top$ then return $\bot$ or $\top$, respectively.
2. Otherwise, transform C into $\mathcal{ALC}$-normal form $C_1 \sqcup \cdots \sqcup C_n$ and return

$$\begin{aligned}&\bigsqcap_{P \in \bigcap_i \mathsf{P}(C_i)} P \\ &\sqcap \bigsqcap_{(C_1',\ldots,C_n') \in \mathsf{Ex}(C_1)\times\cdots\times\mathsf{Ex}(C_n)} \exists r.\mathsf{lcs}\{\mathsf{approx}(C_i' \sqcap \mathsf{val}(C_i)) \mid 1 \le i \le n\} \\ &\sqcap \forall r.\mathsf{lcs}\{\mathsf{approx}(\mathsf{val}(C_i)) \mid 1 \le i \le n\}.\end{aligned}$$

Figure 5.1.1: The algorithm approx

Note that C is transformed into $\mathcal{ALC}$-normal form in Step 2. Moreover, value restrictions are propagated to existential restrictions by computing $\mathsf{approx}(C_i' \sqcap \mathsf{val}(C_i))$ instead of $\mathsf{approx}(C_i')$ in the recursion. Note also that $C_i' \sqcap \mathsf{val}(C_i)$ is not necessarily in $\mathcal{ALC}$-normal form. Therefore, another normalization is necessary in the recursion computing existential restrictions. To see how the above algorithm works, let us return to Examples 5.1.2 and 5.1.3.

Example 5.1.10 Consider $C_{\text{ex}} = (\forall r.B \sqcup (\exists r.B \sqcap \forall r.A)) \sqcap \exists r.A$ from Example 5.1.2. Applying approx to C_{ex} firstly transforms the input into $\mathcal{ALC}$-normal form, yielding $(\forall r.B \sqcap \exists r.A) \sqcup (\exists r.B \sqcap \forall r.A \sqcap \exists r.A)$. Therefore:

$$\begin{aligned}\mathsf{approx}(C_{\text{ex}}) &= \mathsf{approx}((\forall r.A \sqcap \exists r.B) \sqcup (\exists r.A \sqcap \forall r.B \sqcap \exists r.B)) \\ &= \exists r.\mathsf{lcs}\{\exists r.B \sqcap \forall r.A, \exists r.A \sqcap \forall r.B\} \sqcap \\ &\quad \exists r.\mathsf{lcs}\{\exists r.B \sqcap \forall r.A, \exists r.B \sqcap \forall r.B\} \\ &= \exists r.(A \sqcap B) \ \sqcap \ \exists r.B \\ &\equiv \exists r.(A \sqcap B)\end{aligned}$$

The concept description from Example 5.1.3, $D_{\mathsf{ex}} = \exists r.A \sqcap \exists r.B \sqcap \forall r.(\neg A \sqcup \neg B)$, is already in $\mathcal{ALC}$-normal form. Thus, applying approx yields:

$$\begin{aligned}\mathsf{approx}(D_{\mathsf{ex}}) &= \mathsf{approx}(\exists r.A \sqcap \exists r.B \sqcap \forall r.(\neg A \sqcup \neg B)) \\ &= \exists r.\mathsf{lcs}\{\mathsf{approx}(A \sqcap (\neg A \sqcup \neg B))\} \sqcap \\ &\quad \exists r.\mathsf{lcs}\{\mathsf{approx}(B \sqcap (\neg A \sqcup \neg B))\} \\ &= \exists r.\mathsf{approx}((A \sqcap \neg A) \sqcup (A \sqcap \neg B)) \sqcap \\ &\quad \exists r.\mathsf{approx}((B \sqcap \neg A) \sqcup (B \sqcap \neg B)) \\ &\equiv \exists r.(A \sqcap \neg B) \ \sqcap \ \exists r.(B \sqcap \neg A)\end{aligned}$$

□

The following theorem proves that the algorithm approx always finds the correct approximation in the sense of Definition 5.1.1.

Theorem 5.1.11 *Let C be an $\mathcal{ALC}$-concept description. Then* $\mathsf{approx}(C)$ *is the upper $\mathcal{ALE}$-approximation of C, i.e.,*

1. $C \sqsubseteq \mathsf{approx}(C)$, *and*

2. $\mathsf{approx}(C) \sqsubseteq D$ *for every* $\mathcal{ALE}$*-concept description* D *with* $C \sqsubseteq D$*.*

PROOF. Due to the normalization in Step 2 of approx, we may w.l.o.g. assume that C is in $\mathcal{ALC}$-normal form.

1. It suffices to show by induction over the structure of C that the conditions for subsumption from Theorem 5.1.8 hold. If $C \in \{\bot, \top\}$ then trivially $\mathsf{approx}(C) = C$. The same holds if $C \equiv \bot$ or $C \equiv \top$. Otherwise, $C = C_1 \sqcup \cdots \sqcup C_n$. By induction hypothesis (IH), the claim holds for all subterms of C occurring in existential and value restrictions.

- By definition of approx, $\mathsf{P}(\mathsf{approx}(C)) = \bigcap_{i=1}^{n} \mathsf{P}(C_i) \subseteq \mathsf{P}(C)$.
- For every existential restriction $\exists r.\mathsf{lcs}\{\mathsf{approx}(C_i' \sqcap \mathsf{val}(C_i)) \mid 1 \leq i \leq n\}$ in $\mathsf{approx}(C)$, $C_i' \in \mathsf{Ex}(C_i)$ for every i. By IH, $C_i' \sqcap \mathsf{val}(C_i) \sqsubseteq \mathsf{approx}(C_i' \sqcap \mathsf{val}(C_i))$. Thus, by definition of the lcs, $C_i' \sqcap \mathsf{val}(C_i) \sqsubseteq \mathsf{lcs}\{\mathsf{approx}(C_i' \sqcap \mathsf{val}(C_i)) \mid 1 \leq i \leq n\}$.
- By IH, $\mathsf{val}(C_i) \sqsubseteq \mathsf{approx}(\mathsf{val}(C_i))$ for every $i \in \{1, \ldots, n\}$. Consequently, by definition of the lcs, $\mathsf{val}(C_i) \sqsubseteq \mathsf{lcs}\{\mathsf{approx}(\mathsf{val}(C_i)) \mid 1 \leq i \leq n\}$.

2. W.l.o.g., let D be in $\mathcal{ALE}$-normal form. Proof by induction over the structure of C.

If $C \in \{\bot, \top\}$, then $\mathsf{approx}(C) = C$ trivially is least. The same holds if $C \equiv \bot$ or $C \equiv \top$. Otherwise, $C = C_1 \sqcup \cdots \sqcup C_n$, and by IH, the claim holds for all subterms of C occurring in existential and value restrictions. As $C \sqsubseteq D$, the following facts are implied:

- $\mathsf{P}(D) \subseteq \mathsf{P}(C_i)$ for every $i \in \{1, \ldots, n\}$. As $\mathsf{P}(\mathsf{approx}(C))$ is the intersection over all $\mathsf{P}(C_i)$, $\mathsf{P}(D) \subseteq \mathsf{P}(\mathsf{approx}(C))$.
- For all $D' \in \mathsf{Ex}(D)$ and for all $i \in \{1, \ldots, n\}$, there is one $C' \in \mathsf{Ex}(C_i)$ with $C' \sqcap \mathsf{val}(C_i) \sqsubseteq D'$. By IH, $C' \sqcap \mathsf{val}(C_i) \sqsubseteq \mathsf{approx}(C' \sqcap \mathsf{val}(C_i)) \sqsubseteq D'$ for every i. Hence, due to the lcs, $\mathsf{lcs}\{\mathsf{approx}(C' \sqcap \mathsf{val}(C_i)) \mid 1 \leq i \leq n\} \sqsubseteq D'$.
- For all $i \in \{1, \ldots, n\}$, $\mathsf{val}(C_i) \sqsubseteq \mathsf{val}(D)$. By IH, $\mathsf{val}(C_i) \sqsubseteq \mathsf{approx}(\mathsf{val}(C_i)) \sqsubseteq \mathsf{val}(D)$, implying $\mathsf{lcs}\{\mathsf{approx}(\mathsf{val}(C_i)) \mid 1 \leq i \leq n\} \sqsubseteq \mathsf{val}(D)$.

□

The following simple properties are supposed to give further insight into the relationship between a concept description and its approximation.

Corollary 5.1.12 *Let* C, D *be* $\mathcal{ALC}$*-concept descriptions in* $\mathcal{ALC}$*-normal form. Then,*

1. $C \sqsubseteq D$ *implies* $\mathsf{approx}(C) \sqsubseteq \mathsf{approx}(D)$*;*
2. $\mathsf{approx}(C \sqcap D) \sqsubseteq \mathsf{approx}(C) \sqcap \mathsf{approx}(D)$*;*
3. *not generally* $\mathsf{approx}(C \sqcap D) \equiv \mathsf{approx}(C) \sqcap \mathsf{approx}(D)$*;*
4. $\mathsf{approx}(\exists r.C) \equiv \exists r.\mathsf{approx}(C)$ *and* $\mathsf{approx}(\forall r.C) \equiv \forall r.\mathsf{approx}(C)$*; and*
5. *if* C *has only one disjunct on toplevel then* $\mathsf{val}(\mathsf{approx}(C)) \equiv \mathsf{approx}(\mathsf{val}(C))$

PROOF. 1. The fact $D \sqsubseteq \mathsf{approx}(D)$ implies $C \sqsubseteq D \sqsubseteq \mathsf{approx}(D)$. As $\mathsf{approx}(C)$ is the most specific $\mathcal{ALE}$-concept subsuming C, $\mathsf{approx}(C) \sqsubseteq \mathsf{approx}(D)$.

2. Immediate consequence of (1) and $C \sqcap D \sqsubseteq C$ and $C \sqcap D \sqsubseteq D$.

3. See Example 5.1.2.

4. Trivial.
5. By definition of approx, $\mathsf{val}(\mathsf{approx}(C)) = \mathsf{lcs}\{\mathsf{approx}(\mathsf{val}(C_i)) \mid 1 \leq i \leq n\}$. If only one disjunct exists, i.e. $C = C_1$, this equals $\mathsf{approx}(\mathsf{val}(C))$.

□

Note that equality does not generally hold in (4): $\mathsf{approx}(\exists r.\bot) = \bot \neq \exists r.\mathsf{approx}(\bot)$ and $\mathsf{approx}(\forall r.\top) = \top \neq \forall r.\mathsf{approx}(\top)$

Having shown correctness, the natural next question regards the computational complexity of the algorithm approx. Proving an exponential lower bound is an immediate consequence of the computational complexity of the lcs in $\mathcal{ALE}$.

Corollary 5.1.13 *Computing upper $\mathcal{ALC}$-approximations of $\mathcal{ALE}$-concept descriptions is necessarily worst-case exponential.*

PROOF. Consider two $\mathcal{ALE}$-concept descriptions C_1 and C_2 in $\mathcal{ALE}$-normal form. According to the definition, $\mathsf{approx}(C_1 \sqcup C_2) = \mathsf{lcs}(C_1, C_2)$. It has been shown in [BKM99] that the binary lcs of $\mathcal{ALE}$-concept descriptions can be of exponential size on the size of the input concept descriptions. Hence, any computation algorithm is worst-case exponential. □

In order to establish an upper bound, we show that approx is a double-exponential time algorithm. It is an open problem whether or not this bound is tight.

Lemma 5.1.14 *The algorithm* approx *can be realized as a double-exponential time algorithm.*

PROOF. By definition, approx in Step 2 transforms the entire concept C into $\mathcal{ALC}$-normal form. We show that approx is in double-exponential time if this transformation is done 'on the fly', i.e., role level by role level in every recursion step.

The modified computation of $\mathsf{approx}(C)$ starts by transforming C into $D := D_1 \sqcup \cdots \sqcup D_n$, with no disjunction on the topmost role level of every D_i, while the lower role levels remain unchanged. On the topmost role level, D therefore has at most exponentially many disjuncts, i.e., $n = 2^{p(|C|)}$ for some polynome p, each at most of size $|C|$.

By definition, the following subconcepts are computed for $\mathsf{approx}(D)$

1. $S_{\mathsf{P}} := \bigsqcap_{P \in \bigcap_i \mathsf{P}(D_i)} P$;
2. $S_{(D'_1,\ldots,D'_n)} := \exists r.\mathsf{lcs}\{\mathsf{approx}(D'_i \sqcap \mathsf{val}(D_i)) \mid 1 \leq i \leq n\}$
   for every tuple $(D'_1, \ldots, D'_n)$ with $D'_i \in \mathsf{Ex}(D_i)$ for every i; and
3. $S_{\forall} := \forall r.\mathsf{lcs}\{\mathsf{approx}(\mathsf{val}(C_i)) \mid 1 \leq i \leq n\}$.

S_{P} can be computed in polynomial time in the size of D and thus in exponential time in $|C|$. As n is exponential in $|C|$ and as the number of existential restrictions D'_i in every D_i is bounded by $|D_i|$, the number of different $S_{(D'_1,\ldots,D'_n)}$ is at most double exponential in $|C|$. For every single $S_{(D'_1,\ldots,D'_n)}$, the lcs over a set of exponential cardinality in $|C|$ must be computed. Each element of such a set is of the form $\mathsf{approx}(D'_i \sqcap \mathsf{val}(D_i))$. Hence, approx is recursively invoked on a concept description bounded in size by $|C|$ and with a role depth decreased by one. The same holds for the computation of $S_{\forall}$. Thus, the computation tree of approx is of double exponential size in $|C|$. If the lcs is not evaluated then approx runs in double exponential time. It remains to show that computing all lcs expressions preserves this complexity.

Every lcs is defined over an at most exponential number of concepts of at most of double exponential size in $|C|$. Every concept is in propagated $\mathcal{ALE}$-normal form (Definition 5.1.5), since all concepts returned by approx are. As shown in [BKM99], this implies that the size of the resulting lcs is bounded by the product of the sizes of the input concepts. Evaluating all lcs expressions in the computation tree for $\mathsf{approx}(C)$ in a bottom-up fashion therefore yields concept descriptions of size at most double exponential in $|C|$. Since the depth of the result of every lcs computation is bounded by the role depth of C, the whole computation takes at most double-exponential time in $|C|$. □

It is still an open problem whether the above upper bound is tight, i.e., whether any deterministic approximation algorithm computing the $\mathcal{ALE}$-approximation of $\mathcal{ALC}$-concept descriptions is necessarily worst-case double-exponential time. In [LDLT02] an attempt has been made to answer this question in the negative by proposing a deterministic exponential time algorithm to compute $\mathcal{ALC}$-$\mathcal{ALE}$-approximations. This result, however, has been refuted by the same authors by claiming in [LDLT03] that, even w.r.t. a certain compact representation of concept descriptions, no deterministic exponential time algorithm computing $\mathcal{ALC}$-$\mathcal{ALE}$-approximations exists. To the best of our knowledge, the question is still open.

Independently of the worst-case complexity of approximation discussed above, there are ways to speed up approximations significantly in practical applications. In the following section, we show that approximations can be computed more quickly under certain circumstances.

5.1.3 Speeding up approximations

Our aim is to show that approximating an $\mathcal{ALC}$-concept description can be simplified if the concept description in question satisfies certain syntactic conditions. In order to define these conditions, we need to refer to the constructs found in a certain depth of the syntax trees of concepts. The following definition introduces the relevant notation.

Definition 5.1.15 Let $C := \bigsqcup_{i=1}^{k} C_i$ be an $\mathcal{ALC}$-concept description in $\mathcal{ALC}$-normal form. For $d \in \mathbb{N}$ and for every $r \in \mathsf{N}_{\mathrm{role}}$, the sets $Q_r(C,d)$ and $\mathsf{P}_r(C,d)$ are inductively defined by:

- $Q_r(C,0) := \{\exists \mid \bigcup_{i=1}^{k} \mathsf{Ex}_r(C_i) \neq \emptyset\} \cup \{\forall \mid \bigsqcap_{i=1}^{k} \mathsf{val}_r(C_i) \sqsubset \top\}$
  $\mathsf{P}_r(C,0) := \bigcup_{i=1}^{k} \mathsf{P}(C_i)$
- $Q_r(C,d+1) := \bigcup_{i=1}^{k} \bigcup_{C' \in \mathsf{Ex}_r(C_i)} Q_r(C',d) \cup \bigcup_{i=1}^{k} Q_r(\mathsf{val}_r(C_i),d)$
  $\mathsf{P}_r(C,d+1) := \bigcup_{i=1}^{k} \bigcup_{C' \in \mathsf{Ex}_r(C_i)} \mathsf{P}_r(C',d) \cup \bigcup_{i=1}^{k} \mathsf{P}_r(\mathsf{val}_r(C_i),d).$ □

For an $\mathcal{ALC}$-concept description C and $i \in \mathbb{N}$ the *quantor set* $Q_r(C,i)$ denotes the set of quantors referring to role r used on the ith role-level of C. Hence, every i between 0 and the maximum role-depth of C, the quantor set $Q_r(C,i)$ is a nonempty subset of $\{\forall, \exists\}$. Similarly, the *name set* $\mathsf{P}_r(C,i)$ denotes the set of atomic concepts used on the ith role level inside a quantor referring to r.

In the above definition, we write Ex_r and val_r in order to emphasize that in case of more than one role, Q_r and P_r are still defined individually for every role.

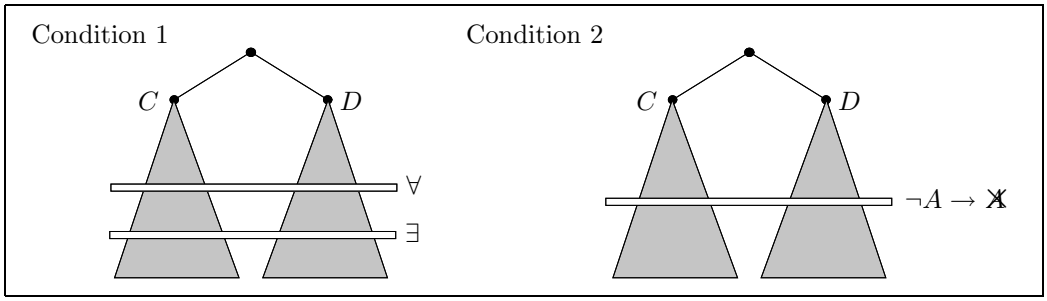


Figure 5.1.2: Nice $\mathcal{ALC}$-concepts

Example 5.1.16 Let $\mathsf{P} := \{P, Q, R, S\}$ and let $C := (\exists r.(P \sqcap Q) \sqcap \forall r.(R \sqcup (\exists r.\neg S)))$. Then $Q_r(C, 0) = \{\forall, \exists\}$, $Q_r(C, 1) = \{\exists\}$ and $Q_r(C, i) = \emptyset$ for every $i \geq 2$. Moreover, $\mathsf{P}_r(C, 0) = \emptyset$, $\mathsf{P}_r(C, 1) = \{P, Q, R\}$, and $\mathsf{P}_r(C, 2) = \{\neg S\}$.

We are now ready to introduce 'nice' $\mathcal{ALC}$-concept descriptions.

Definition 5.1.17 Let C be an $\mathcal{ALC}$-concept description in negation normal form. Then C is *nice* iff for every $d \in \mathbb{N}$ and every $r \in \mathsf{N}_{\text{role}}$ it holds that

1. $|Q_r(C, d)| \leq 1$ and
2. $N_r(C, d)$ does not contain a concept name and its negation. □

Intuitively, C is nice iff (w.r.t. every role r) firstly, on every role level either no $\exists$- or no $\forall$-quantors occur, and secondly, a concept name and its negation do not occur on the same role level. Figure 5.1.2 gives an intuition of how nice concepts looks like for $\mathsf{N}_{\text{role}} = \{r\}$.

It has been shown in [BT02b] that nice $\mathcal{ALC}$-concepts can be approximated conjunct by conjunct, which gives rise to the following theorem.

Theorem 5.1.18 *Let $C \sqcap D$ be a nice $\mathcal{ALC}$-concept description. Then* $\mathsf{approx}(C \sqcap D)$ *is equivalent to* $\mathsf{approx}(C) \sqcap \mathsf{approx}(D)$.

Although this simplification does not improve the worst case complexity obtained in Lemma 5.1.14, it is easy to see that splitting an $\mathcal{ALC}$-concept descriptions into its conjuncts before approximation can drastically reduce the number of existential restrictions generated by the lcs.

Moreover, nice concepts can play an important role when approximating concepts defined in a TBox. Firstly, by a dynamic programming approach, nice-ness of defined concepts can be tested in polynomial time in the size of the TBox, i.e., without expanding the concept in question. Secondly, if several approximations have to be computed w.r.t. a TBox then nice concepts are good candidates for caching: consider a TBox containing the definition, e.g., $A \equiv B \sqcap C$ such that B is a nice defined concept and C is also nice. Then the approximation of B can be re-used for the computation of the approximation of A.

5.2 Other approximations

We finish our section about approximation by mentioning an approximation algorithm for $\mathcal{ALEN}$-approximations of $\mathcal{ALCN}$-concept descriptions. We have shown in [BKT02a] that our

algorithm from Definition 5.1.9 can be extended to number restrictions. This extension, however, needs to allow for interactions between number restrictions and existential or value restrictions. For example, the concept description $\exists r.P \sqcap \exists r.\neg P$ implies the number restriction $(\geqslant 2\ r)$. On the other hand, the concept description $\forall r.P \sqcap (\geqslant 2\ r)$ implies the existential restriction $\exists r.P$. Moreover, a concept description of the form $\exists r.(P \sqcap Q) \sqcap \exists r.(\neg P \sqcap Q) \sqcap (\leqslant 2\ r)$ implies the value restriction $\forall r.Q$ because Q occurs in both existential restrictions which cannot be interpreted by one single successor w.r.t. the role r. Hence, all r-successors are witnesses of Q.

In the approximation algorithm shown in [BKT02a], the additional complexity of handling the above implication effects is hidden in a specific normal form for $\mathcal{ALCN}$-concept descriptions. The approximation algorithm itself is similar to the one for $\mathcal{ALC}$-$\mathcal{ALE}$-approximation introduced in Definition 5.1.9.

An in-depth examination of $\mathcal{ALEN}$-approximation of $\mathcal{ALCN}$-concept descriptions is beyond the scope of this thesis. The relevant results will be presented in the forthcoming Ph.D. thesis by Turhan.

CHAPTER 6

Conclusion

The present work provides results in three main directions which can be described in a nutshell as follows.

- *Tractable reasoning revisited*: In the 1990s, DL research has largely answered the question for practically relevant yet tractable DL formalisms in the negative. Due to novel application domains, especially the Life Sciences, and a surprising tractability result by Baader in [Baa03b], we have re-visited this question, this time looking in a new direction: general TBoxes and extensions thereof defined over the DL $\mathcal{EL}$ and extensions thereof.

  As main positive result, we have devised $\mathcal{EL}^{++}(\mathcal{D})$-CBoxes as a tractable DL formalism with optimal expressivity in the sense that every additional standard DL constructor, every extension of the TBox formalism, or every more powerful concrete domain, makes reasoning intractable, as we have proven.

- *Non-standard inferences for knowledge maintenance*: Non-standard inferences, such as the lcs, the msc, and matching, can support domain experts in maintaining DL knowledge bases in a structured and well-defined way. In order to extend their availability and promote their use, the present work extends the state of the art of non-standard inferences both w.r.t. theory and implementation.

  Our main results are implementations and performance evaluations of known matching algorithms for the DLs $\mathcal{ALE}$ and $\mathcal{ALN}$, optimal non-deterministic polynomial time algorithms for matching under acyclic side conditions in $\mathcal{ALN}$ and sublanguages, and optimal algorithms for matching w.r.t. cyclic (and hybrid) $\mathcal{EL}$-TBoxes.

- *Non-standard inferences over GCIs*: The utility of GCIs in modern DL knowledge bases and the relevance of non-standard inferences to knowledge maintenance naturally motivates the question for a tractable DL formalism in which both can be provided.

  As main result, we have proposed hybrid $\mathcal{EL}$-TBoxes as a solution to this hitherto open question: GCIs are supported, and we have devised both a tractable subsumption algorithm and optimal algorithms to compute the non-standard inferences matching, lcs, and msc w.r.t. hybrid $\mathcal{EL}$-TBoxes.

In the following, we review in more detail the results obtained in the above mentioned directions.

Tractable reasoning with GCIs

Re-visiting the question for practically relevant yet tractable DL formalisms, we have chosen general $\mathcal{EL}$-TBoxes as a starting point for three reasons: firstly, because of the well-known utility of GCIs for KR applications; secondly, because $\mathcal{EL}$ and small extensions thereof play a significant role in KR for the Life Sciences; and thirdly, because of the surprising tractability result for subsumption w.r.t. cyclic $\mathcal{EL}$-TBoxes [Baa03b]. The latter is in strong contrast to the PSPACE-complete subsumption problem for cyclic $\mathcal{FL}_0$-TBoxes, a seemingly equally inexpressive DL formalism.

As a first step, we have proven in [Bra04b] that subsumption w.r.t. general TBoxes is tractable in $\mathcal{EL}$ even if the underlying TBox formalism is extended by role hierarchies. Using similar techniques for an extended subsumption procedure, this tractability result has been notably improved to $\mathcal{EL}^{++}$-CBoxes, a novel DL formalism that extends $\mathcal{EL}$ by the bottom concept, nominals, and p-admissible concrete domains, and extends general TBoxes by RIs.[1] Furthermore, we prove that not only subsumption w.r.t. $\mathcal{EL}^{++}$-CBoxes is tractable, but also satisfiability, ABox consistency, and the instance problem in case an ABox is additionally taken into account. Hence, all relevant DL standard reasoning tasks are tractable.

In order to show optimal expressivity of our $\mathcal{EL}^{++}$-CBoxes, we have proven that subsumption becomes intractable when admitting any other standard DL constructor, any non p-admissible concrete domain, or any more expressive TBox formalism. Although the crucial boundary for this optimality argument is NP-hardness, we have studied the exact complexity of reasoning w.r.t. extensions of $\mathcal{EL}$ in further detail. Extending $\mathcal{EL}$ by disjunction or number restrictions makes subsumption w.r.t. the empty TBox co-NP-complete, while subsumption w.r.t. acyclic TBoxes becomes co-NP-hard when adding the constructor allsome. W.r.t. general TBoxes, subsumption becomes PSPACE-hard when extending $\mathcal{EL}$ by inverse roles, and EXPTIME-complete for any of the constructors atomic negation, disjunction, at-most restrictions, at-least restrictions, role negation, role union, or reflexive transitive closure of roles. Extending $\mathcal{EL}$ by non p-admissible concrete domains or extending general TBoxes by functional roles also makes subsumption EXPTIME-complete.

Together with the known EXPTIME-completeness of subsumption w.r.t. general TBoxes for $\mathcal{EL}$ extended by value restrictions [GMWK02] and the undecidability of subsumption w.r.t. cyclic $\mathcal{EL}$-TBoxes extended by arbitrary role value maps [Baa03a], this completes the picture of the complexity of reasoning in extensions of $\mathcal{EL}$.

Note that several additional complexity results have been obtained, see Section 1.6.1 for a complete overview. In particular, we have shown in Theorem 3.2.13 that subsumption in $\mathcal{FL}_0$ w.r.t. general TBoxes is EXPTIME-complete.

As shown above, $\mathcal{EL}^{++}$-CBoxes are optimal in that every extension by a standard DL constructor or TBox construct destroys tractability. As shown in Section 3.2, $\mathcal{EL}^{++}$-CBoxes are sufficient to express many constructors relevant in ontology applications, especially role hierarchies, transitive roles, right identities, disjointness constraints, the unique name assumption, and domain restrictions on roles. Furthermore, examples of practically relevant p-admissible concrete domains have been discussed in Section 3.2.3. In particular, the main requirements of a KR formalism for the Life Sciences discussed in Section 1.2 are met.

From an expressivity point of view, it might be regretted that inverse roles cannot be expressed without losing tractability. As pointed out in Section 2.2, there are common examples of definitions in cyclic TBoxes for which unintuitive models can only be ruled

[1] Note that that subsumption becomes undecidable even w.r.t. cyclic $\mathcal{EL}$-TBoxes when arbitrary role value maps are admitted [Baa03b].

out by inverse roles. Moreover, GCIs together with inverse roles can express domain restrictions on roles: it is easy to see that the GCI $\exists r^-.\top \sqsubseteq C$ restricts the range of r to the concept C.

In general, it seems that $\mathcal{EL}^{++}$-CBoxes are not only attractive for KR in the Life Sciences, but more general also in other contexts, where very large yet relatively inexpressive DL formalisms are used. One such example might be certain subdomains of the Semantic Web for which the full expressivity of OWL is not needed and where a tractable reasoning algorithm has a benefit to offer.

This naturally leads to the question of the performance of reasoning w.r.t. $\mathcal{EL}^{++}$-CBoxes in practical applications. Experience has shown that the theoretical worst-case complexity of subsumption w.r.t. the underlying DL formalism is not always a reliable measure for the performance of the corresponding DL reasoner. For instance, the subsumption problem was tractable in the DL underlying the DL system CLASSIC [BBMR89, BPS94], while subsumption is EXPTIME-complete in the DL $\mathcal{SHIQ}$ on which the reasoner FaCT [Hor98] is built. In a practical comparison on the same knowledge bases, however, FaCT on average performs by far better than CLASSIC.

In the case of $\mathcal{EL}^{++}$-CBoxes, there are not only theoretical results to draw our conclusions from, but also a prototype implementation: the reasoner Cel implements $\mathcal{EL}^{+}$-CBoxes, a fragment of $\mathcal{EL}^{++}$-CBoxes without the bottom concept and without nominals. It has been shown in [BLS05] that Cel outperforms highly optimized tableaux reasoners, such as FaCT or Racer, on several biomedical ontologies. As discussed in Section 3.4, Cel is a relatively straightforward implementation of our subsumption algorithm from Section 3.2.2. Hence, it seems highly promising to investigate further into optimization techniques for Cel, and thus, into tractable subsumption w.r.t. $\mathcal{EL}^{++}$-CBoxes.

As Cel does not yet support ABoxes, the question about the performance of ABox-reasoning w.r.t. underlying $\mathcal{EL}^{++}$-CBoxes remains open. It is already known that $\mathcal{EL}$, and thus $\mathcal{EL}^{++}$ likewise, is too expressive to hand down ABox reasoning to highly optimized Database Management Systems, an approached pursued successfully for the DL formalism DL-Lite in [CDGL$^+$05a]. Nevertheless, it should be noted that other approaches to ABox reasoning exist which performed well w.r.t. large ABoxes [HM01a].

Non-standard inferences

The considerable size of real-world DL knowledge bases together with the fact that typically entire groups of domain experts develop them over several years motivates the question of a standard methodology for knowledge maintenance and of appropriate tools to support knowledge engineers in the relevant tasks. In Section 1.4, we have argued the case for non-standard inferences as well-defined reasoning services by which the desired tools can be provided.

In the present work, we have devised sound and complete non-deterministic polynomial-time algorithms to solve matching problems under acyclic side conditions for the DL $\mathcal{ALN}$ and its sublanguages $\mathcal{FL}_\bot$ and $\mathcal{FL}_\neg$. The algorithms are optimal in terms of computational complexity because the corresponding decision problem is known to be NP-hard [BKBM99]. As a consequence, our algorithms also show NP-completeness of the corresponding decision problems. Note that, in contrast to subsumption, complexity results for matching do not automatically transfer to sublanguages or superlanguages.

Moreover, we have devised a deterministic exponential time algorithm to solve matching problems w.r.t. cyclic $\mathcal{EL}$-TBoxes with greatest-fixedpoint semantics. The algorithm is sound and complete in that all minimal matchers are computed, which is optimal since even w.r.t. the empty TBox the least matcher to an $\mathcal{EL}$-matching problem does not always

exist. As we have also shown that the solutions of matching problems w.r.t. cyclic $\mathcal{EL}$-TBoxes with greatest-fixedpoint semantics can be exponential in number and can grow exponentially large in the size of the input, the relevant algorithm for cyclic $\mathcal{EL}$-TBoxes is optimal. Moreover, we proved that our matching algorithm generalizes matching in $\mathcal{EL}$ w.r.t. the empty TBox, implying NP-hardness of deciding the solvability of matching problems modulo equivalence w.r.t. cyclic $\mathcal{EL}$-TBoxes. It remains open whether this bound is tight. In the case of matching modulo subsumption, deciding the solvability w.r.t. cyclic $\mathcal{EL}$-TBoxes with greatest-fixedpoint semantics is tractable. See also below for our matching algorithm w.r.t. hybrid $\mathcal{EL}$-TBoxes.

In addition to the above theoretical results, we have implemented matching algorithms for the DLs $\mathcal{ALE}$ and $\mathcal{ALN}$, and evaluated their performance in detail. To the best of our knowledge, these are the first implementations of independent, general-purpose matching algorithms for DLs. Our extensive tests presented in Sections 4.5.1 and 4.5.2 suggest that even the deterministic exponential space matching algorithm for $\mathcal{ALE}$ performs well on standard hardware even w.r.t. relatively large randomly generated matching problems. In the case of $\mathcal{ALN}$, the algorithm took an average of 1.2 seconds for matching problems of average size of 703. For $\mathcal{ALE}$, matching took on average 0.8 seconds with an average input size of 528. Our direct comparison on $\mathcal{FL}_\neg$-matching problems even suggests that the $\mathcal{ALE}$ matching algorithm performs better than its $\mathcal{ALN}$ counterpart with better worst-case complexity.

Besides matching, the present work has introduced the novel inference service approximation first suggested as a DL inference service in [BKM00]. We have devised a sound and complete deterministic double-exponential time algorithm computing upper $\mathcal{ALE}$-approximations of $\mathcal{ALC}$-concept descriptions. We were also able to show that the size of $\mathcal{ALE}$-approximations of $\mathcal{ALC}$-concept descriptions grows exponential in the size of the input concept description in the worst case. Hence, any $\mathcal{ALC}$-$\mathcal{ALE}$-approximation algorithm is at least worst-case exponential. Finally, in order to speed-up of the computation of approximations, we have shown for $\mathcal{ALC}$-concept descriptions of a certain structure, called nice concepts, that our approximation algorithm can be simplified significantly.

In the case of approximation, the question of the practicability of an implementation might be even more interesting because of the even higher worst-case complexity. As shown in [BKT02b], computing approximations w.r.t. a TBox from the domain of chemical process engineering on standard hardware exhibited good run-times below 3 seconds despite input concept sizes of up to 740. Although we have shown that $\mathcal{ALE}$-approximations of $\mathcal{ALC}$-concept descriptions can be exponentially large in the input, the approximations computed for our 'meaningful' concepts from a realistic application domain were *smaller* than the input: on average, the $\mathcal{ALE}$-approximation had one third the size of the original $\mathcal{ALC}$-concept.

It should be stressed that non-standard inferences support a unique approach to knowledge maintenance, with the basic understanding that ontologies with a formally well-defined semantics should be maintained by tools with an equally well-defined semantics. Moreover, non-standard inferences can be used to have new concept definitions *constructed* automatically by sound and complete algorithms. In contrast, most common contributions to methodologies for ontology development, see [CC05, JBCV98] for an overview, deal rather with the project-management aspect of knowledge maintenance, while low-level tasks, such as creating or modifying concept definitions, are typically considered to be done fully manually by domain experts.

By means of the system SONIC [Tur05], several non-standard inference services are made available to the common knowledge editor PROTÉGÉ [GMF+03]. Although matching is not supported at the moment, the modular architecture of SONIC makes it relatively easy to

add new reasoning services. Due to its recent release, the SONIC system could not yet be tested by independent domain experts.

Non-standard inferences cannot be defined properly in a DL as expressive as OWL-Lite or OWL-DL. Hence, any similar inference service for such expressive DLs will have to approximate optimal solutions or resort to heuristics. One example in this direction is the good common subsumer defined w.r.t. expressive background terminologies, see Section 1.7.2. Moreover, it has been shown that the lcs, and thus other non-standard inferences, do not always exist w.r.t. cyclic $\mathcal{EL}$-TBoxes with descriptive semantics. This result carries over to general TBoxes. However, we have shown in the present work how non-standard inferences can be provided in the presence of GCIs, see below.

Non-standard inferences over GCIs

In order to combine the advantages of general TBoxes and the utility non-standard inferences, we have introduced hybrid $\mathcal{EL}$-TBoxes as a novel TBox formalism where GCIs and non-standard inferences are available. A hybrid $\mathcal{EL}$-TBox is a pair $(\mathcal{F}, \mathcal{T})$ of a general TBox $\mathcal{F}$ ('foundation') and a possibly cyclic TBox $\mathcal{T}$ ('terminology') defined over the same set of atomic concepts and roles. The foundation $\mathcal{F}$ is interpreted by descriptive semantics while the terminology $\mathcal{T}$ is interpreted by greatest-fixedpoint semantics. We have shown that hybrid $\mathcal{EL}$-TBoxes generalize both general $\mathcal{EL}$-TBoxes and cyclic $\mathcal{EL}$-TBoxes with descriptive semantics.

For hybrid $\mathcal{EL}$-TBoxes, we have devised a tractable, sound and complete subsumption algorithm based on a polynomial reduction to cyclic $\mathcal{EL}$-TBoxes with greatest-fixedpoint semantics. For the latter formalism, a tractable subsumption algorithm has been presented in [Baa03b].

For hybrid $\mathcal{EL}$-TBoxes, we have devised a deterministic exponential time matching algorithm that is sound and complete in the sense that it computes all minimal matchers, which is optimal since the least matcher to an $\mathcal{EL}$-matching problem does not always exist. Our algorithm is also optimal w.r.t. computational complexity because we have shown that matchers of matching problems w.r.t. hybrid $\mathcal{EL}$-TBoxes with greatest-fixedpoint semantics can be exponentially large in the size of the input. The results on hybrid TBoxes are obtained by a polynomial reduction to cyclic $\mathcal{EL}$-TBoxes with greatest-fixedpoint semantics, and the above mentioned deterministic exponential time matching algorithm for the latter formalism. By means of the same reduction, and an appropriate lcs and msc algorithm from [Baa03b], we could also devise a tractable algorithm computing the binary lcs and msc w.r.t. hybrid $\mathcal{EL}$-TBoxes.

The use of complex TBox formalisms, e.g., cyclic and especially general TBoxes, has been a barrier to the use of non-standard inferences. The first step to overcome this has been taken by algorithms to compute the lcs and msc w.r.t. cyclic $\mathcal{EL}$-TBoxes with greatest-fixedpoint semantics [Baa03a]. By means of hybrid TBoxes proposed in the present work, the lcs, msc, and matching can now also be provided for GCIs. In particular, as long as matching is used only as a query mechanism, i.e., only the existence of matchers is interesting, our matching algorithm can be applied to standard general $\mathcal{EL}$-TBoxes without any changes.

A natural question to discuss is whether hybrid $\mathcal{EL}$-TBoxes can be extended without sacrificing tractability of subsumption and the availability of non-standard inferences. Concerning the foundation part of a hybrid TBox, the limit of expressivity is clearly defined by $\mathcal{EL}^{++}$-CBoxes, as discussed in depth above. It is open, however, whether cyclic $\mathcal{EL}^{++}$-TBoxes with RIs interpreted with greatest-fixedpoint semantics can be classified in polynomial time and whether non-standard inferences exist for them.

Nevertheless, an interesting starting point might be to extend hybrid TBoxes by RIs be-

cause it has already been shown in [Baa03b] that cyclic $\mathcal{EL}$-TBoxes with greatest-fixedpoint semantics extended by RIs can be classified in polynomial time. The same obviously holds for general $\mathcal{EL}$-TBoxes with RIs, a sub-formalism of $\mathcal{EL}^{++}$-CBoxes. Hence, it remains to extend our polynomial reduction by RIs and to devise appropriate lcs and matching algorithms for cyclic $\mathcal{EL}$-TBoxes extended by RIs.

Bibliography

[ACDG+05] A. Acciarri, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. Quonto: Querying ontologies. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, pages 1670–1671, 2005. → pp. 31, 32, 91

[All83] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, 1983. → p. 46

[Baa90] F. Baader. Terminological cycles in KL-ONE-based knowledge representation languages. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI-90)*, pages 621–626, Boston, USA, 1990. → p. 80

[Baa96] F. Baader. Using automata theory for characterizing the semantics of terminological cycles. *Annals of Mathematics and Artificial Intelligence*, 18(2–4):175–219, 1996. → pp. 18, 80

[Baa03a] F. Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In G. Gottlob and T. Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 319–324. Morgan-Kaufmann Publishers, 2003. → pp. 20, 24, 27, 117, 118, 119, 126, 129, 131, 152, 155

[Baa03b] F. Baader. Terminological cycles in a description logic with existential restrictions. In G. Gottlob and T. Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 325–330. Morgan-Kaufmann Publishers, 2003. → pp. 18, 21, 23, 26, 40, 42, 43, 59, 80, 82, 83, 84, 85, 151, 152, 155, 156

[BBB+98] P. G. Baker, A. Brass, S. Bechhofer, C. Goble, N. Paton, and R. Stevens. TAMBIS: Transparent access to multiple bioinformatics information sources. In J. Glasgow, T. Littlejohn, F. Major, R. Lathrop, D. Sankoff, and C. Sensen, editors, *6th International Conference on Intelligent Systems for Molecular Biology*, pages 25–34, Montreal, Canada, 1998. AAAI Press. → p. 17

[BBK01] F. Baader, S. Brandt, and R. Küsters. Matching under side conditions in description logics. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 213–218, Seattle, Washington, 2001. Morgan-Kaufmann Publishers. → pp. 19, 25, 26, 100, 101, 103, 104, 105

[BBL05] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005)*, Edinburgh, UK, 2005. Morgan-Kaufmann Publishers. → pp. 23, 24, 30, 80

[BBMR89] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. CLASSIC: A Structural Data Model for Objects. In J. Clifford, B. G. Lindsay, and D. Maier, editors, *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, Portland, Oregon*, pages 58–67. ACM Press, 1989. → pp. 18, 153

[BDS93] M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993. → p. 18

[BER+98] J. A. Blake, J. T. Eppig, J. E. Richardson, M. T. Davisson, and The Mouse Genome Informatics Group. The Mouse Genome Database (MGD): a community resource. Status and enhancements. *Nucleic Acids Research*, 26(1):130–137, 1998. → p. 7

[BFH+99] A. Borgida, E. Franconi, I. Horrocks, D. L. McGuinness, and P. F. Patel-Schneider. Explaining ALC subsumption. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Proceedings of the International Workshop on Description Logics (DL'99)*, number 22 in CEUR-WS, Sweden, 1999. Linköping University. Proceedings online available from http://CEUR-WS.org/Vol-22/. → p. 34

[BFH00] A. Borgida, E. Franconi, and I. Horrocks. Explaining ALC subsumption. In W. Horn, editor, *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2000)*, pages 209–213, Berlin, Germany, 2000. IOS Press. → p. 34

[BH91] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 452–457, Sydney, AU, 1991. → p. 45

[BH95] F. Baader and B. Hollunder. Embedding defaults into terminological representation systems. *Journal of Automated Reasoning*, 14:149–180, 1995. → p. 34

[BHGS01] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: A reason-able ontology editor for the semantic Web. *Lecture Notes in Computer Science*, 2174:396–??, 2001. → pp. 17, 33

[BK98] F. Baader and R. Küsters. Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions. In O. Herzog and A. Günter, editors, *Proceedings of the 22nd Annual German Conference on Artificial Intelligence (KI-98)*, volume 1504 of *Lecture Notes in Computer Science*, pages 129–140, Bremen, Germany, 1998. Springer-Verlag. → p. 20

[BK99] F. Baader and R. Küsters. Matching in description logics with existential restrictions. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Proceedings of the International Workshop on Description Logics (DL'99)*, number 22 in CEUR-WS, Sweden, 1999. Linköping

University. Proceedings online available from http://CEUR-WS.org/Vol-22/. → pp. 25, 96

[BK00a] F. Baader and R. Küsters. Matching in description logics with existential restrictions. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *Proceedings of the Seventh International Conference on Knowledge Representation and Reasoning (KR2000)*, pages 261–272, Breckenridge, CO, 2000. Morgan-Kaufmann Publishers. → pp. 19, 26, 95, 96, 97, 126

[BK00b] A. Borgida and R. Küsters. What's not in a name: Some Properties of a Purely Structural Approach to Integrating Large DL Knowledge Bases. In F. Baader and U. Sattler, editors, *Proceedings of the 2000 International Workshop on Description Logics (DL2000)*, number 33 in CEUR-WS, Aachen, Germany, 2000. RWTH Aachen. Proceedings online available from http://CEUR-WS.org/Vol-33/. → p. 16

[BK01] F. Baader and R. Küsters. Unification in a description logic with transitive closure of roles. In R. Nieuwenhuis and A. Voronkov, editors, *Proceedings of the 8th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2001)*, volume 2250 of *Lecture Notes in Computer Science*, pages 217–232, Havana, Cuba, 2001. Springer-Verlag. → p. 31

[BK02] F. Baader and R. Küsters. Unification in a description logic with inconsistency and transitive closure of roles. In I. Horrocks and S. Tessaris, editors, *Proceedings of the 2002 International Workshop on Description Logics (DL2002)*, Toulouse, France, 2002. Proceedings online available from http://CEUR-WS.org/Vol-53/. → p. 31

[BKBM99] F. Baader, R. Küsters, A. Borgida, and D. McGuinness. Matching in description logics. *Journal of Logic and Computation*, 9(3):411–447, 1999. → pp. 19, 25, 26, 55, 94, 98, 99, 101, 102, 103, 116, 153

[BKM98] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. LTCS-Report LTCS-98-09, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 1998. See http://lat.inf.tu-dresden.de/research/reports.html. → p. 95

[BKM99] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In T. Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 96–101. Morgan-Kaufmann Publishers, 1999. → pp. 15, 20, 80, 95, 96, 146, 147

[BKM00] F. Baader, R. Küsters, and R. Molitor. Rewriting concepts using terminologies. In A.G. Cohn, F. Giunchiglia, and B. Selman, editors, *Proceedings of the Seventh International Conference on Knowledge Representation and Reasoning (KR2000)*, pages 297–308, San Francisco, CA, 2000. Morgan-Kaufmann Publishers. → pp. 21, 154

[BKT02a] S. Brandt, R. Küsters, and A.-Y. Turhan. Approximating $\mathcal{ALCN}$-concept descriptions. In *Proceedings of the 2002 International Workshop on Description Logics (DL2002)*, CEUR-WS, 2002. Proceedings online available from http://CEUR-WS.org/Vol-53/. → pp. 28, 29, 148, 149

[BKT02b] S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. In D. Fensel, F. Giunchiglia, D. McGuiness, and M.-A. Williams, editors, *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*, pages 203–214, San Francisco, CA, 2002. Morgan-Kaufmann Publishers. → pp. 17, 28, 29, 154

[BL84] R. J. Brachman and H. J. Levesque. The tractability of subsumption in frame-based description languages. In *Proceedings of the 4th National Conference on Artificial Intelligence (AAAI-84)*, pages 34–37, Austin, TX, 1984. → p. 80

[BL85] R. J. Brachman and H. J. Levesque, editors. *Readings in Knowledge Representation*. Morgan-Kaufmann Publishers, Los Altos (CA), USA, 1985. → p. 1

[BL04] S. Brandt and H. Liu. Implementing matching in $\mathcal{ALN}$. In *Proceedings of the KI-2004 Workshop on Applications of Description Logics (KI-ADL'04)*, CEUR-WS, Ulm, Germany, September 2004. → p. 26

[BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001. → p. 10

[BLS05] F. Baader, C. Lutz, and B. Suntisrivaraporn. Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice? In *Proceedings of the 4th International Workshop on Methods for Modalities (M4M-05)*, Berlin, Germany, 2005. → pp. 7, 29, 89, 90, 153

[BM96] A. Borgida and D. L. McGuinness. Asking queries about frames. In L. C. Aiello, J. Doyle, and S. Shapiro, editors, *KR'96: Principles of Knowledge Representation and Reasoning*, pages 340–349. Morgan-Kaufmann Publishers, San Francisco, California, 1996. → p. 19

[BM05] S. Brandt and J. Model. Subsumption in $\mathcal{EL}$ w.r.t. hybrid TBoxes. In *Proceedings of the 28th Annual German Conference on Artificial Intelligence (KI 2005)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2005. → p. 24

[BMC03] S. Bechhofer, R. Möller, and P. Crowther. The DIG description logic interface. In *Proceedings of the 2003 International Workshop on Description Logics (DL2003)*, CEUR-WS, 2003. Proceedings online available from http://CEUR-WS.org/Vol-81/. → p. 2

[BMPS+91] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick, and A. Borgida. Living with CLASSIC: When and how to use a KL-ONE language. In J. F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 401–456. Morgan-Kaufmann Publishers, San Mateo (CA), USA, 1991. → p. 18

[BN98] F. Baader and P. Narendran. Unification of concept terms in description logics. In H. Prade, editor, *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI'98)*, pages 331–335. John Wiley & Sons Ltd, 1998. → pp. 19, 54, 98

[BN01] F. Baader and P. Narendran. Unification of concepts terms in description logics. *Journal of Symbolic Computation*, 31(3):277–305, 2001. → p. 31

[BN03] F. Baader and W. Nutt. Basic description logics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 43–95. Cambridge University Press, 2003. → pp. 17, 41

[BPS94] A. Borgida and P. Patel-Schneider. A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994. → p. 153

[Bra91] R. J. Brachman. Interests and issues in description (terminological) logics. In *Proceedings of the 1991 International Workshop on Terminological Logics*, pages 28–32, 1991. → p. 2

[Bra00] S. Brandt. Matching under side conditions in description logics. Diploma thesis, RWTH Aachen, Germany, 2000. → pp. 19, 100, 101, 102, 104

[Bra03] S. Brandt. Implementing matching in $\mathcal{ALE}$—first results. In *Proceedings of the 2003 International Workshop on Description Logics (DL2003)*, CEUR-WS, 2003. Proceedings online available from http://CEUR-WS.org/Vol-81/. → p. 26

[Bra04a] S. Brandt. On subsumption and instance problem in $\mathcal{ELH}$ w.r.t. general TBoxes. In *Proceedings of the 2004 International Workshop on Description Logics (DL2004)*, CEUR-WS, 2004. Proceedings online available from http://CEUR-WS.org/Vol-104/. → pp. 23, 24

[Bra04b] S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In R. López de Mantáras and L. Saitta, editors, *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, pages 298–302. IOS Press, 2004. → pp. 22, 23, 24, 60, 80, 89, 152

[BS85] R. J. Brachman and J. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985. → p. 2

[BS03] A. Borgida and L. Serafini. Distributed description logics: Assimilating information from peer sources. *Journal of Data Semantics*, 1:153–184, 2003. → p. 32

[BST04] F. Baader, B. Sertkaya, and A.-Y. Turhan. Computing the least common subsumer w.r.t. a background terminology. In J. J. Alferes and J. A. Leite, editors, *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA 2004)*, volume 3229 of *Lecture Notes in Computer Science*, pages 400–412, Lisbon, Portugal, 2004. Springer-Verlag. → p. 32

[BT01] S. Brandt and A.-Y. Turhan. Using non-standard inferences in description logics — what does it buy me? In *Proceedings of the KI-2001 Workshop on Applications of Description Logics (KIDLWS'01)*, number 44 in CEUR-WS, Vienna, Austria, September 2001. RWTH Aachen. Proceedings online available from http://CEUR-WS.org/Vol-44/. → pp. 13, 15, 26, 27

[BT02a] F. Baader and A.-Y. Turhan. On the problem of computing small representations of least common subsumers. In *Proceedings of the 25th Annual German Conference on Artificial Intelligence (KI 2002)*, Lecture Notes in Artificial Intelligence, Aachen, Germany, 2002. Springer-Verlag. → p. 132

[BT02b] S. Brandt and A.-Y. Turhan. An approach for optimized approximation. In *Proceedings of the KI-2002 Workshop on Applications of Description Logics (ADL 2002)*, CEUR-WS, Aachen, Germany, September 2002. RWTH Aachen. Proceedings online available from http://CEUR-WS.org/Vol-63/. → pp. 28, 29, 148

[BTK03] S. Brandt, A.-Y. Turhan, and R. Küsters. Extensions of non-standard inferences to description logics with transitive roles. In M. Vardi and A. Voronkov, editors, *Proceedings of the 10th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2003)*, Lecture Notes in Computer Science. Springer-Verlag, 2003. → p. 27

[CAB+98] J. M. Cherry, C. Adler, C. Ball, S. A. Chervitz, S. S. Dwight, E. T. Hester, Y. Jia, G. Juvik, T. Y. Roe, M. Schroeder, S. Weng, and D. Botstein. SGD: Saccharomyces genome database. *Nucleic Acids Research*, 26(1):73–79, 1998. → p. 7

[Cal96] D. Calvanese. Reasoning with inclusion axioms in description logics: algorithms and complexity. In W. Wahlster, editor, *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI'96)*, pages 303–307. John Wiley & Sons Ltd, 1996. → pp. 18, 61

[CBH92] W. W. Cohen, A. Borgida, and H. Hirsh. Computing least common subsumers in description logics. In W. Swartout, editor, *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92)*, pages 754–760, San Jose, California, July 1992. The MIT Press. → p. 19

[CC05] M. Cristani and R. Cuel. A survey on ontology creation methodologies. *International Journal on Semantic Web Information Systems*, 1(2):49–69, 2005. → p. 154

[CDGL+98] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Information integration: Conceptual modeling and reasoning support. In *Proceedings of the 6th International Conference on Cooperative Information Systems (CoopIS'98)*, pages 280–291, 1998. → p. 32

[CDGL+01] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Data integration in data warehousing. *International Journal of Cooperative Information Systems*, 10(3):237–271, 2001. → p. 32

[CDGL02] D. Calvanese, G. De Giacomo, and M. Lenzerini. A framework for ontology integration. In I. Cruz, S. Decker, J. Euzenat, and D. McGuinness, editors, *The Emerging Semantic Web—Selected Papers from the First Semantic Web Working Symposium*, pages 201–214. IOS Press, 2002. → p. 32

[CDGL+05a] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005. → pp. 30, 31, 90, 91, 153

[CDGL+05b] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Efficiently managing data intensive ontologies. In P. Bouquet and G. Tummarello, editors, *Proceedings of the 2nd Italian Semantic Web Workshop (SWAP 2005)*, CEUR-WS, 2005. Proceedings online available from http://CEUR-WS.org/Vol-166/. → p. 92

[CH94a] W. W. Cohen and H. Hirsh. The learnability of description logics with equality constraints. *Machine Learning*, 17(2/3):169–199, 1994. Special Issue for COLT '92. → p. 19

[CH94b] W. W. Cohen and H. Hirsh. Learning the CLASSIC description logic: Theoretical and experimental results. In J. Doyle, E. Sandewall, and P. Torasso, editors, *KR'94: Principles of Knowledge Representation and Reasoning*, pages 121–133. Morgan-Kaufmann Publishers, San Francisco, California, 1994. → p. 19

[Chu00] C. G. Chute. Clinical classification and terminology: some history and current observations. *Journal of the American Medical Informatics Association: JAMIA*, 3(7):298–303, 2000. → p. 6

[Con98] The FlyBase Consortium. FlyBase: a drosophila database. *Nucleic Acids Research*, 26(1):85–88, 1998. → p. 7

[Con00] The Gene Ontology Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000. → p. 7

[CRP+93] R. Côté, D. Rothwell, J. Palotay, R. Beckett, and L. Brochu. The systematized nomenclature of human and veterinary medicine. Technical report, SNOMED International, Northfield, IL, 1993. → p. 6

[DG84] W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, 1(3):267–284, 1984. → p. 90

[DGL94] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, pages 205–212, Seattle, US, 1994. → pp. 61, 63

[DLN+92] F. M. Donini, M. Lenzerini, D. Nardi, B. Hollunder, W. Nutt, and A. Spaccamela. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 53(2–3):309–327, 1992. → p. 50

[DLNN95] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. Research Report RR-95-07, DFKI, Saarbrücken Universität, Germany, 1995. → p. 51

[Don03] F. M. Donini. Complexity of reasoning. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 96–136. Cambridge University Press, 2003. → p. 18

[FHH04] R. Fikes, P. J. Hayes, and I. Horrocks. OWL-QL – a language for deductive query answering on the semantic web. *Journal of Web Semantics*, 2(1), 2004. → p. 32

[FL79] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979. → p. 66

[FP96] Michael Frazier and Leonard Pitt. CLASSIC learning. *Machine Learning*, 25:151–193, 1996. → p. 20

[FvHH+01] D. Fensel, F. van Harmelen, I. Horrocks, D. L. McGuinness, and P. F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, 2001. → p. 11

[Gan99] B. Ganter. Attribute exploration with background knowledge. *Theoretical Computer Science*, 217(2):215–233, 1999. → p. 32

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, New York, 1979. → pp. 50, 51, 58

[GKM05] J. Galinski, A. Kaya, and R. Möller. Development of a server to support the formal semantic web query language owl-ql. In I. Horrocks, U. Sattler, and F. Wolter, editors, *Proceedings of the 2005 International Workshop on Description Logics (DL2005)*, number 147 in CEUR-WS, 2005. Proceedings online available from http://CEUR-WS.org/Vol-147/. → p. 32

[GMF+03] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubezy, H. Eriksson, N. F. Noy, and S. W. Tu. The evolution of protege: an environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, 58(1):89–123, 2003. → pp. 2, 17, 33, 135, 154

[GMWK02] R. Givan, D. McAllester, C. Witty, and D. Kozen. Tarskian set constraints. *Information and Computation*, 174(2):105–131, 2002. → pp. 18, 22, 59, 152

[GN87] Michael Genesereth and Nils Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan-Kaufmann Publishers, San Mateo (CA), USA, 1987. → p. 5

[GP04] B. C. Grau and B. Parsia. From SHOQ(D) toward E-connections. In *Proceedings of the 2004 International Workshop on Description Logics (DL2004)*, CEUR-WS, 2004. Proceedings online available from http://CEUR-WS.org/Vol-104/. → p. 32

[Gru93a] T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers. → p. 5

[Gru93b] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993. → p. 5

[GW99] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, 1999. → p. 32

[Hay79] P. J. Hayes. The logic of frames. In D. Metzing, editor, *Frame Conceptions and Text Understanding*. Walter de Gruyter and Co., Berlin, 1979. → p. 1

[HB91] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In J. F. Allen, R. Fikes, and E. Sandewall, editors, *KR'91: Principles of Knowledge Representation and Reasoning*, pages 335–346. Morgan-Kaufmann Publishers, San Mateo (CA), USA, 1991. → p. 2

[Hen01] J. Hendler. Agents and the semantic web. *IEEE Intelligent Systems*, 16(2):30–37, 2001. → p. 11

[HHL99] J. Heflin, J. Hendler, and S. Luke. SHOE: A knowledge representation language for internet applications. Technical Report CS-TR-4078, University of Maryland, College Park, 1999. → p. 10

[HM01a] V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 161–168, Seattle, Washington, 2001. Morgan-Kaufmann Publishers. → p. 153

[HM01b] V. Haarslev and R. Möller. RACER system description. *Lecture Notes in Computer Science*, 2083:701–712, 2001. → pp. 2, 132

[Hof05] M. Hofmann. Proof-theoretic approach to description-logic. In P. Panangaden, editor, *Proceedings of the Twentieth Annual IEEE Symposium on Logic in Computer Science, LICS 2005*, pages 229–237. IEEE Computer Society Press, June 2005. → pp. 30, 81

[Hor98] I. R. Horrocks. Using an expressive description logic: FaCT or fiction? In A. G. Cohn, L. Schubert, and S. C. Shapiro, editors, *KR'98: Principles of Knowledge Representation and Reasoning*, pages 636–645. Morgan-Kaufmann Publishers, San Francisco, California, 1998. → pp. 2, 132, 153

[HPSvH03] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003. → p. 11

[HRG96] I. Horrocks, A. L. Rector, and C. A. Goble. A description logic based schema for the classification of medical data. In *Knowledge Representation Meets Databases. Proceedings of the 3rd Workshop KRDB'96*, 1996. → pp. 5, 7, 8

[HS03] J. Hladik and U. Sattler. A translation of looping alternating automata to description logics. In *Proceedings of the 19th Conference on Automated Deduction (CADE-19)*, volume 2741 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2003. → p. 60

[HS05] I. Horrocks and U. Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005)*, Edinburgh, UK, 2005. Morgan-Kaufmann Publishers. → p. 18

[HST99] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, September 1999. → p. 18

[HT02] I. Horrocks and S. Tessaris. Querying the semantic web: a formal approach. In *Proceedings of the International Semantic Web Conference*, pages 177–191, Chia, Sardinia, Italy, 2002. → p. 32

[JBCV98] D. Jones, T. Bench-Capon, and P. Visser. Methodologies for ontology development. In *Proceedings of the IT&KNOWS Conference of the 15th IFIP World Computer Congress*. Chapman-Hall, 1998. → p. 154

[Kaz05] Y. Kazakov. *Saturation-Based Decision Procedures for Extensions of the Guarded Fragment*. Ph.D. dissertation, Naturwissenschaftlich-Technische Fakultät I, Universität des Saarlandes, Germany, 2005. → p. 30

[KB01] R. Küsters and A. Borgida. What's in an attribute? Consequences for the least common subsumer. *Journal of Artificial Intelligence Research*, 14:167–203, 2001. → p. 20

[KLWZ04] O. Kutz, C. Lutz, F. Wolter, and M. Zakharyaschev. E-connections of abstract description systems. *Artificial Intelligence*, 156(1):1–73, 2004. → p. 32

[KM01a] R. Küsters and R. Molitor. Approximating Most Specific Concepts in Description Logics with Existential Restrictions. In F. Baader, editor, *Proceedings of the Joint German/Austrian Conference on Artificial Intelligence, 24th German / 9th Austrian Conference on Artificial Intelligence (KI 2001)*, volume 2174 of *Lecture Notes in Artificial Intelligence*, Vienna, Austria, 2001. Springer–Verlag. → p. 20

[KM01b] R. Küsters and R. Molitor. Computing Least Common Subsumers in ALEN. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 219–224. Morgan-Kaufmann Publishers, 2001. → p. 20

[KN03] Y. Kazakov and H. De Nivelle. Subsumption of concepts in $\mathcal{FL}_0$ for (cyclic) terminologies with respect to descriptive semantics is PSPACE-complete. In *Proceedings of the 2003 International Workshop on Description Logics (DL2003)*, CEUR-WS, 2003. Proceedings online available from http://CEUR-WS.org/Vol-81/. → pp. 18, 22, 80

[Küs01] R. Küsters. *Non-Standard Inferences in Description Logics*, volume 2100 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2001. Ph.D. dissertation. → pp. 16, 18, 19, 20, 28, 38, 41, 141

[LDLT02] C. Le Duc and N. Le Thanh. Approximation from a description logic with disjunction to another without disjunction. In *Proceedings of the KI-2002 Workshop on Applications of Description Logics (ADL 2002)*, CEUR-WS, Aachen, Germany, September 2002. RWTH Aachen. Proceedings online available from http://CEUR-WS.org/Vol-63/. → pp. 35, 147

[LDLT03] C. Le Duc and N. Le Thanh. On the problems of representing least common subsumer and computing approximation in DLs. In *Proceedings of the 2003 International Workshop on Description Logics (DL2003)*, CEUR-WS, 2003. → pp. 35, 147

[Liu05] H. Liu. Matching in description logics with existential restrictions and terminological cycles. Master's thesis, Dresden University of Technology, Germany, 2005. → p. 26

[LS00] C. Lutz and U. Sattler. The complexity of reasoning with boolean modal logic. In *Advances in Modal Logic 2000 (AiML 2000)*, Leipzig, Germany, 2000. Final version appeared in *Advances in Modal Logic Volume 3*, 2001. → p. 66

[Lut01] C. Lutz. NExpTime-complete description logics with concrete domains. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proceedings of the International Joint Conference on Automated Reasoning*, number 2083 in Lecture Notes in Artifical Intelligence, pages 45–60, Siena, Italy, 2001. Springer-Verlag. → p. 78

[Lut02] C. Lutz. *The Complexity of Reasoning with Concrete Domains*. Ph.D. dissertation, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2002. → p. 64

[Lut03] C. Lutz. Description logics with concrete domains—a survey. In *Advances in Modal Logics Volume 4*. World Scientific Publishing Co. Pte. Ltd., 2003. → pp. 45, 46, 76

[MB95] D. McGuinness and A. Borgida. Explaining subsumption in description logics. In C. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 816–821, San Mateo (CA), USA, 1995. Morgan-Kaufmann Publishers. → p. 34

[McG96] D. McGuinness. *Explaining Reasoning in Description Logics*. Ph.D. dissertation, Department of Computer Science, Rutgers University, New Brunswick, New Jersey, 1996. → pp. 19, 34

[Min81] M. Minsky. A framework for representing knowledge. In J. Haugeland, editor, *Mind Design*, pages 95–128. The MIT Press, Cambridge (MA), USA, 1981. → p. 1

[MM03] S. A. McIlraith and D. L. Martin. Bringing semantics to web services. *IEEE Intelligent Systems*, 18(1):90–93, 2003. → p. 11

[Mol00] R. Molitor. *Unterstützung der Modellierung verfahrenstechnischer Prozesse durch Nicht-Standardinferenzen in Beschreibungslogiken*. Ph.D. dissertation, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2000. → p. 20

[MRI95] D. L. McGuinness, L. A. Resnick, and C. Isbell. Description logic in practice: A CLASSIC application. In C. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 2045–2046, San Mateo (CA), USA, 1995. Morgan-Kaufmann Publishers. → p. 19

[MSZ01] S. A. McIlraith, T. Cao Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, 2001. → p. 11

[MY60] R. McNaughton and H. Yamada. Regular expressions and state graphs for automata. *IEEE Transactions on Electronic Computers*, 9(1):39–47, 1960. → p. 58

[NB03] D. Nardi and R. J. Brachmann. An introduction to description logics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 1–40. Cambridge University Press, 2003. → p. 17

[Neb90] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990. → pp. 18, 41, 54, 55, 57, 80, 98

[Neb91] B. Nebel. Terminological cycles: Semantics and computational properties. In J. F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 331–361. Morgan-Kaufmann Publishers, San Mateo (CA), USA, 1991. → pp. 4, 40, 43, 44

[PSK05] B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL ontologies. In *Proceedings of the 14th International Conference on the World Wide Web WWW'05*, pages 633–640, New York, NY, USA, 2005. ACM Press. → p. 35

[Qui67] M. R. Quillian. Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioral Science*, 12:410–430, 1967. → p. 1

[RBG+97] A. Rector, S. Bechhofer, C. A. Goble, I. Horrocks, W. A. Nowlan, and W. D. Solomon. The GRAIL concept modelling language for medical terminology. *Artificial Intelligence in Medicine*, 9:139–171, 1997. → p. 8

[RCC92] D. A. Randell, Z. Cui, and A. G. Cohn. A spatial logic based on regions and connection. In B. Nebel, C. Rich, and W. Swartout, editors, *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR-03)*, pages 165–176, Cambridge, MA, October 1992. Morgan-Kaufmann Publishers. → p. 46

[Rec03] A. Rector. Medical informatics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 406–426. Cambridge University Press, 2003. → p. 5

[RG98] K. J. Rothman and S. Greenland. *Modern Epidemiology*. Lippincott-Raven, Philadelphia (PA), USA, 1998. → p. 6

[RH97] A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI-97)*. AAAI Press, 1997. → pp. 7, 9

[RNG93] A. Rector, W. Nowlan, and A. Glowinski. Goals for concept representation in the GALEN project. In *Proceedings of the 17th annual Symposium on Computer Applications in Medical Care, Washington, USA, SCAMC*, pages 414–418, 1993. → p. 5

[SC03] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003)*. Morgan-Kaufmann Publishers, 2003. → pp. 34, 35

[Sch91] K. Schild. A correspondence theory for terminological logics: preliminary report. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 466–471, Sydney, AU, 1991. → pp. 60, 66

[Sch92] A. Schaerf. On the role of subsumption algorithms in concept description languages. In *Proceedings of the 1992 Workshop on Issues in Description Logics: Users Meet Developers*, pages 86–97, 1992. → p. 2

[SH02] S. Schulz and U. Hahn. Necessary parts and wholes in bio-ontologies. In D. Fensel, F. Giunchiglia, D. McGuinness, and M.-A. Williams, editors, *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR-02)*, pages 387–394, San Francisco, CA, 2002. Morgan-Kaufmann Publishers. → p. 8

[Shi01] G. A. Shif. DAML: The foundation of an intelligent Web. *IEEE Distributed Systems Online*, 2(1), 2001. → p. 11

[SL83] J. G. Schmolze and T. A. Lipkis. Classification in the KL-ONE representation system. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, Karlsruhe, Germany, August 1983. Morgan-Kaufmann Publishers. → p. 20

[SP04] E. Sirin and B. Parsia. Pellet: An OWL DL reasoner. In *Proceedings of the 2004 International Workshop on Description Logics (DL2004)*, CEUR-WS, 2004. Proceedings online available from http://CEUR-WS.org/Vol-104/. → p. 2

[Spa00] K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED RT. *Journal of the American Medical Informatics Association*, (Fall Symposium Special Issue), 2000. → p. 9

[Spa01] K. Spackman. Normal forms for description logic expressions of clinical concepts in SNOMED RT. *Journal of the American Medical Informatics Association*, (Symposium Supplement), 2001. → p. 7

[SS89] M. Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In R. J. Brachman, H. J. Levesque, and R. Reiter, editors, *KR'89: Principles of Knowledge Representation and Reasoning*, pages 421–431. Morgan-Kaufmann Publishers, San Mateo (CA), USA, 1989. → p. 2

[SSS91] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991. → p. 50

[ST04] L. Serafini and A. Tamilin. Local tableaux for reasoning in distributed description logics. In *Proceedings of the 2004 International Workshop on Description Logics (DL2004)*, CEUR-WS, 2004. Proceedings online available from http://CEUR-WS.org/Vol-104/. → p. 32

[Tar55] A. Tarski. A lattice-theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955. → p. 42

[TK04] A.-Y. Turhan and C. Kissig. SONIC—non-standard inferences go OILED. In D. Basin and M. Rusinowitch, editors, *Proceedings of the 2nd International Joint Conference on Automated Reasoning (IJCAR'04)*, volume 3097 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2004. → pp. 33, 135

[Tob00] S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research*, 12:199–217, May 2000. → p. 18

[Tur05] A.-Y. Turhan. Pushing the SONIC border—SONIC 1.0. In R. Letz, editor, *Fifth International Workshop on First-Order Theorem Proving (FTP 2005)*. Technical Report, University of Koblenz, 2005. → pp. 33, 135, 154

[TW05] D. Toman and G. Weddell. On reasoning about structural equality in XML: a description logic approach. *Theoretical Computer Science*, 336(1):181–203, 2005. → p. 62

[WM05] M. Wessel and R. Möller. A high performance semantic web query answering engine. In I. Horrocks, U. Sattler, and F. Wolter, editors, *Proceedings of the 2005 International Workshop on Description Logics (DL2005)*, number 147 in CEUR-WS, 2005. Proceedings online available from http://CEUR-WS.org/Vol-147/. → p. 32