Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies

DISSERTATION

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

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von

Dipl.-Inf. Boontawee Suntisrivaraporn
geboren am 18. Februar 1980 in Chiang Rai, Thailand

Gutachter: Prof. Dr.-Ing. Franz Baader
Technische Universität Dresden;
Prof. Ian Horrocks
University of Oxford;
Prof. Dr. rer. nat. habil. Ralf Möller
Technische Universität Hamburg-Harburg

Tag der Verteidigung: 21. Januar 2009

Dresden, im Februar 2009

Contents

List of Algorithms

List of Figures

List of Tables

Chapter 1

Introduction

Knowledge representation (KR) is an important subject in artificial intelligence and cognitive science. Generally speaking, it is the approach to store *explicit* knowledge about a particular domain so that computers are able to process and use it, and above all to infer *implicit* knowledge from the one explicitly given, hence knowledge representation and reasoning. Description Logics belong to a successful family of KR formalisms, allowing to represent and reason with conceptual knowledge about a domain of discourse.

This thesis proposes several reasoning techniques for a tractable Description Logic that form a core of automated reasoning support for design and maintenance of ontologies, and investigates their usefulness and usability in realistic ontology applications from the biomedical domain. It suggests that the use of Description Logics and their reasoning support in ontology development helps to increase the clarity and coherence which are important criteria for a high-quality ontology. Moreover, it suggests that, for the specific application area of biomedical ontologies, the tractable Description Logic is sufficient in terms of expressivity and is robustly scalable in terms of automated reasoning.
The claim has been supported by:

- the design and implementation of an optimized algorithm for classifying an ontology formulated in the tractable Description Logic;

- the design and implementation of supplemental reasoning techniques useful for design and maintenance of ontologies, namely, extracting modules from an ontology and explaining a given consequence;

- the development of a modeling paradigm for the renown medical ontology SNOMED CT using the tractable Description Logic; and,

- the empirical evaluation of the implemented reasoning techniques through extensive and systematic experiments on large-scale biomedical ontologies, including an overview comparison with other state-of-the-art systems.

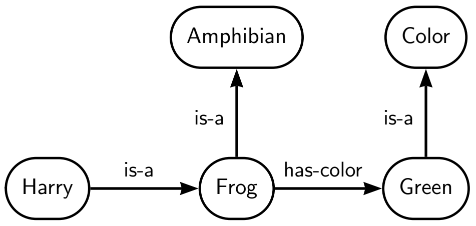


Figure 1.1: An example of a semantic network.

1.1 Knowledge Representation with Description Logics

Description Logics (DLs) have evolved from, thus greatly been influenced by, the early KR formalisms of *semantic networks* [Qui67] and *frames* [Min81]. Both predecessors of DLs share the notions of classes of individuals and relations between such classes.

These notions are realized in semantic networks as *vertices* and *edges* in a labeled directed graph. Vertices represent either *individuals* or *classes* of individuals (also called *concepts*), and labeled edges represent *relations* between them. A special type of relation, called is-a, is used in semantic networks to specify the generality or specificity of classes. It is this relationship that provides the basis for the inheritance of properties and that defines a hierarchy over concepts [Bra79]. Other kinds of relationships are realized as edges with other labels, e.g., has-color in Figure 1.1.

In the case of frame systems, concepts are realized as *frames* similar to the notion of classes in object-oriented programming languages. Each frame has a name, a collection of more general frames and a collection of *slots*. Slots are used to specify properties of concepts by linking the current frame to others in a similar sense as edges in semantic networks.

The main problem with both semantic networks and frame systems is that they lacked a formally well-defined semantics. For instance, it is unclear what the edge has-color in Figure 1.1 is intended to mean. One possible reading is that "frogs may only have color green," while another is that "frogs have at least a color green." Yet, the edge may be understood as a default property of frogs that can be overridden later when more knowledge is specified. Moreover, allowing vertices to represent both individuals and classes of individuals is ambiguous (e.g., Harry is intended to be an individual frog as opposed to Frog and Amphibian). Having to rely on their own *operational semantics*, different reasoning algorithms for the same formalism could behave differently upon the same knowledge base. To overcome this problem, *declarative semantics* had to be defined formally and independently of any specific reasoning algorithms. Attempts [Hay79, BL85] to employ relatively small fragments of first-order logics in these early KR systems have eventually resulted in 'logic-based concept languages' which have later become known as Description Logics.

The quest for tractable Description Logics

The quest for tractable (i.e., polynomial-time decidable) Description Logics started in the 1980s after the first intractability results for DLs were shown [BL84, Neb88]. Until relatively recently, it was restricted to DLs that extend the basic language $\mathcal{FL}_0$, which comprises the concept constructors conjunction and universal quantification.[1] The main reason for this focusing was that, when clarifying the logical status of property edges in semantic networks and slots in frames, the decision was taken that edges and slots should be read as universal quantifications rather than existential quantifications. In our example, the edge has-color would read that "frogs may only have color green."

In most applications of DLs, it is crucial to reason with *terminologies* or *TBoxes*[2], rather than with isolated concept descriptions. Unfortunately, as soon as terminologies are taken into consideration, tractability turns out to be unattainable in $\mathcal{FL}_0$. Classifying even the simplest form of terminologies (known as *acyclic* or *unfoldable* TBoxes) that admit only acyclic concept definitions was shown to be coNP-hard [Neb90]. If the most general form of terminologies is admitted (known as *general* TBoxes), which consists of general concept inclusion (GCI) axioms as supported by all modern DL systems, then classification in $\mathcal{FL}_0$ even becomes ExpTime-complete [BBL05].

For these reasons, and also because of the need for expressive DLs in applications, from the mid 1990s on, the DL community has mainly given up on the quest of finding tractable DLs. Instead, it investigated more and more expressive DLs, for which reasoning is worst-case intractable. The goal was then to find practical reasoning procedures, i.e., algorithms that are easy to implement and optimize, and which—though worst-case exponential or even worse—behave well in practice (see, e.g., [HST00, HS04]). This line of research has resulted in the availability of highly optimized DL systems for expressive Description Logics based on tableau algorithms [Hor98, HM01b], and in successful applications—most notably is the recommendation by the W3C of the DL-based *Web Ontology Language* (better known as *OWL*) [HPSv03] as the ontology language for the Semantic Web.

At the beginning of the present decade, the choice of value restrictions as a *sine qua non* of DLs has been reconsidered. On the one hand, it was shown that the DL $\mathcal{EL}$, which allows for conjunction and existential restrictions, has better algorithmic properties than $\mathcal{FL}_0$. To be more precise, classification of both acyclic and cyclic $\mathcal{EL}$ TBoxes is tractable [Baa03], and this remains so even if general TBoxes with GCIs are admitted [Bra04b]. Table 1.1 compares the worst-case complexity of reasoning in $\mathcal{FL}_0$ and $\mathcal{EL}$ w.r.t. different kinds of terminologies (refer to the corresponding columns in Table 2.3 on page 23). On the other hand, there are applications where value restrictions are not needed, and where the expressive power of $\mathcal{EL}$ or small extensions thereof appear to be sufficient. In fact, the Systematized Nomenclature of Medicine, Clinical Terms, [SPSW01, Spa05] employs $\mathcal{EL}$ with an acyclic TBox extended with role inclusion axioms. Also, the Gene Ontology [ABB$^+$00], the thesaurus of the US

[1]These and other concept constructors, including their syntactic format and formal semantics, are described in Table 2.1 on page 19.

[2]Various types of DL terminologies are introduced in Section 2.2

Type of terminologies	$\mathcal{FL}_0$	$\mathcal{EL}$
the empty TBox	polynomial [BL84]	polynomial [BKM98]
acyclic TBoxes	coNP-complete [Neb90]	polynomial [Baa03]
cyclic TBoxes	PSpace-complete [Baa96, Kd03]	polynomial [Baa03]
general TBoxes	ExpTime-complete [BBL05]	polynomial [Bra04b]

Table 1.1: Comparing the Description Logics $\mathcal{FL}_0$ and $\mathcal{EL}$.

National Cancer Institute and the Foundational Model of Anatomy can be seen as acyclic $\mathcal{EL}$ TBoxes. Finally, large parts of the GALEN Medical Knowledge Base [RH97] can also be expressed in $\mathcal{EL}$ with GCIs, role hierarchy, and transitive roles.

DL systems

As mentioned above, very expressive DLs have been investigated and practical algorithms based on the tableau calculus have been devised. The tableau-based algorithms for expressive DLs were then optimized and implemented in the DL reasoning systems FaCT [Hor98] and Racer [HM01b]. These implementations used several optimization techniques including the ones developed in [BHN$^+$94, Hor97, HST00, HMT01]. With the highly effective optimization techniques, these reasoning systems turned out to perform surprisingly well on TBoxes from practical applications. It has been observed that hard cases leading to the worst-case behaviors of the algorithms rarely occurred in practice (see, e.g, [Hor98, HST00, HM01a]). This observation encouraged research on pushing expressivity of DLs further and developing practical algorithms and new optimization techniques.

Current tableau-based DL systems, such as FaCT^{++} [TH06], RacerPro [HM01b] and Pellet [SPC$^+$07], not only offer more expressive DLs (i.e., up to $\mathcal{SROIQ}$ [HKS06] which is the logical underpinning of the new Web Ontology Language OWL 2 [CHM$^+$08, CMW$^+$08]) but also employ additional optimizations that have been tailored toward specific applications like biomedical ontologies (see, e.g., [HT05]). Alternative DL systems include KAON2 [Mot06], which implemented an algorithm based on resolution reasoning and disjunctive datalog; and HermiT [MSH07], which implemented a novel calculus known as 'hypertableau.'

As opposed to the reasoners mentioned above, the CEL reasoner [BLS06] supports the lightweight DL $\mathcal{EL}^+$, a tractable extension of $\mathcal{EL}$ (see Chapter 2). At first sight, one might think that a polynomial-time algorithm is always better suited for implementation than worst-case exponential-time algorithms such as the ones underlying those modern DL reasoners. However, due to a plethora of sophisticated optimization techniques that have been developed for tableau algorithms over the last decade, it is far from obvious whether a straightforward implementation of the polynomial-time algorithm can compete with highly-optimized implementations of tableau algorithms. A case in point is our experience with implementing the polynomial-time classification algorithms for cyclic $\mathcal{EL}$ TBoxes introduced in [Baa03]: direct implementations of both the algorithm for subsumption w.r.t. descriptive semantics (based on a reduction to satisfiability of propositional Horn formulas [DG84]) and the algorithm for subsumption w.r.t. greatest fixpoint semantics (based on computing the greatest simulation on a graph [HHK95]) did not lead to satisfactory results on the Gene Ontology [Sun05b].

The CEL system consists of the first implementation of a *refined* polynomial-time classification algorithm [BLS07], where an obvious obstacle for efficient implementation of the algorithm given in [BBL05] is removed—namely, the uninformed, brute-force search for applicable completion rules (see Subsection 4.1.4 for the details). With almost no further optimizations, the first implementation has demonstrated high performance on specific applications of biomedical ontologies [BLS05]. Not only have these empirical results of CEL encouraged the use of the tractable DL family of $\mathcal{EL}$, but they have also sparked interest in further research into new optimization techniques for expressive DLs. By taking into account the underlying logic of the input TBox, a tableau-based reasoner can wisely select the most optimal algorithm and/or enable specific optimizations to perform reasoning (see, e.g., [HT05, HMW08]).

Besides the standard reasoning of classification, CEL supports incremental classification, module extraction and axiom pinpointing. The reasoning techniques implemented in the CEL system, together with the implementation and empirical evaluation, are the major results of the present thesis which are described in Chapter 4, 5 and 6, respectively.

Formal ontologies

In the context of knowledge representation and reasoning, *(formal) ontologies*[3] are specifications of conceptualization [Gru93b]. Conceptualization usually stands for entities in reality, their categories and relationships among them. A specification assigns symbolic representation to entities, categories and relationships, and constrains the possible interpretations by means of formal axioms, cf. [Gru93b, Gua98]. Hence, a terminology in the DL sense, e.g., a general TBox, can be seen as a formal ontology.

Ontologies based on Description Logics can be used, for example, to formalize technical vocabularies and to perform semantic indexing and query answering in variety of applications, including:

- natural language processing,
- object-oriented database,
- software information systems,
- multimedia search systems,
- chemical process engineering,
- the Semantic Web, and
- biomedical informatics.

[3]According to [Qui53], *Ontology* (discipline) is the study of "what there is." In our context, *an ontology* is merely a formal specification of knowledge about a domain of discourse.

In this thesis, we focus attention on ontologies from the domain of biomedical informatics because of their shared characteristics that match the Description Logic under consideration. The next section investigates a few biomedical ontologies and discusses their common characteristics and challenges of reasoning with them.

1.2 Biomedical Ontologies

Given the vast knowledge of biology and medicine acquired even before the advent of computing systems, not to mention the complexity of this knowledge, it is not astonishing that researchers in these scientific branches have encountered the problem of representing their knowledge in a systematic way. Several efforts to systematize biomedical knowledge and standardize terms have eventually resulted in either classifications of diseases, controlled vocabularies, thesauri, terminologies or ontologies. By formalizing knowledge in an unequivocal way, the biomedical community can create a common understanding of the subject in the sense that it helps reduce redundancy in and heterogeneity of the domain knowledge.

The Systematized Nomenclature of Medicine, Clinical Terms (Snomed ct) is a comprehensive clinical and medical ontology that covers a wide range of concepts in the domain, including anatomy, diseases, pharmaceutical products, clinical findings and medical procedures [SPSW01, Spa05]. The presence of this terminology dated back to 1965 when the College of American Pathologist (CAP) released the Systematized Nomenclature of Pathology (Snop) which was extended in 1997 to the first version of Systematized Nomenclature of Medicine, known as Snomed Reference Terminology (rt) [SCC97, Spa00, Rec07]. It was claimed to be the first version of this terminology to use the formal semantics (through the KRSS syntax [PSS93]) of the Description Logic. The terminology has since been continually revised and finally merged with Clinical Terms Version 3 [OPR95] to form the much more comprehensive terminology Snomed ct, which comprises almost four hundred thousand concept definitions like

$$\begin{aligned}\mathsf{AmputationOfFinger} \equiv\ & \mathsf{HandExcision} \sqcap \\ & \exists \mathsf{roleGroup}.(\ \exists \mathsf{direct\text{-}procedure\text{-}site}.\mathsf{Finger}_S \sqcap \\ & \qquad \exists \mathsf{method}.\mathsf{Amputation}\ )\end{aligned}$$

A small extension of the Description Logic $\mathcal{EL}$ has so far been used as a primary language for development where automated reasoning of classification has proved useful in the generation of Snomed ct in 'normal form' [Spa01] for distribution purposes.

In 2007, the International Health Terminology Standards Development Organisation (IHTSDO) has been founded with the purpose to internationalize and standardize Snomed ct as the reference clinical terminology among the member countries [IHT07]. At the time of writing, Snomed ct is being translated into various other languages apart from English and is freely available to all of the (currently nine) IHTSDO member countries.

Due to its appropriate underlying logical formalism and large scale, Snomed ct was taken as the prime ontology in our case study and experiments. More insight into this medical ontology, including some logical issues, is given in a dedicated chapter—Chapter 3.

In 1992, the European project GALEN[4] was launched in order to facilitate the integration of medical information systems by means of a common reference model for medical terminology. Unlike the approach to SNOMED RT by the CAP, who translated an existing classification system for its previous version of the terminology into DL, the strategy of the GALEN project was to invent a suitable KR formalism before developing the actual terminology. Through compilation of specific requirements for the medical domain, the 'GALEN Representation and Integration Language (GRAIL)' was devised and used to develop the GALEN medical ontology [RH97].

In order to benchmark his DL reasoner FaCT, Horrocks [Hor97] has translated the GALEN ontology into the DL format by proposing a mapping from GRAIL statements to equivalent logical axioms formulated in the DL $\mathcal{ALCH}f_{R^+}$ or $\mathcal{SH}f$. An investigation of GRAIL under scrutiny has revealed that it also supports so-called inverse roles, but this was not included in the ontology fragments used as benchmarks in [Hor97]. The mapping can easily be extended to take into account inverse roles. Since concept disjunction, negation and universal quantification have not been included in the GRAIL language, a more fine-tuned DL for GALEN is $\mathcal{ELHI}f_{R^+}$ [Vu08].

An interesting feature of GALEN that distinguishes it from most biomedical ontologies is that it makes use of GCIs which can be used to add levels of granularity and to supplement constraints. A classical example [HRG96] for the former case is the use of a GCI like

$$\mathsf{Ulcer} \sqcap \exists \mathsf{has\text{-}loc}.\mathsf{Stomach} \sqsubseteq \mathsf{Ulcer} \sqcap \exists \mathsf{has\text{-}loc}.(\mathsf{Lining} \sqcap \exists \mathsf{part\text{-}of}.\mathsf{Stomach})$$

to bridge the term 'ulcer of stomach' to the more fine-grained term 'ulcer of lining of stomach' since it is known that ulcer of the stomach is specific to the lining of the stomach. An example for the latter case is the use of a GCI like

$$\mathsf{VitaminK1} \sqsubseteq \exists \mathsf{has\text{-}function}.\mathsf{Catalysing}$$

to supplement the knowledge that "vitamin K1 has a function as a catalyst," which would have been inappropriate to have been incorporated into VitaminK1's definition.

Apart from SNOMED CT and GALEN, the repository of Open Biomedical Ontologies (OBO) is a large library of ontologies from the biological and medical domains. Most of the ontologies available in the repository have been written in 'OBO flat file format,' which was originally designed for the Gene Ontology (GO) [ABB$^+$00]. The OBO format is relatively informal, but there have been attempts to map this format to Description Logic semantics. An example is given in [Sun05b] where two translations of the Gene Ontology were proposed, one of which turned out to correspond to the OBO's intended semantics.

[4]Generalised Architecture for Languages, Encyclopaedias and Nomenclatures in Medicine; see `http://www.OpenGALEN.org`.

More recently, Golbreich et al. has defined a semantic mapping from the OBO flat file format to OWL [GHH$^+$07]. As a consequence of this mapping, several other biomedical ontologies readily available in the OBO format have been being translated into OWL. The translation is beneficial to both the biomedical ontology community and the reasoning community. On the one hand, the ontology developers from the biomedical community can exploit ontology design and maintenance methodologies, ontology editors and automated reasoning tools, several of which are available for free. On the other hand, the translation also benefits the reasoning community since it gives rise to new ontologies for benchmarking reasoning algorithms and implementations.

Similar to SNOMED CT and GALEN, biomedical ontologies developed using the OBO language turned out to be expressible in the DL $\mathcal{EL}$ or tractable extensions thereof. Unlike GALEN, however, they do not use GCIs and purely rely on concept definitions. Notable examples of OBO ontologies are the Gene Ontology, the thesaurus of the US National Cancer Institute (NCI) and the Foundational Model of Anatomy (FMA). All of the biomedical ontologies mentioned so far have been used as benchmarks in the empirical evaluation of various reasoning techniques. Their individual characteristics, including the size and types of axioms, are described in Section 6.1.

Common characteristics shared among biomedical ontologies can be summarized as follows:

- Disjunction, negation, universal quantification and cardinality restrictions are not explicitly required to formulate sensible ontologies in the biomedical domain. Only concept constructors in $\mathcal{EL}$, i.e., conjunction and existential quantification, appear to be adequate.

- Transitivity and role hierarchy axioms play an indispensable role in biomedical ontologies; while other role axioms, such as right-identity, functionality, domain and range restrictions, are sometimes required.

- As an inevitable consequence of the complexity of the domain, biomedical ontologies are typically of very large scale, comprising hundreds of thousands of concept definitions in some cases.

For most existing biomedical ontologies, the scalability of reasoning seems to outweigh the expressivity of the ontology language. In order to address these specific requirements, the DL language and reasoning techniques developed in this thesis have been tailored toward ontologies of the kind—the language is sufficiently expressive for the application at hand[5] and the reasoning techniques can be accomplished in polynomial time.[6]

1.3 Tasks for Ontology Design and Maintenance

Before identifying the tasks for design and maintenance of formal ontologies, it is essential to cast some light on what makes good ontologies. For this reason, we

[5]An exception is GALEN; see Section 6.1 for some discussion.
[6]An exception is *full* axiom pinpointing; see Section 5.2 for the details.

summarize the five basic principles for the design of formal ontologies proposed in [Gru93a].

- *Clarity*: the ontology should effectively convey the intended meaning of its defined terms. The meaning should be dissociated from the social and computational context. Complete definitions providing both necessary and sufficient conditions are preferred over primitive ones providing only necessary conditions.

- *Coherence*: the ontology should be logically consistent. Also, implicit consequences that contradict the domain knowledge should not be inferred from the ontology.

- *Extensibility*: the ontological structure should be so that it is possible to extend the ontology or refine some of its definitions monotonically, i.e., the meaning of existing terms should be preserved.

- *Minimal encoding bias*: the ontology should be specified at the knowledge level, independent of a specific symbol-level encoding.

- *Minimal ontological commitment*: the ontology should make as few claims about the domain of discourse as possible, i.e., only terms essential for the intended use of the ontology are defined. The weakest theory should be used to minimize ontological commitment and thus allow the largest number of potential models.

Note that DL-based knowledge representation systems promote good ontologies according to some of the above criteria. Clarity and minimal encoding criteria are attained as direct results from the well-defined formal semantics of Description Logics. Primitive and full concept definitions in DLs provide unambiguous utility to specify terms, while general concept inclusions allow to supplement additional constraints without having to interfere with existing definitions. Additionally, minimal encoding bias can be alleviated with the help of advanced ontology editors and visualization tools (e.g., Protégé as shown in Figure 1.2) that avoid hassles of a specific syntax (e.g., OWL) and thus help to promote coding at the knowledge level.

More relevant to the present dissertation are the roles of reasoning support in shaping good ontologies in terms of coherence and extensibility. In [LLS06, LBF$^+$06], the authors described tasks relevant for ontology design and maintenance, and argued how logical reasoning support can be used to accomplish them. The following are tasks for ontology design and maintenance that directly relate to reasoning support proposed in this thesis.

Authoring concept definitions. One of the most central activities during ontology design and maintenance are the formulation of new concept definitions (in the case of design) and the refinement of existing concept definitions (in the case of maintenance). Due to the *declarative* style of DL semantics, the ontology developer cannot use some execution model to guide his intuition about the effects of design decisions. Unwanted implicit consequences may be incurred without awareness of the developer and can be far from easy to detect by hand.

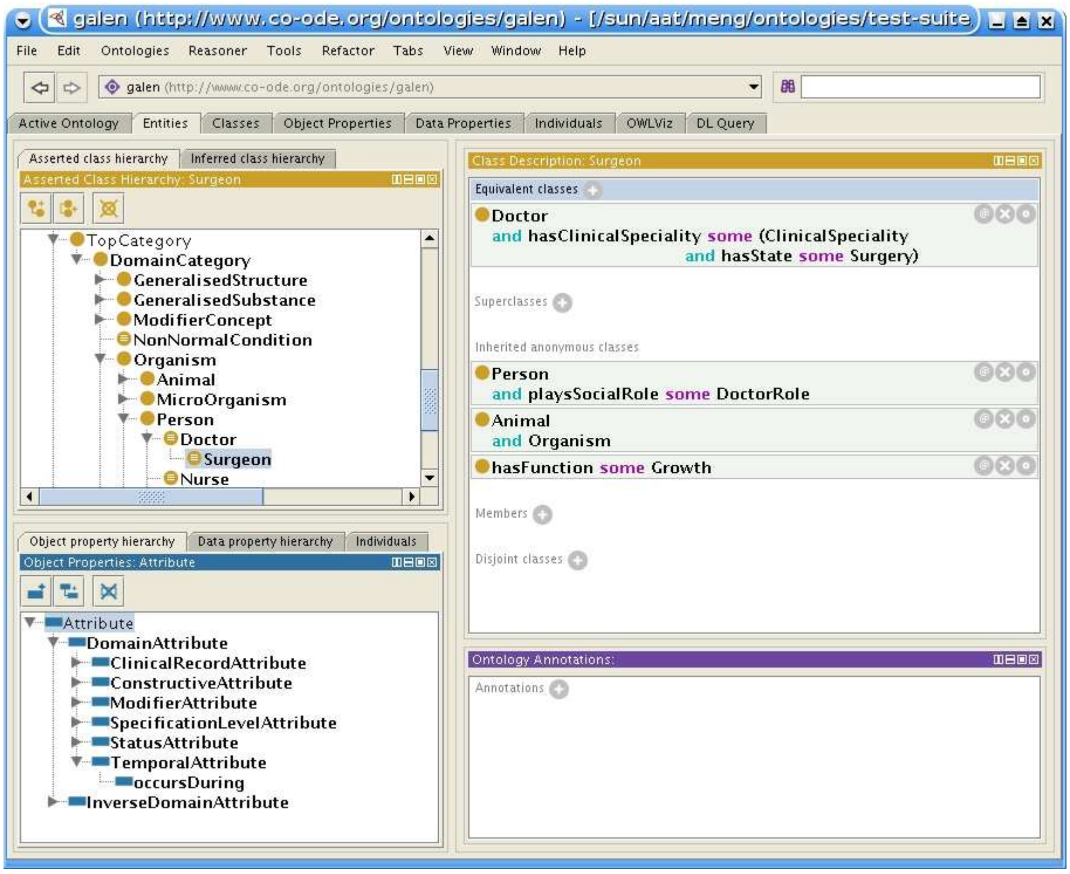

galen (http://www.co-ode.org/ontologies/galen) - [/sun/aat/meng/ontologies/test-suite]
File Edit Ontologies Reasoner Tools Refactor Tabs View Window Help
galen (http://www.co-ode.org/ontologies/galen)
Active Ontology Entities Classes Object Properties Data Properties Individuals OWLViz DL Query
Asserted class hierarchy Inferred class hierarchy
Asserted Class Hierarchy: Surgeon
TopCategory
DomainCategory
GeneralisedStructure
GeneralisedSubstance
ModifierConcept
NonNormalCondition
Organism
Animal
MicroOrganism
Person
Doctor
Surgeon
Nurse
Object property hierarchy Data property hierarchy Individuals
Object Properties: Attribute
Attribute
DomainAttribute
ClinicalRecordAttribute
ConstructiveAttribute
ModifierAttribute
SpecificationLevelAttribute
StatusAttribute
TemporalAttribute
occursDuring
InverseDomainAttribute
Class Description: Surgeon
Equivalent classes
Doctor and hasClinicalSpeciality some (ClinicalSpeciality and hasState some Surgery)
Superclasses
Inherited anonymous classes
Person and playsSocialRole some DoctorRole
Animal and Organism
hasFunction some Growth
Members
Disjoint classes
Ontology Annotations:
Annotations

Figure 1.2: The Protégé ontology editor.

Such implicit consequences could be that the ontology is logically inconsistent (i.e., there is no model); that a concept in the ontology is unsatisfiable (i.e., it cannot be instantiated); or that one concept is a subconcept of another. The first two types of consequences immediately indicate flaws in the ontology since an ontology is intended to represent at least a possible model, and a concept to represent a class of objects. Subsumption may or may not be intended depending on its intuition in the domain of discourse. At any rate, implicit subsumptions need to be detected and reported to the domain expert for inspection.

The mentioned tasks directly correspond to the reasoning problems of *consistency*, *satisfiability* and *subsumption* in DL. Most DL systems usually support *classification* which is the computation of the *subsumption hierarchy*. Not only is classification useful in detecting unwanted subsumptions, it also provides with a visualization of the ontology's structure and is the premier way to navigate and access the ontology. Classification is normally implemented by means of multiple subsumption checks, therefore it is of utmost important for ontology maintenance that such a computation can be done incrementally when a small change is applied, i.e., previous classification information is reused.

Error management. Similar to writing large software programs, building large-scale ontologies is an error-prone endeavor. The aforementioned reasoning support can help alert the developer to the existence of errors. For example, 'amputation of finger' is inferred to be a subconcept of the concept 'amputation of upper limb' in SNOMED CT, which is clearly unintended [SBSS07, SMH07] and reveals a modeling error. However, given an unintended subsumption relationship in a large ontology like SNOMED CT with almost four hundred thousand axioms, it is not always easy to find the erroneous axioms responsible for it by hand.

Automated reasoning support for error management comes in three flavors: pinpointing, explanation and revision. *Pinpointing* identifies those concept definitions responsible for an error, while *explanation* aims to provide a convincing argument that also involves explaining the interplay between the relevant concept definitions. *Automatic revision* goes one step further by making concrete suggestions for how to resolve the error. In the scope of this thesis, we focus attention on pinpointing.

Ontology import. One of the first decisions to be made when building an ontology is whether to start from scratch or to reuse available knowledge in existing ontologies. For example, when building an ontology describing medicinal products of a pharmaceutical company, concepts of specific medical substances and human body parts may be used. In order to guarantee certain relationships among those concepts, the designer may want to include more details about them. Since these details have already been formulated properly in a standardized ontology like SNOMED CT, it should be less time consuming and more accurate to *import* the ontology. The problem, however, is that such standardized ontologies are typically designed to be comprehensive, thus very large, therefore importing the whole ontology unnecessarily introduces overhead in computation.

It is thus helpful to be able to extract a small portion of the ontology that contains only concept definitions relevant to the needs, i.e., knowledge about the concepts to be imported. To this end, the automated reasoning support of module extraction computes a subset of the ontology that is ensured to be small and adequately capture the meaning of the imported concepts.

There are a few other tasks for ontology design and maintenance that can be mapped to so-called 'non-standard inferences', such as least common subsumers, most specific concepts and concept matching. Non-standard inferences have been investigated thoroughly in [Küs00, Bra06, Tur07].

1.4 Dissertation Outline

In this section, a content outline of the present dissertation is provided. Since a large portion of the (technical and empirical) results presented in this dissertation have already been published in scientific journals, conferences or workshops, it also gives an account of relevant publications.

The dissertation consists of three major parts. The first part, consisting of Chapter 1, 2 and 3, is concerned about introduction of the terminological and notational conventions that govern the rest of the thesis. Moreover, it gives motivations of the work from real-world ontology applications. Technical results are presented in the second part, which is divided into two chapters: Chapter 4 presents techniques for standard reasoning, while Chapter 5 does for supplemental reasoning. The last part reports on several evaluations using realistic life science ontologies in Chapter 6 and gives summary of the outcome of the work in Chapter 7.

Chapter 2 is dedicated to most of preliminaries about Description Logics (DLs) that are frequently referred to by other chapters. It introduces a number of well-known DL dialects with emphasis on the $\mathcal{EL}$ family, depicts a small medical ontology that will be used as a running example in technical chapters, and gives formal definitions of an ontology, standard and supplemental reasoning problems.

In Chapter 3, we present a case study of using DLs in the $\mathcal{EL}$ family, featuring the large-scale medical ontology Snomed ct. Several ontological and logical issues found in Snomed ct are addressed in Section 3.2, and solutions to them using the DL $\mathcal{EL}^+$ are proposed in Section 3.3.

These results have been published in:

[SBSS07] Boontawee Suntisrivaraporn, Franz Baader, Stefan Schulz, and Kent Spackman. Replacing SEP-triplets in Snomed ct using tractable Description Logic operators. In Jim Hunter Riccardo Bellazzi, Ameen Abu-Hanna, editor, *Proceedings of the 11th Conference on Artificial Intelligence in Medicine (AIME'07)*, volume 4594 of *Lecture Notes in Computer Science*, pages 287–291. Springer-Verlag, 2007.

[SSB07] Stefan Schulz, Boontawee Suntisrivaraporn, and Franz Baader. Snomed ct's problem list: Ontologists' and logicians' therapy suggestions. In *Proceedings of The Medinfo 2007 Congress*, Studies in Health Technology and Informatics (SHTI-series), pages 802–806. IOS Press, 2007.

[LBF$^+$06] Carsten Lutz, Franz Baader, Enrico Franconi, Domenico Lembo, Ralf Möller, Riccardo Rosati, Ulrike Sattler, Boontawee Suntisrivaraporn, and Sergio Tessaris. Reasoning support for ontology design. In Bernardo Cuenca Grau, Pascal Hitzler, Connor Shankey, and Evan Wallace, editors, *Proceedings of the second international workshop OWL: Experiences and Directions*, 2006.

In Chapter 4, we present several technical results for standard reasoning with $\mathcal{EL}^+$ ontologies. Section 4.1 describes a polynomial-time classification algorithm, together with normalization and reduction of range restrictions which are also available in $\mathcal{EL}^+$. A refined version of this algorithm for implementation purposes is proposed in Subsection 4.1.4, while optimization techniques are discussed in Subsection 4.1.5.

In Section 4.2, we introduce a modification to the refined classification algorithm in which the computation is directed by the goal subsumption. The so-called goal-directed subsumption algorithm is then used to effectively answer subsumption queries.

In contrast to tableau-based algorithms, the innate output of the $\mathcal{EL}^+$ classification algorithm is complete subsumer sets. By exploiting this fact, we develop a simplified variant of the 'enhanced traversal method' to efficiently construct the concept hierarchy from the subsumer sets. Comparisons of the original and simplified methods are discussed in Section 4.3.

Section 4.4 describes an algorithm for incremental classification that can be utilized not only in certain scenarios of incremental reasoning but also in querying complex subsumptions.

Finally, we investigate in Section 4.5 a technique for encoding ABox assertions as general concept inclusions in the TBox. With such an encoding, an ontology equipped with an assertional component can be realized by exploiting the classification algorithm, as well as its optimizations.

Most of the results have been published in the following papers:[7]

[BLS05] Franz Baader, Carsten Lutz, and Boontawee Suntisrivaraporn. Is tractable reasoning in extensions of the Description Logic $\mathcal{EL}$ useful in practice? In *Proceedings of the 2005 International Workshop on Methods for Modalities (M4M-05)*, 2005.

[BLS06] Franz Baader, Carsten Lutz, and Boontawee Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In U. Furbach and N. Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 287–291. Springer-Verlag, 2006.

[BLS07] Franz Baader, Carsten Lutz, and Boontawee Suntisrivaraporn. Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice? *Journal of Logic, Language and Information, Special Issue on Method for Modality (M4M)*, 2007.

[Sun08] Boontawee Suntisrivaraporn. Module extraction and incremental classification: A pragmatic approach for $\mathcal{EL}^+$ ontologies. In Sean Bechhofer, Manfred Hauswirth, Joerg Hoffmann, and Manolis Koubarakis, editors, *Proceedings of the 5th European Semantic Web Conference (ESWC'08)*, volume 5021 of *Lecture Notes in Computer Science*, pages 230–244. Springer-Verlag, 2008.

Chapter 5 describes technical results for two important supplemental reasoning services: modularization in Section 5.1 and axiom pinpointing in Section 5.2. In Subsection 5.1.1, we introduce a new kind of module based on reachability in directed hypergraph. We present a number of interesting properties of this kind of module, and

[7]Results presented in this dissertation are in fact extensions to those in the referred publications because here we consider the *more expressive* DL $\mathcal{EL}^+$ (refer to Definition 6 on page 22).

in Subsection 5.1.2, a connection between the goal-directed subsumption algorithm and reachability-based modules is established. In the last subsection, we compare our method based on reachability to other module extraction approaches. In particular, we prove that the reachability-based module is indeed equivalent to the minimal module based on syntactic locality.

In Section 5.2, various techniques for axiom pinpointing are discussed and newly developed. We start with identifying hardness complexity of the problem in Subsection 5.2.1. The black-box approach is addressed in Subsection 5.2.2, while the glass-box approach, as well as a combined approach, is discussed in Subsection 5.2.3. Finally, a novel approach using the reachability-based module to optimize the black-box method is described in Section 5.2.4.

Most of the results in this chapter has been published in:

[Sun08] Boontawee Suntisrivaraporn. Module extraction and incremental classification: A pragmatic approach for $\mathcal{EL}^+$ ontologies. In Sean Bechhofer, Manfred Hauswirth, Joerg Hoffmann, and Manolis Koubarakis, editors, *Proceedings of the 5th European Semantic Web Conference (ESWC'08)*, volume 5021 of *Lecture Notes in Computer Science*, pages 230–244. Springer-Verlag, 2008.

[BPS07a] Franz Baader, Rafael Peñaloza, and Boontawee Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}$. In *Proceedings of the 2007 International Workshop on Description Logics (DL2007)*, CEUR-WS, 2007.

[BPS07b] Franz Baader, Rafael Peñaloza, and Boontawee Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}^+$. In *Proceedings of the 30th German Conference on Artificial Intelligence (KI'07)*, volume 4667 of *Lecture Notes in Computer Science*, Osnabrück, Germany, 2007. Springer.

[BS08] Franz Baader and Boontawee Suntisrivaraporn. Debugging SNOMED CT using axiom pinpointing in the Description Logic $\mathcal{EL}^+$. In *Proceedings of the 3rd Knowledge Representation in Medicine Conference (KR-MED'08): Representing and Sharing Knowledge Using* SNOMED, 2008.

Chapter 6 describes several large-scale ontologies from the life science application domain that were used as benchmarks in our empirical evaluations. Section 6.1 presents the information concerning various aspects of these ontologies. In Section 6.2, we describe the experimental setting and results for each technique evaluation.

We then summarize the technical and empirical results achieved in this PhD project and suggest directions for future work in Chapter 7.

Last but not least, the following list includes other works that have been published during the course of this PhD project but are out of scope of the present dissertation:

[SMS06] Stefan Schulz, Kornel Markó, and Boontawee Suntisrivaraporn. Complex occurrents in clinical terminologies and their representation in a formal language. In *Proceedings of the First European Conference on* SNOMED CT *(SMCS'06)*, Copenhagen, Denmark, 2006.

[TBK$^+$06] Anni-Yasmin Turhan, Sean Bechhofer, Alissa Kaplunova, Thorsten Liebig, Marko Luther, Ralf Möller, Olaf Noppens, Peter Patel-Schneider, Boontawee Suntisrivaraporn, and Timo Weithöner. Dig 2.0 – towards a flexible interface for Description Logic reasoners. In Bernardo Cuenca Grau, Pascal Hitzler, Connor Shankey, and Evan Wallace, editors, *Proceedings of the Second International Workshop OWL: Experiences and Directions*, November 2006.

[BNS08] Franz Baader, Novak Novakovic, and Boontawee Suntisrivaraporn. A proof-theoretic subsumption reasoner for hybrid $\mathcal{EL}$-TBoxes. In *Proceedings of the 2008 International Workshop on Description Logics (DL2008)*, CEUR-WS, 2008.

[SQJH08] Boontawee Suntisrivaraporn, Guilin Qi, Qiu Ji, and Peter Haase. A modularization-based approach to finding all justifications for OWL DL entailments. In John Domingue and Chutiporn Anutariya, editors, *Proceedings of the 3th Asian Semantic Web Conference (ASWC'08)*, volumn 5367 of *Lecture Notes in Computer Science*. Springer-Verlag, 2008.

Chapter 2

Preliminaries

In this chapter, we introduce the preliminary notions for knowledge representation systems based on Description Logics (or DL system, for short), with emphasis on the DL family of $\mathcal{EL}$ which is considered throughout the thesis.

In general, a DL system comprises three main components—the description language, the ontology, and the reasoning services—which make up a set of logical machinery for various ontology-based tasks, from design and maintenance to re-use and inter-operation. Figure 2.1 depicts generic architecture of such a system, with which ontology developers and users may be interacting through an ontology graphical user interface (GUI). In the first section, we introduce core elements of any description language that are building blocks of conceptual representation objects and classes of objects from real world application domains. Then, we give definitions of various kinds of ontological axioms in Section 2.2, which can be distinguished into two groups, namely terminological and assertional axioms as suggested in Figure 2.1. Based on these axioms, the TBox and ABox components of an ontology are defined, and a small example of life science ontology motivated by Galen and Snomed is also given. Section 2.3 and 2.4 formally define logical inference problems that correspond to standard and supplemental reasoning services provided by DL systems, respectively. Non-standard inferences, such as the computation of least common subsumer and most specific concept, have been proposed and developed to help construct an ontology in a bottom-up fashion. This is, however, beyond the scope of the present thesis, and we refer interested readers to [Küs00, Tur07].

2.1 The $\mathcal{EL}$ Family of Description Logics

As mentioned before, Description Logics (DLs) are a family of symbolic languages that are used to formulate concepts and relationships among them. Alongside ontological constructors (introduced in the next section), each DL is identified by its (concept) description language. The description language specifies a set of so-called *concept constructors* allowed in a given logic. These constructors are used to inductively define complex *concept descriptions* starting with two disjoint sets of *concept names* CN and

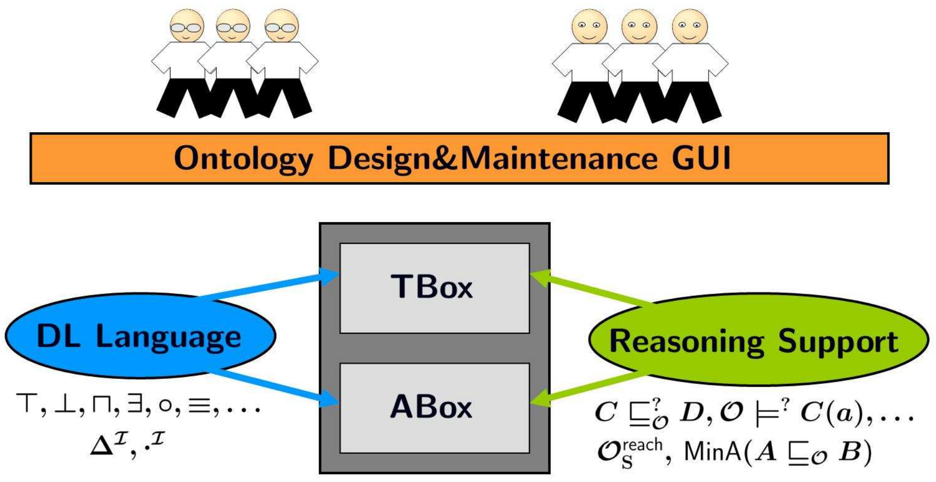

Ontology Design&Maintenance GUI
TBox
ABox
DL Language
$\top, \bot, \sqcap, \exists, \circ, \equiv, \ldots$
$\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}$
Reasoning Support
$C \sqsubseteq^?_{\mathcal{O}} D, \mathcal{O} \models^? C(a), \ldots$
$\mathcal{O}^{\mathsf{reach}}_{\mathrm{S}}$, $\mathsf{MinA}(A \sqsubseteq_{\mathcal{O}} B)$

Figure 2.1: Architecture of a Description Logic-based knowledge representation system (for short, DL system).

role names RN.[1] As a general rule, the more concept constructors the logic provides, the more expressive the concept descriptions can be formulated. Table 2.1 lists core concept constructors that are widely considered in the literature. The second and third column show the syntax and semantics elements, respectively. The DL that provides exactly the concept constructors from this table is known as *attributive language with complements* or $\mathcal{ALC}$, which is the smallest Boolean-closed DL.[2] There are, however, a number of interesting sub-Boolean DLs, most of which disallow disjunction and (full)[3] negation. As motivated in the first chapter, we are concerned with the sub-Boolean DL $\mathcal{EL}$ and a few extensions thereof. In the following, we give a formal definition for the syntax and semantics of this logic.

Definition 1 (Syntax of $\mathcal{EL}$). Let CN and RN be two disjoint sets of concept and role names, respectively. Then, $\mathcal{EL}$ *concept descriptions* or *concepts* are defined inductively as follows:

- each concept name $A \in \mathsf{CN}$ is an $\mathcal{EL}$ concept description, and
- if C, D are $\mathcal{EL}$ concept descriptions and $r \in \mathsf{RN}$ is a role name, then the top concept $\top$, concept conjunction $C \sqcap D$, and existential quantification $\exists r.C$ are

[1] In certain (expressive) description languages, a third set Ind of individuals can be used to describe so-called nominal concepts. In the scope of this thesis, however, we consider individuals only when it comes to the assertional component of an ontology.

[2] Strictly speaking, a DL must provide at least one quantifier, existential or universal. Thus, the logic with the first five concept constructors is not a DL in this respect and is, in fact, equivalent to the propositional logic.

[3] Some DLs, such as $\mathcal{ALE}$, allow for so-called *atomic negation* $\neg A$ which can be applied only in front of *undefined* concept names A (see Definition 3 on page 20).

Constructor name	Syntax	Semantics
top concept	$\top$	$\Delta^\mathcal{I}$
bottom concept	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$C^\mathcal{I} \cap D^\mathcal{I}$
disjunction	$C \sqcup D$	$C^\mathcal{I} \cup D^\mathcal{I}$
negation	$\neg C$	$\Delta^\mathcal{I} \setminus C^\mathcal{I}$
existential quantification	$\exists r.C$	$\{d \in \Delta^\mathcal{I} \mid \exists e : (d, e) \in r^\mathcal{I} \wedge e \in C^\mathcal{I}\}$
universal quantification	$\forall r.C$	$\{d \in \Delta^\mathcal{I} \mid \forall e : (d, e) \in r^\mathcal{I} \rightarrow e \in C^\mathcal{I}\}$

Table 2.1: Syntax and semantics of concept constructors.

also $\mathcal{EL}$ concept descriptions.

An $\mathcal{EL}$ concept description is *atomic* if it is the top concept or a concept name from CN. Otherwise, it is said to be *complex*. $\diamond$

By convention, we use letters (possibly with subscripts and/or superscripts) A, B to range over concept names, C, D, E over concept descriptions, and r, s, t over role names. An example of $\mathcal{EL}$ concept description is:

$$\mathsf{Ulcer} \sqcap \exists \mathsf{has\text{-}location}.\mathsf{Mouth} \sqcap \exists \mathsf{caused\text{-}by}.\mathsf{Bacterium}$$

which represents the concept of 'bacterial oral ulcer.' Concept descriptions in other DL dialects are defined in the same fashion but with different sets of constructors. The upper part of Table 2.3 on page 23 illustrates the supported concept constructors of various DLs related to $\mathcal{EL}$. Note that different logics, such as $\mathcal{EL}$ and $\mathcal{ELH}$, may provide the same set of concept constructors, but they deviate from each other relative to their ontological constructors. When it is obvious from the context, we may omit the prefix (e.g. '$\mathcal{EL}$') and simply say concept descriptions.

Like any DLs, the semantics of $\mathcal{EL}$ concept descriptions is defined through interpretations:

Definition 2 (Semantics of $\mathcal{EL}$). An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a non-empty set $\Delta^\mathcal{I}$ of interpretation domain and an interpretation function $\cdot^\mathcal{I}$, which assigns to each concept name $A \in \mathsf{CN}$ a subset $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ and to each role name $r \in \mathsf{RN}$ a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$. The interpretation function is extended to concept descriptions by the inductive definitions given in the right column of Table 2.1. $\diamond$

2.2 Ontology: TBox and ABox

In the last section, we have seen concept constructors which are used to define concept descriptions. To formulate statements about facts in the domain of interest, we need an ontological formalism. For instance, one might be interested in stating

Constructor name	Syntax	Semantics
concept definition	$A \equiv C$	$A^{\mathcal{I}} = C^{\mathcal{I}}$
concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
concept disjointness	$C \sqcap D \sqsubseteq \bot$	$C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$
domain restriction	$\mathsf{domain}(r) \sqsubseteq C$	$\{d \in \Delta^{\mathcal{I}} \mid \exists e\colon (d,e) \in r^{\mathcal{I}}\} \subseteq C^{\mathcal{I}}$
range restriction	$\mathsf{range}(r) \sqsubseteq C$	$\{e \in \Delta^{\mathcal{I}} \mid \exists d\colon (d,e) \in r^{\mathcal{I}}\} \subseteq C^{\mathcal{I}}$
functionality	$\mathsf{functional}(r)$	$\forall d \in \Delta^{\mathcal{I}}\colon\ \sharp\{e \in \Delta^{\mathcal{I}} \mid (d,e) \in r^{\mathcal{I}}\} \leq 1$
reflexivity	$\mathsf{reflexive}(r)$	$\forall d \in \Delta^{\mathcal{I}}\colon (d,d) \subseteq r^{\mathcal{I}}$
transitivity	$\mathsf{transitive}(r)$	$\forall d,e,f \in \Delta^{\mathcal{I}}\colon (d,e),(e,f) \in r^{\mathcal{I}} \to (d,f) \in r^{\mathcal{I}}$
role hierarchy	$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
role inclusion	$r_1 \circ \cdots \circ r_k \sqsubseteq s$	$r_1^{\mathcal{I}} \circ \cdots \circ r_k^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$r(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$

Table 2.2: Syntax and semantics of ontological constructors.

that "endocarditis is an inflammation that has location on endocardium tissue," that "the part-of relation is transitive," or that "Mr. Matt is taking antifungal antibiotics." Similarly, such a formalism is characterized by a set of *ontological constructors* the description language provides. Table 2.2 lists most commonly used constructors, of which the middle and right columns show their syntax and semantics elements. These ontological constructors can be divided into three groups: concept axiom constructors, role axiom constructors, and assertion (individual axiom) constructors. Our examples above belong to these three groups, respectively, and can be formulated formally as follows:

$$\mathsf{Endocarditis} \equiv \mathsf{Inflammation} \sqcap \exists \mathsf{has\text{-}location}.\mathsf{Endocardium}$$
$$\mathsf{transitive}(\mathsf{part\text{-}of})$$
$$\mathsf{on\text{-}medication}(\mathsf{MATT}, \mathsf{ANONYM1}), \mathsf{AntifungalAntibiotics}(\mathsf{ANONYM1})$$

Conventionally, concept names are capitalized (e.g. Endocarditis), role names are written in lower case (e.g. part-of), and individuals are written in upper case (e.g. MATT).

Based on these ontological constructors, we formally define the notions of TBox, ABox, and finally, DL-based ontology. Intuitively, an ontology consists of the terminological component (henceforth, TBox) and the assertional component (henceforth, ABox). While the TBox contains definitions, relations, and constraints of terminology (concepts and roles) used in the ontology, the ABox holds assertional statements about concrete individuals w.r.t. that terminology. In DLs, there are a few kinds of TBoxes, varying in their expressivity and complexity. We start with the simplest form of TBox—*unfoldable TBox*:

Definition 3 (TBox). Let A be a concept name and C a concept description. Then, $A \equiv C$ is a *concept definition* and A is said to be *fully defined*, and $A \sqsubseteq C$ is a *primitive concept definition* and A is said to be *primitively defined.* Let $\lhd$ stand for $\equiv$ or $\sqsubseteq$. Then, a *TBox* $\mathcal{T}$ is a finite set of (possibly primitive) concept definitions such that the *uniqueness* condition hold: for each concept name A, there is at most one concept definition $A \lhd C$ for some concept description C.

A *cyclic dependency* is a sequence of definitions $A_1 \lhd C_1, \ldots, A_n \lhd C_n$ such that $A_{(i \bmod n)+1}$ occurs in C_i for all $1 \leq i \leq n$. A TBox $\mathcal{T}$ is *acyclic* or *unfoldable* if it contains no cyclic dependency. Otherwise, it is said to be *cyclic.*

A concept name P in $\mathcal{T}$ is *undefined* if it is neither fully defined nor primitively defined in $\mathcal{T}$. ◇

Note that defined concept names in an unfoldable TBox can always be unfolded into an independent concept description, in the sense that the TBox itself can be dispensed with. A much more expressive formalism of TBox is called *general TBox* which is supported by most state-of-the-art DL reasoners.

Definition 4 (General TBox). Let C, D be concept descriptions. Then, the expression $C \sqsubseteq D$ is a *(general) concept inclusion (GCI).* A *general TBox* is a finite set of concept inclusions. ◇

It is obvious that general TBoxes are more general than unfoldable TBoxes, since GCIs can be used to express (primitive) concept definitions. In fact, a primitive concept definition is a special kind of GCI, whereas a concept definition can be expressed by means of two GCIs: $A \equiv C$ by $A \sqsubseteq C$ and $C \sqsubseteq A$.

Apart from concept definitions and inclusions, there are other interesting and ontologically prominent axiom constructors, some of which are listed in the upper part of Table 2.2. In some cases, one constructor can be simulated by another. For instance, a domain restriction $\mathsf{domain}(r) \sqsubseteq C$ can be expressed by the GCI $\exists r.\top \sqsubseteq C$. Also, reflexivity, transitivity and role hierarchy are specializations of role inclusion axioms of the forms $\epsilon \sqsubseteq r$, $r \circ r \sqsubseteq r$ and $r \sqsubseteq s$, respectively.[4]

Definition 5 (ABox). Let Ind be a set of individuals disjoint from CN and RN. Then, expressions of the forms $C(a)$ and $r(a, b)$ are called *concept assertion* and *role assertion*, respectively, where $a, b \in \mathsf{Ind}$, $r \in \mathsf{RN}$, and C is a concept description. An *ABox* is a finite set of concept and role assertions. ◇

Similar to concept and role names, we adopt the convention that the letters $a, b, c, d, \ldots$ range over individuals. Having defined TBox and ABox, we are now ready to give the definition of ontology. As mentioned earlier, an ontology consists of terminological and assertional parts. Though the assertional component commonly boils down to the ABox we have just defined, the terminological part is somewhat more variable. Some DLs only allow for unfoldable TBoxes, while some others allow for general TBoxes. Yet, there are additional ontological constructors that are supported by some other DLs. In the following, we formally define a tractable DL which is of the main focus of this dissertation:

[4]Nevertheless, we list redundant ones as well for the sake of comprehensibility.

Definition 6 ($\mathcal{EL}^+$ Ontology). Let CN, RN, Ind be disjoint sets of concept names, role names, and individuals, respectively. An *unrestricted $\mathcal{EL}^+$ ontology* $\mathcal{O}$ is a finite set of axioms of the forms indicated in $\mathcal{EL}^+$ column of Table 2.3 and shown in Table 2.2, where $A \in \mathsf{CN}$, $a, b \in \mathsf{Ind}$, $r, r_i, s \in \mathsf{RN}$, and C, D are $\mathcal{EL}$ concept descriptions based on CN and RN.

The ontology $\mathcal{O}$ is an $\mathcal{EL}^+$ *ontology* if it satisfies the syntactic restriction given in Definition 8 on page 22. ◇

In this thesis, the terms 'terminology' and 'ontology' are solely used in the technically strict sense of a DL knowledge base, i.e., a set of logical axioms and assertions. By definition, an ontology may comprise both knowledge about the domain (TBox) and knowledge about individuals in the domain (ABox). We sometimes refer to these two ontological components, respectively, as *terminological* and *assertional* parts of the ontology. The size of an ontology $\mathcal{O}$, denoted by $|\mathcal{O}|$, is the number of symbols used to write it. In the context of axiom pinpointing (see Section 5.2), cardinality of an ontology is considered instead of its size, i.e., the number of ontological axioms occurring in it.

Table 2.3 depicts various well-known DLs with their supported features, where ✔ denotes compulsory features, while ✓ denotes optional features that may or may not be supported.[5] We extend the notion of *atomic concept descriptions* to include the bottom concept $\bot$, which is the only new concept description in $\mathcal{EL}^+$ in addition to the classical $\mathcal{EL}$.

Following Definition 2, we can extend the interpretation function to cover the bottom concept and define the semantics of TBox, ABox, and hence ontology. Intuitively, an ontology has a *model* if all of its axioms can be simultaneously satisfied. Formally, we give the following definition:

Definition 7 (Semantics of $\mathcal{EL}^+$ Ontology). Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation, α an axiom, $\mathcal{O}$ an $\mathcal{EL}^+$ ontology. Then, we say that $\mathcal{I}$ *satisfies* α (written, $\mathcal{I} \models \alpha$) if, and only if, the semantics condition for α from Table 2.2 is fulfilled under $\mathcal{I}$. The interpretation $\mathcal{I}$ is a *model* of $\mathcal{O}$ (written, $\mathcal{I} \models \mathcal{O}$) if, and only if, for each axiom $\alpha \in \mathcal{O}$, $\mathcal{I} \models \alpha$. ◇

The semantics of other DLs are defined in the same fashion by properly taking into consideration the corresponding semantics conditions.

As aforementioned, an $\mathcal{EL}^+$ ontology has to satisfy a certain syntactic restriction, without which the logic turns out to be intractable (and even undecidable) [BBL08]. Basically, undecidability is the result of the intricate interplay of range restrictions and role inclusions. The syntactic restriction defined below prevents precisely this interplay.

Definition 8 ($\mathcal{EL}^+$ Syntactic Restriction). For an ontology $\mathcal{O}$ and role names r, s, we write $\mathcal{O} \models r \sqsubseteq s$ (i.e., s is a supperrole of r w.r.t. $\mathcal{O}$) if, and only if, $r = s$

[5] A few features of $\mathcal{SHIF}$ and $\mathcal{SROIQ}$, including inverse role, nominal and qualified number restriction, are omitted here. Interaction between range restrictions and role inclusions in $\mathcal{EL}^+$ has to conform with the syntactic restriction specified in Definition 8, while role inclusions in $\mathcal{SROIQ}$ need to satisfy an acyclicity condition [HKS06].

DL dialects	$\mathcal{HL}$	$\mathcal{EL}$	$\mathcal{FL}_0$	$\mathcal{ELH}$	$\mathcal{EL}^+$	$\mathcal{ALC}$	$\mathcal{SHIF}$	$\mathcal{SROIQ}$
top concept	✔	✔	✔	✔	✔	✔	✔	✔
bottom concept					✔	✔	✔	✔
conjunction	✔	✔	✔	✔	✔	✔	✔	✔
disjunction						✔	✔	✔
negation						✔	✔	✔
exist. restrictions		✔		✔	✔	✔	✔	✔
value restrictions			✔			✔	✔	✔
concept definition	✓	✓	✓	✓	✔	✓	✓	✓
concept inclusion	✓	✓	✓	✓	✔	✓	✓	✓
domain restriction		✓	✓	✓	✔	✓	✓	✓
concept disjointness					✔	✓	✓	✓
range restriction					✔	✓	✓	✓
functionality							✔	✔
reflexivity					✔			✔
transitivity					✔		✔	✔
role hierarchy				✔	✔		✔	✔
role inclusion					✔			✔
concept assertion	✓	✓	✓	✓	✔	✓	✓	✔
role assertion		✓	✓	✓	✔	✓	✓	✔

Table 2.3: Logical constructors in $\mathcal{EL}$-related formalisms.

or $\mathcal{O}$ contains role inclusions $r_1 \sqsubseteq r_2, \ldots, r_{k-1} \sqsubseteq r_k$ with $r = r_1$ and $s = r_k$. Also, we write $\mathcal{O} \models \mathsf{range}(r) \sqsubseteq C$ if there is a role name s such that $\mathcal{O} \models r \sqsubseteq s$ and $\mathsf{range}(s) \sqsubseteq C \in \mathcal{O}$. The $\mathcal{EL}^+$ *syntactic restriction* is as follows: If $r_1 \circ \cdots \circ r_k \sqsubseteq s \in \mathcal{O}$ with $k \geq 1$ and $\mathcal{O} \models \mathsf{range}(s) \sqsubseteq C$, then $\mathcal{O} \models \mathsf{range}(r_k) \sqsubseteq C$. $\diamond$

Intuitively, the restriction ensures that a role inclusion $r_1 \circ \cdots \circ r_k \sqsubseteq s$, $k \geq 1$, does not induce any new range constraint on the role composition $r_1 \circ \cdots \circ r_k$. Formally, it ensures that if the role inclusion implies a role relationship $(d, e) \in s^{\mathcal{I}}$ in the model, then the range restrictions on s do not impose new concept memberships on e. Note that the condition is vacuously true if the role inclusion is simply a reflexivity statement, role hierarchy statement, a transitivity statement, or a *generalized left-identity* axiom of the form $r_1 \circ \cdots \circ r_k \sqsubseteq r_k$.

Before moving to the next section, we would like to demonstrate practical pertinence of the DL $\mathcal{EL}^+$ to biomedical ontologies by an example. Figure 2.2 portrays a small $\mathcal{EL}^+$ ontology that is motivated by the realistic biomedical ontologies GALEN and SNOMED.[6] The ontology mainly concerns various kinds of inflammatory diseases,[7] related concepts, and relationships among them. Most syntactic elements of

[6]We disclaim responsibility of correctness of the statements in this ontology, both from clinical and ontological perspectives. The example is created with only one objective in mind—to demonstrate features of the logic.

[7]According to `www.dictionary.com`, *appendicitis* is an inflammation of the vermiform appendix,

α_1	$\mathsf{Appendix}$	$\sqsubseteq$	$\mathsf{BodyPart} \sqcap \exists \mathsf{part\text{-}of}.\mathsf{Intestine}$
α_2	$\mathsf{Endocardium}$	$\sqsubseteq$	$\mathsf{Tissue} \sqcap \exists \mathsf{part\text{-}of}.\mathsf{HeartValve} \sqcap \exists \mathsf{part\text{-}of}.\mathsf{HeartWall}$
α_3	$\mathsf{HeartValve}$	$\sqsubseteq$	$\mathsf{BodyValve} \sqcap \exists \mathsf{part\text{-}of}.\mathsf{Heart}$
α_4	$\mathsf{HeartWall}$	$\sqsubseteq$	$\mathsf{BodyWall} \sqcap \exists \mathsf{part\text{-}of}.\mathsf{Heart}$
α_5	$\mathsf{Appendicitis}$	$\equiv$	$\mathsf{Inflammation} \sqcap \exists \mathsf{has\text{-}location}.\mathsf{Appendix}$
α_6	$\mathsf{Endocarditis}$	$\equiv$	$\mathsf{Inflammation} \sqcap \exists \mathsf{has\text{-}location}.\mathsf{Endocardium}$
α_7	$\mathsf{Pancarditis}$	$\equiv$	$\mathsf{Inflammation} \sqcap \exists \mathsf{has\text{-}exact\text{-}location}.\mathsf{Heart}$
α_8	$\mathsf{Inflammation}$	$\sqsubseteq$	$\mathsf{Disease} \sqcap \exists \mathsf{acts\text{-}on}.\mathsf{Tissue}$
α_9	$\mathsf{HeartDisease}$	$\equiv$	$\mathsf{Disease} \sqcap \exists \mathsf{has\text{-}location}.\mathsf{Heart}$
α_{10}	$\mathsf{Tissue} \sqcap \mathsf{Disease}$	$\sqsubseteq$	$\bot$
α_{11}	$\mathsf{HeartDisease} \sqcap \exists \mathsf{causative\text{-}agent}.\mathsf{Virus}$	$\sqsubseteq$	$\mathsf{ViralDisease} \sqcap \exists \mathsf{has\text{-}state}.\mathsf{NeedsTreatment}$
α_{12}	ϵ	$\sqsubseteq$	$\mathsf{part\text{-}of}$
α_{13}	$\mathsf{part\text{-}of} \circ \mathsf{part\text{-}of}$	$\sqsubseteq$	$\mathsf{part\text{-}of}$
α_{14}	$\mathsf{has\text{-}location} \circ \mathsf{part\text{-}of}$	$\sqsubseteq$	$\mathsf{has\text{-}location}$
α_{15}	$\mathsf{has\text{-}exact\text{-}location}$	$\sqsubseteq$	$\mathsf{has\text{-}location}$

Figure 2.2: An example $\mathcal{EL}^+$ ontology $\mathcal{O}_{\mathsf{med}}$.

the aforementioned role expressivity can be simulated by role inclusions.[8] For example, $\mathsf{reflexive}(\mathsf{part\text{-}of})$ can be written as the role inclusion α_{12}, where ϵ denotes the nullary role composition, whereas $\mathsf{transitive}(\mathsf{part\text{-}of})$ can be written as the role inclusion α_{13}. It is worthwhile to note, in particular, that the role inclusion of the form $r \circ s \sqsubseteq r$, called *right-identity*, plays an important role in medical ontologies [Spa00, HS04]. In our example, α_{14} is a right-identity axiom, and we say that the role $\mathsf{part\text{-}of}$ is a right identity for the role $\mathsf{has\text{-}location}$. The cardinality of $\mathcal{O}_{\mathsf{med}}$, as explicitly given by axiom indexes, is 15, whereas the size $|\mathcal{O}_{\mathsf{med}}|$ is 97.

2.3 Standard Reasoning Services

Different DL systems offer different reasoning services (also known as inference problems) to make certain *implicit* knowledge logically captured in an ontology *explicit.* There are, however, a class of reasoning services that are commonly considered mandatory and supported by most DL systems. We refer to this class as *standard reasoning services.* It includes

- concept satisfiability,
- concept subsumption,

endocarditis is an inflammation of the lining of the heart and the heart valves, and *pancarditis* is an inflammation of the entire heart.

[8]Except for range restriction, which we will discuss in detail in Chapter 4

- instance checking,
- consistency checking,
- classification,
- concept hierarchy computation,
- instance retrieval, and
- realization.

The first three reasoning services are *local*, in the sense that they perform on particular concepts and/or instances w.r.t. axioms in the ontology. On the contrary, the remaining reasoning services are *global*, in the sense that they always take into account all concepts and/or instances occurring in the ontology. As we shall see later, a global reasoning service could be realized by a finite number of calls to local reasoning services.

In this section, we first give formal definitions of these inference problems and then show some intuition w.r.t. our example ontology $\mathcal{O}_{\mathsf{med}}$.

Definition 9 (Satisfiability). Let $\mathcal{O}$ be an ontology and C, D concept descriptions. Then, C is *satisfiable w.r.t.* $\mathcal{O}$ if there exists a model $\mathcal{I}$ of $\mathcal{O}$ such that $C^{\mathcal{I}} \neq \emptyset$. Otherwise, C is *unsatisfiable w.r.t.* $\mathcal{O}$.

Two concepts C and D are said to be *disjoint w.r.t.* $\mathcal{O}$ if their conjunction $C \sqcap D$ is unsatisfiable w.r.t. $\mathcal{O}$. ◇

Note that it is sometimes relevant to consider concept satisfiability w.r.t. the empty ontology, i.e., consider the concept description alone. In such a case, we consider an arbitrary interpretation that makes C non-empty. For example, all concept names occurring in $\mathcal{O}_{\mathsf{med}}$ are satisfiable w.r.t. the ontology. In fact, it is not hard to see that an interpretation with two domain elements can satisfy all concept names by properly partitioning them. An example of an unsatisfiable concept w.r.t. $\mathcal{O}_{\mathsf{med}}$ is $\mathsf{Endocarditis} \sqcap \mathsf{Endocardium}$. Again, it is not hard to see that, for any models $\mathcal{I}$ of $\mathcal{O}_{\mathsf{med}}$, $\mathsf{Endocarditis}^{\mathcal{I}}$ and $\mathsf{Endocardium}^{\mathcal{I}}$ are disjoint, and thus their intersection is the empty set.

Definition 10 (Subsumption). Let $\mathcal{O}$ be an ontology and C, D concept descriptions. Then, D *subsumes* C *w.r.t.* $\mathcal{O}$ (written, $\mathcal{O} \models C \sqsubseteq D$ or $C \sqsubseteq_{\mathcal{O}} D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{O}$. We call C a *subsumee* or *subconcept*, and D a *subsumer* or *superconcept*.

D *strictly subsumes* C *w.r.t.* $\mathcal{O}$ (written, $\mathcal{O} \models C \sqsubset D$ or $C \sqsubset_{\mathcal{O}} D$) if, and only if, $C \sqsubseteq_{\mathcal{O}} D$, but not vice versa. If C and D subsume each other w.r.t. $\mathcal{O}$, i.e., $C \sqsubseteq_{\mathcal{O}} D$ and $D \sqsubseteq_{\mathcal{O}} C$, then we say that C and D are *equivalent w.r.t.* $\mathcal{O}$ (written, $\mathcal{O} \models C \equiv D$ or $C \equiv_{\mathcal{O}} D$). ◇

If concept definitions or inclusions are allowed, then we can restrict our attention to subsumption between concept *names*. To be more precise, $C \sqsubseteq_{\mathcal{O}} D$ if, and only if, $A \sqsubseteq_{\mathcal{O} \cup \{A \equiv C, B \equiv D\}} B$ if, and only if, $A \sqsubseteq_{\mathcal{O} \cup \{A \sqsubseteq C, D \sqsubseteq B\}} B$, where A, B are fresh concept names not occurring in $\mathcal{O}$. We shall see in the next chapter how this idea can be realized to support complex subsumption queries. In our example, it holds that $\mathsf{HeartDisease}$ subsumes $\mathsf{Endocarditis}$ and $\mathsf{Pancarditis}$ w.r.t. $\mathcal{O}_{\mathsf{med}}$. A more entangled example is that the following concept description:

$$\mathsf{Inflammation} \sqcap \exists \mathsf{has\text{-}location.HeartWall} \sqcap \exists \mathsf{causative\text{-}agent.Virus}$$

is subsumed by $\exists\mathsf{has\text{-}state.NeedsTreatment}$ w.r.t. $\mathcal{O}_{\mathsf{med}}$. To see this, note that $\mathsf{Inflammation}$ implies $\mathsf{Disease}$, and $\exists\mathsf{has\text{-}location.HeartWall}$ implies $\exists\mathsf{has\text{-}location}.\exists\mathsf{part\text{-}of.Heart}$. By the right-identity axiom α_{14}, the latter implies $\exists\mathsf{has\text{-}location.Heart}$. This, together with $\mathsf{Disease}$, fulfills the definition of $\mathsf{HeartDisease}$. Now, it is obvious to see that the left hand side of our subsumption implies the left hand side of α_{11}, thus also $\exists\mathsf{has\text{-}state.NeedsTreatment}$.

In DLs without (any form of) negation such as $\mathcal{EL}$, concept satisfiability is not interesting since any concept descriptions are satisfiable.[9] Having the bottom concept, however, (un)satisfiability and subsumption are inter-reducible in $\mathcal{EL}^+$. On the one hand, a concept description C is unsatisfiable w.r.t. $\mathcal{O}$ if, and only if, $C \sqsubseteq_{\mathcal{O}} \bot$. On the other hand, $C \sqsubseteq_{\mathcal{O}} D$ if, and only if, A is unsatisfiable w.r.t. $\mathcal{O}$ extended with GCIs $A \sqsubseteq C$ and $A \sqcap D \sqsubseteq \bot$ with A a fresh concept name.[10]

In case the ontology has an assertional component, thus individuals, we might be interested in knowing implicit relationship between these individuals and concepts. With the next standard reasoning service, this can be done:

Definition 11 (Instance checking). Let $\mathcal{O}$ be an ontology, C a concept description, r a role name, and a, b individuals. Then, the individual a (the pair of individuals (a, b), resp.) is *an instance of C (r, resp.) w.r.t. $\mathcal{O}$* (written, $\mathcal{O} \models C(a)$ ($\mathcal{O} \models r(a, b)$, resp.)) if, and only if, $a^{\mathcal{I}} \in C^{\mathcal{I}}$ ($(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$, resp.) for all models $\mathcal{I}$ of $\mathcal{O}$. ◇

So far, we have considered local inference problems, in the sense that they concern a particular concept (pair of concepts) and/or an individual. In the following, we formally introduce standard reasoning services that concern the whole ontology (i.e., all concepts and/or individuals):

Definition 12 (Consistency). Let $\mathcal{O}$ be an ontology. Then, $\mathcal{O}$ is *consistent* if, and only if, it has a model; otherwise, it is *inconsistent*. ◇

Consistency checking is a coarse-grained way to logically detect an error in the ontology. If the ontology is inconsistent, i.e., does not admit a model, then by definition any logical consequences vacuously follow. Thus, the ontology is meaningless. In our example, $\mathcal{O}_{\mathsf{med}}$ is consistent since it admits at least the aforementioned model with two domain elements. However, though an ontology is consistent, it may still contain local errors, such as, unsatisfiable concept names or unintended subsumption relationships. For this reason, consistency checking alone is not sufficient to support the developers in ontology design and maintenance.

[9] In fact, any $\mathcal{EL}$ concept descriptions C are satisfied by any interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with $A^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $r^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, for all concept names A and role names r occurring in C.

[10] In more expressive DLs with negation, the mentioned subsumption is reduced to asking whether $C \sqcap \neg D$ is unsatisfiable (see, e.g., [BN07]).

A much more interesting reasoning service, which is based on concept subsumption, therefore comes into play—ontology classification:

Definition 13 (Classification). Let $\mathcal{O}$ be an ontology and $\mathsf{CN}(\mathcal{O})$ the set of concept names occurring in $\mathcal{O}$. Then, *classification* of $\mathcal{O}$ is the identification of subsumptions between all pairs of concept names in $\mathcal{O}$, i.e., for all $A, B \in \mathsf{CN}(\mathcal{O})$, determine whether or not $A \sqsubseteq_{\mathcal{O}} B$. ◇

It immediately follows from the definition that classification can be done by at most a quadratic number of calls to the subsumption procedure. However, several heuristic and optimization techniques can be employed to reduce the number of subsumption calls by avoiding redundant calls and re-using old and told subsumption information. We will discuss these optimizations in more detail in Section 4.3. Apart from speeding up classification by reducing the number of subsumption calls, modern DL systems often represent classification results in a so-called *directed acyclic graph (DAG)*, where a directed edge links a concept name (an equivalence class of concept names) to an *immediate* subsumer (an equivalence class of immediate subsumers). We will later refer to this graph as the *concept hierarchy*.

Definition 14 (Concept hierarchy). Let $\mathcal{O}$ be an ontology and $\mathsf{CN}(\mathcal{O})$ the set of concept names occurring in $\mathcal{O}$. Then, the *concept hierarchy* (or *subsumption hierarchy*) of $\mathcal{O}$ is the most compact representation $(\mathsf{CN}(\mathcal{O}), \prec)$ of the partial ordering $(\mathsf{CN}(\mathcal{O}), \sqsubseteq_{\mathcal{O}})$ induced by the subsumption relation w.r.t. $\mathcal{O}$. ◇

A partial ordering is a reflexive, transitive, and antisymmetric relation. Obviously, the subsumption relation is a partial ordering with $\equiv_{\mathcal{O}}$ the equivalence relation. The relation $\prec$ denotes the immediate subsumption relation, which is the smallest relation such that its reflexive-transitive closure is identical to $\sqsubseteq_{\mathcal{O}}$. Precisely, $A \prec B$ if, and only if, (i) $A \sqsubseteq_{\mathcal{O}} B$; (ii) $A \not\equiv_{\mathcal{O}} B$; and (iii) there is no X such that $A \not\equiv_{\mathcal{O}} X$, $B \not\equiv_{\mathcal{O}} X$, and $A \sqsubseteq_{\mathcal{O}} X \sqsubseteq_{\mathcal{O}} B$. Intuitively, concept hierarchy is the most space-efficient way to represent classification results, dispensing with the non-immediate subsumptions which can be easily computed by reflexive-transitive closure. Most, if not all, DL systems indeed support output in this compact format. In our example, we have that $\mathsf{Endocarditis} \prec \mathsf{HeartDisease} \prec \mathsf{Disease}$, but $\mathsf{Endocarditis} \not\prec \mathsf{Disease}$. Considering the whole ontology, the concept hierarchy of $\mathcal{O}_{\mathsf{med}}$ can be depicted as in Figure 4.4 on page 67.

Stemmed from instance checking, it is natural to consider the following service: given an ontology and a concept name, retrieve all those individuals in the ontology that are instances of the concept name. This reasoning service is known as *instance retrieval*:

Definition 15 (Instance retrieval). Let $\mathcal{O}$ be an ontology, and A a concept name in $\mathcal{O}$. Then, the *instance retrieval* problem for A in $\mathcal{O}$ is the computation of all individuals a from $\mathsf{Ind}(\mathcal{O})$ such that $\mathcal{O} \models A(a)$. ◇

Conversely, the *realization* problem is, given an ontology and an individual, to determine all (most specific) concept names, to which the individual belongs:

Definition 16 (Realization). Let $\mathcal{O}$ be an ontology, and a an individual in $\mathcal{O}$. Then, the *realization* problem for a in $\mathcal{O}$ is the computation of all most specific concept names A from $\mathsf{CN}(\mathcal{O})$ such that $\mathcal{O} \models A(a)$, i.e., for every concept names $B \in \mathsf{CN}(\mathcal{O})$ with $B \sqsubset_{\mathcal{O}} A$, we have $\mathcal{O} \not\models B(a)$. $\diamond$

For each concept name $A \in \mathsf{CN}(\mathcal{O})$, instance retrieval gives us a set of instances belonging to A. Obviously, if we have $A \sqsubseteq_{\mathcal{O}} B$, then $\mathcal{O} \models A(a)$ implies $\mathcal{O} \models B(a)$ for any individual $a \in \mathsf{Ind}(\mathcal{O})$. In other words, if we extend the subsumption relation w.r.t. $\mathcal{O}$ to $\leq_{\mathcal{O}}$: $(\mathsf{CN}(\mathcal{O}) \cup \mathsf{Ind}(\mathcal{O})) \times (\mathsf{CN}(\mathcal{O}) \cup \mathsf{Ind}(\mathcal{O}))$ by including individuals and the instance-of relation, then $\leq_{\mathcal{O}}$ is a partial ordering. Precisely, $X \leq_{\mathcal{O}} Y$ holds if, and only if,

- $X = Y$, for $X, Y \in \mathsf{Ind}(\mathcal{O})$,
- $X \sqsubseteq_{\mathcal{O}} Y$, for $X, Y \in \mathsf{CN}(\mathcal{O})$, or
- $\mathcal{O} \models Y(X)$, for $X \in \mathsf{Ind}(\mathcal{O})$ and $Y \in \mathsf{CN}(\mathcal{O})$.

Based on the same idea for concept hierarchy, we can extract the smallest relation $\prec$ such that its reflexive-transitive closure is identical to $\leq_{\mathcal{O}}$. This relation, in conjunction with individuals and concept names, is called the *realization hierarchy* $(\mathsf{CN}(\mathcal{O}) \cup \mathsf{Ind}(\mathcal{O}), \prec)$ of $\mathcal{O}$. Similarly, this is the most compact representation of instance retrieval results, since for each individual a only the most-specific concept names, to which an individual a belongs, are kept. All other concept names, to which a belongs, can be easily computed by transitive closure from the realization hierarchy.

Classification (realization, resp.) results in the form of the concept (realization, resp.) hierarchy can be exploited by ontology editors to graphically display inferred results to the user. This way, the user may navigate the hierarchy by expanding or collapsing it from the top concept down to the bottom concept and individuals.

2.4 Supplemental Reasoning Services

The reasoning services defined in the previous section are supported by all state-of-the-art DL reasoners, including CEL [BLS06], FaCT [Hor98, TH06], HermiT [MSH07], KAON2 [Mot06], Pellet [SPC$^+$07], and RacerPro [HM01b].[11] As argued before, these reasoning services undoubtedly help facilitate ontology engineering tasks, in particular during the design and maintenance phases. As limitations of standard reasoning services in real-world applications have been addressed, new *non-standard* inference problems have emerged as important reasoning services (see, e.g., [Küs00, Bra06, Tur07] and [SC03, LW07, CHKS07, Sun08]).

Similar to writing large programs, building large-scale ontologies is an error-prone endeavor. Given an ontology under development, the ontology designer iteratively authors new concept definitions and adds them to the ontology. Without adequate knowledge of DLs or without enough attention to existing logical constraints and their potential interaction with newly added definitions, the addition could lead to undesired

[11]A list of DL reasoners: `http://www.cs.man.ac.uk/~sattler/reasoners.html`.

consequences. The ontology might be turned inconsistent, or certain concepts might become unsatisfiable. In the first case, any subsumption follows from the ontology, which makes it globally meaningless. In the latter case, the unsatisfiable concepts cannot instantiate individuals. Although this does not affect the whole ontology, it does not reflect the initial intention that concepts represent classes of individuals. Moreover, unsatisfiable concepts are equivalent to the bottom concept and thus can be unified. Even in the case that there is no logical inconsistency, an inferred subsumption might not be intended. As a real-world example, the subsumption relationship between ‘amputation of finger’ and ‘amputation of arm’ follows from SNOMED, which is clearly unintended [SBSS07, SMH07]. Standard reasoning of subsumption helps reveal such a modeling error. However, out of almost four hundred thousand concept definitions, it is virtually impossible to pinpoint the source of this error by hand.

To overcome this problem, some work on automating this process has recently been invested. In this section, we formally introduce the notions of justification and modularization, two supplemental reasoning services prominent in design and maintenance of large-scale ontologies.

2.4.1 Justification and debugging

Given a subsumption relationship or another questionable consequence (e.g., unsatisfiability or instantiation), justification computes a *minimal* subset (all *minimal* subsets) of the ontology that has this consequence. In what follows, we call this minimal set a *MinA*.[12]

Definition 17 (MinA). Let $\mathcal{O}$ be an ontology, and σ a logical entailment such that $\mathcal{O} \models \sigma$. Then, a subset $\mathcal{S} \subseteq \mathcal{O}$ is a *minimal axiom set (MinA) for σ w.r.t. $\mathcal{O}$* if, and only if, (i) $\mathcal{S} \models \sigma$, and (ii), for every $\mathcal{S}' \subset \mathcal{S}$, $\mathcal{S}' \not\models \sigma$. ◇

Intuitively, each axiom in a MinA is relevant to the entailment in question in the sense that the entailment no longer follows without it. In other words, axioms in a MinA are indispensable to retain the entailment. Note that MinAs need not be unique nor minimum relative to set cardinality. For example, consider the subsumption relationship $\sigma = (\mathsf{Endocarditis} \sqsubseteq \mathsf{HeartDisease})$ that holds in $\mathcal{O}_{\mathsf{med}}$. It is not hard to verify that the sets $\mathcal{S}_1 = \{\alpha_2, \alpha_3, \alpha_6, \alpha_8, \alpha_9, \alpha_{14}\}$ and $\mathcal{S}_2 = \{\alpha_2, \alpha_4, \alpha_6, \alpha_8, \alpha_9, \alpha_{14}\}$ are all the MinAs for σ w.r.t. $\mathcal{O}_{\mathsf{med}}$.

In the context of ontology justification, it is common to consider the dual problem: given a subsumption relationship or another questionable entailment (e.g., unsatisfiability or instantiation), compute a *maximal* subset (all *maximal* subsets) of the ontology that does *not* have this entailment. In what follows, we call this maximal set a *MaNA*.

Definition 18 (MaNA). Let $\mathcal{O}$ be an ontology, and σ a logical entailment such that $\mathcal{O} \models \sigma$. Then, a subset $\mathcal{S} \subseteq \mathcal{O}$ is a *maximal non-axiom set (MaNA) for σ w.r.t. $\mathcal{O}$* if, and only if, (i) $\mathcal{S} \not\models \sigma$, and (ii), for every $\mathcal{S}' \supset \mathcal{S}$, $\mathcal{S}' \models \sigma$. ◇

[12]A minimal set of axioms that has the entailment is sometimes called MUPS [SC03] or JUST [PSK05, KPHS07] in the literature.

Intuitively, given an ontology $\mathcal{O}$ and an unwanted entailment σ, a MaNA is a candidate for the new, revised ontology with σ suppressed. The set complement of a MaNA corresponds to a so-called *diagnosis* of an error [Rei87], i.e., a minimal *hitting set* of all the MinAs.

Definition 19 (Hitting set). Let U be a universal set, and $\mathbf{C} = \{S_1, \ldots, S_n\}$ a collection of subsets of U. A *hitting set H for* $\mathbf{C}$ is a subset of U such that $S_i \cap H \neq \emptyset$ for all $1 \leq i \leq n$. A hitting set H is *minimal* if there is no $H' \subset H$ such that H' is a hitting set for $\mathbf{C}$; and is *cardinality minimal* if there is no hitting set $H' \subset H$ with $|H'| < |H|$. $\diamond$

Definition 20 (Diagnosis). Let $\mathcal{O}$ be an ontology, and σ a logical entailment such that $\mathcal{O} \models \sigma$. Then, a subset $\mathcal{S} \subseteq \mathcal{O}$ is a *diagnosis for σ w.r.t. $\mathcal{O}$* if, and only if, (i) $\mathcal{O} \backslash \mathcal{S} \not\models \sigma$, and (ii), for every $\mathcal{S}' \subset \mathcal{S}$, $\mathcal{O} \backslash \mathcal{S}' \models \sigma$. $\diamond$

In DLs, if the ontology corresponds to the universal set U, and the set of all MinAs corresponds to the collection of subsets $\mathbf{C}$; then a minimal hitting set for $\mathbf{C}$ is a diagnosis.

Maximality of MaNAs (hence, minimality of diagnoses) ensures that changes required to be applied to the ontology are kept to the minimum. *Axiom pinpointing*, i.e., the computation of MinAs, can be seen as a first step toward automated explanation of an error, while the computation of MaNAs (equivalently, diagnoses) can be seen as a first step toward automated revision of a faulty ontology.

2.4.2 Modularization

Given an ontology $\mathcal{O}$ (recall that an ontology is a set of axioms), a module essentially is a subset of $\mathcal{O}$ that preserves a statement *of interest* or the statements involving symbols *of interest*. Here, *symbols* specifically refer to individuals, concept names, and role names. A set of interested symbols is called a *signature*. The signature of an ontology $\mathcal{O}$, denoted by $\mathsf{Sig}(\mathcal{O})$, is the disjoint union of the sets of concept names, role names and individuals occurring in the ontology $\mathcal{O}$, i.e., $\mathsf{CN}(\mathcal{O}) \cup \mathsf{RN}(\mathcal{O}) \cup \mathsf{Ind}(\mathcal{O})$. Similarly, we define the signature of x (written, $\mathsf{Sig}(x)$) with x a concept, role, individual or axiom, in an obvious way. We call a potential logical entailment introduced in Section 2.3 a *statement.*[13] A statement can be unsatisfiability, subsumption, equivalence, or instantiation that may or may not actually hold in the ontology.

Definition 21 (Module). Let $\mathcal{L}$ be a DL dialect, $\mathcal{O}$ an $\mathcal{L}$ ontology, and σ a statement formulated in $\mathcal{L}$. Then, an $\mathcal{O}' \subseteq \mathcal{O}$ is a *module for σ in $\mathcal{O}$* (for short, a *σ-module in $\mathcal{O}$*) whenever: $\mathcal{O} \models \sigma$ if, and only if, $\mathcal{O}' \models \sigma$.

We say that $\mathcal{O}'$ is a *module for a signature* $\mathbf{S}$ *in* $\mathcal{O}$ (for short, an $\mathbf{S}$*-module in* $\mathcal{O}$) if for every $\mathcal{L}$-statement σ with $\mathsf{Sig}(\sigma) \subseteq \mathbf{S}$, $\mathcal{O}'$ is a σ-module in $\mathcal{O}$. $\diamond$

[13]In [CHKS07, Sun08], the term ‘axiom’ was used both as a potential entailment and as an ontological axiom. We use different terms here in order to avoid unnecessary confusion.

Intuitively, a module in an ontology $\mathcal{O}$ is a subset $\mathcal{O}' \subseteq \mathcal{O}$ that preserves a statement of interest or the statements over a signature of interest. Observe that the definition is generic in the sense that the whole ontology is a module in itself. Nevertheless, such a module is uninteresting, since it does not give us new information concerning the ontology relative to the statement or the signature. In the following, we are interested in reasonably small modules that are relatively cheap to extract.

Given an ontology $\mathcal{O}$ with an entailment σ, i.e., $\mathcal{O} \models \sigma$, it is immediate that a MinA for σ w.r.t. $\mathcal{O}$ is a σ-module in $\mathcal{O}$. Moreover, it is a minimal module. In fact, a σ-module satisfies only the first of the two conditions for being a MinA for σ. Our MinAs $\mathcal{S}_1$ and $\mathcal{S}_2$ from the previous subsection are (minimal) modules for $\mathsf{Endocarditis} \sqsubseteq \mathsf{HeartDisease}$ in $\mathcal{O}_{\mathsf{med}}$. Unfortunately, a procedure for computing MinAs cannot be used effectively and efficiently to extract a module. First, MinAs are only defined for a consequence, i.e., a statement that follows from the ontology. Second, computation of MinAs is normally expensive and involve standard inference, such as subsumption.

In [CHKS07], a specific type of module based on *syntactic locality* has been introduced for the expressive DL $\mathcal{SHOIQ}$. The notion of syntactic locality naturally extends to $\mathcal{SROIQ}$ with the presence of (acyclic) role inclusions. We recap the definition of this module modulo the DL $\mathcal{EL}^+$ here and show a relationship between this and our module in Section 5.1.

Definition 22 (Locality-based modules). Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology, and $\mathbf{S}$ a signature. The following grammar recursively defines $\mathbf{Con}^\perp(\mathbf{S})$:

$$\mathbf{Con}^\perp(\mathbf{S}) ::= A^\perp \mid (C^\perp \sqcap C) \mid (C \sqcap C^\perp) \mid (\exists r.C^\perp) \mid (\exists r^\perp.C)$$

with r a role name, C a concept description, $A^\perp, r^\perp \notin \mathbf{S}$, and $C^\perp \in \mathbf{Con}^\perp(\mathbf{S})$.

An $\mathcal{EL}^+$ axiom α is *syntactically local* w.r.t. $\mathbf{S}$ if it is one of the following forms: *(1)* RI $R^\perp \sqsubseteq s$ where $R^\perp$ is either a role name $r^\perp \notin \mathbf{S}$ or a role composition $r_1 \circ \cdots \circ r_n$ with $r_i \notin \mathbf{S}$ for some $i \leq n$, or *(2)* GCI $C^\perp \sqsubseteq C$ where $C^\perp \in \mathbf{Con}^\perp(\mathbf{S})$. We write $\mathsf{local}(\mathbf{S})$ to denote the collection of all $\mathcal{EL}^+$ axioms that are syntactically local w.r.t. $\mathbf{S}$.

If $\mathcal{O}$ can be partitioned into $\mathcal{O}_1$ and $\mathcal{O}_2$ such that every axiom in $\mathcal{O}_2$ is syntactically local w.r.t. $\mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_1)$, then $\mathcal{O}_1$ is a *locality-based module* for $\mathbf{S}$ in $\mathcal{O}$. We say that $\mathcal{O}_1$ is minimal if there is no $\mathcal{O}_1' \subset \mathcal{O}_1$ that is a locality-based module for $\mathbf{S}$ in $\mathcal{O}$. ◇

In [CHKS08], it has been shown that, given an ontology $\mathcal{O}$ and a signature $\mathbf{S}$, there always exists a *unique*, minimal locality-based module for $\mathbf{S}$ in $\mathcal{O}$, denoted by $\mathcal{O}^{\mathsf{loc}}_{\mathbf{S}}$.

In the following, we define an important property of a module in order to be used in optimization in axiom pinpointing.

Definition 23 (Subsumption module). Let $\mathcal{O}$ be an ontology, and A a concept name occurring in $\mathcal{O}$. Then, an $\mathcal{O}' \subseteq \mathcal{O}$ is a *subsumption module for A in $\mathcal{O}$* whenever: $A \sqsubseteq_{\mathcal{O}} B$ if, and only if, $A \sqsubseteq_{\mathcal{O}'} B$ holds for all concept names B occurring in $\mathcal{O}$.

The subsumption module $\mathcal{O}'$ for A in $\mathcal{O}$ is called *strong* if the following holds for all concept names B occurring in $\mathcal{O}$: if $A \sqsubseteq_{\mathcal{O}} B$, then every MinA for $A \sqsubseteq_{\mathcal{O}} B$ is a subset of $\mathcal{O}'$. ◇

Obviously, $\mathcal{O}$ itself is a strong subsumption module for every concept name A occurring in $\mathcal{O}$. In Section 5.1, we introduce a definition of a specific type of module that not only enjoys the nice property of strong subsumption modularity but is also typically quite small compared to the whole ontology.

Chapter 3

A Case Study: Snomed ct

Now that all the essential reasoning problems have been identified and formally defined in the previous chapter, it should be appropriate to see how they fit in real-world application scenarios. As already motivated in Introduction, the Description Logic (DL) dialect under consideration, as well as fragments and slight extensions thereof, has been and still is used as the logical underpinning of several biomedical ontologies, most of which are of large scale. Due to the need for scalability of design and maintenance, it is important that reasoning support can be achieved in reasonable runtime if doable at all. This chapter presents a case study of the use of the Description Logic $\mathcal{EL}^+$ in Snomed ct, which should hopefully shed some light on the usefulness of logic-based design and maintenance of large-scale biomedical ontologies.

3.1 The Systematized Nomenclature of Medicine

The Systematized Nomenclature of Medicine, Clinical Terms (Snomed ct) is a comprehensive clinical and medical ontology that covers a wide range of concepts in the domain [SPSW01, Spa05]. It was produced by merging Snomed Reference Terminology (rt) [SCC97, Spa00, Rec07] with Clinical Terms version 3 (CTV3) [OPR95]. Snomed rt was developed by the College of American Pathologists (CAP) with the aim to be a comprehensive clinical reference terminology for the retrieval and analysis of data relating to the causes of diseases, the treatment of patients, or even retrieval of health care information (e.g., medical literature search) [SCC97]. The rt version was the first generation of the Snomed terminology to use the formal semantics (through the KRSS syntax [PSS93]) of the Description Logic. This was mainly to address multiple hierarchical is-a relationships which had been undesirably indirect in the earlier version (Snomed III). In 1993, the UK National Health Service (NHS) has adopted the Read Codes, which had been developed by a medical practitioner Read, for health electronic records. Later on, the terminology has been greatly expanded and enhanced to have become Clinical Terms version 3 (CTV3). Between 1999 and 2002, the UK NHS and CAP, along with Keiser Permanente, jointly worked to merge CTV3 and Snomed rt. The key feature of the merge was the comprehensiveness of Snomed ct, with 55% of the source concepts from CTV3 only and 31% from rt only. After the merge, the ontology has become freely available in both the US and UK.

In 2007, the International Health Terminology Standards Development Organisation (IHTSDO) has been founded with the purpose to internationalize and standardize SNOMED CT as the reference clinical terminology among the member countries [IHT07]. Nowadays, SNOMED CT is freely available to all of the (currently nine) IHTSDO member countries and has been licensed in several other countries as well. Additionally, IHTSDO gives out the ontology under 'affiliate license' for academic and research purposes.

SNOMED RT and SNOMED CT have been developed with the help of the Description Logic $\mathcal{EL}$ (see Definition 1 on page 19) which can be seen as an unfoldable TBox (see Definition 3 on page 20) extended with role inclusions.[1] An example of role inclusion that occurs in the ontology is

$$\textsf{causative-agent} \sqsubseteq \textsf{associated-with}$$

A concept in SNOMED CT is called *primitive* if it is defined with a primitive concept definition, and *fully defined* if it has a full concept definition.[2] Examples of full and primitive concept definitions are in order:

$$\begin{array}{rcl} \textsf{AmputationOfFinger} & \equiv & \textsf{HandExcision} \sqcap \\ & & \exists\textsf{roleGroup}.(\ \exists\textsf{direct-procedure-site}.\textsf{Finger}_S \sqcap \\ & & \qquad\qquad\quad \exists\textsf{method}.\textsf{Amputation}\ ) \\ \textsf{Finger}_S & \sqsubseteq & \textsf{DigitOfHand}_S \sqcap \textsf{Hand}_P \end{array}$$

Two interesting remarks concerning these examples are as follow. First, SNOMED CT purposefully uses the special role roleGroup to group two or more existential quantifications in a definition [SDMW02]. In our example, the method of amputation is grouped with finger structure as the direct procedure site. In some definitions, there may be more than one such group, and without explicitly grouping them by roleGroup, existential quantifications may inadvertently interact in an undesired way. Spackman et al. have given an example of 'Tetralogy of Fallot' in [SDMW02]:

$$\begin{array}{rcllll} \textsf{TetralogyOfFallot} & \equiv & \exists\textsf{rG}.(\exists\textsf{s}.\textsf{RightVentricle} & \sqcap & \exists\textsf{m}.\textsf{Hypertrophy}) & \sqcap \\ & & \exists\textsf{rG}.(\exists\textsf{s}.\textsf{Aorta} & \sqcap & \exists\textsf{m}.\textsf{Overriding}) & \sqcap \\ & & \exists\textsf{rG}.(\exists\textsf{s}.\textsf{PulmonaryValve} & \sqcap & \exists\textsf{m}.\textsf{Stenosis}) & \sqcap \\ & & \exists\textsf{rG}.(\exists\textsf{s}.\textsf{InterventricularSeptum} & \sqcap & \exists\textsf{m}.\textsf{IncompleteClosure}) & \end{array}$$

where rG, s and m are abbreviations for roles roleGroup, site and morphology, respectively. Obviously, if the preceding roleGroup existentials were dropped, any arbitrary combination of morphology and site would be possible. For instance, if we query for subconcepts of $\exists\textsf{s}.\textsf{PulmonaryValve} \sqcap \exists\textsf{m}.\textsf{IncompleteClosure}$, the modified definition of Tetralogy (without roleGroup) is unintendedly retrieved as an answer. Second, SNOMED CT uses multiple codes for each anatomical entity (e.g., hand) to represent

[1]In SNOMED CT lingo, this is known as the 'stated form' as opposed to the 'distributed form.'

[2]Note that the notion of 'primitive concept name' here is different from that in, e.g., [Baa03, Sun05b]. In the latter papers, primitive concept names are those that do not occur on the left-hand side of a definition.

the structure (Hand_S), the entity (Hand_E) and the part (Hand_P)—the encoding known as SEP-triplet (see Subsection 3.2.1 below).[3] Then, the primitive concept definition given above expresses that the finger is part of the hand.

As of January/2005 release, SNOMED CT contains 13 role inclusions, 38 719 concept definitions and 340 972 primitive concept definitions (see Section 6.1 for more detail).

For the rest of the thesis, SNOMED and SNOMED CT are used interchangeably to refer to the SNOMED, Clinical Terms. The next section discusses issues of SEP-triplet encoding in SNOMED CT among others, and Section 3.3 proposes a modeling paradigm for the ontology by using the full machinery of $\mathcal{EL}^+$.

3.2 Issues in SNOMED CT

In [SSB07], we have outlined an array of existing issues in SNOMED CT both from ontologist's and logician's perspectives. Some of the issues can be solved by upgrading the underpinning logical formalism to the Description Logic $\mathcal{EL}^+$ and by exploiting DL reasoning support, whereas some others require ontological analysis under scrutiny. In this section, we focus attention to those issues that logical reasoning based on the DL $\mathcal{EL}^+$ can solve or, to a lesser degree, alleviate.

Individuals are missing in SNOMED CT, while numerous SNOMED concepts (such as Europe, Germany and Thailand) intuitively stand for unique individuals. Instead of formulating these individuals as *instances* of the concept GeographicLocation, in SNOMED CT they are simply concepts (thus, classes of individuals) subsumed by GeographicLocation.

Top-level categories in SNOMED CT are concept names that mimic meta-classes in the sense that they categorize other concept names by means of subsumption. For instance, the concept Niece is a subconcept of the concept SpecialConcept which is obviously incorrect from the ontological point of view. In SNOMED CT, the concept SpecialConcept is one of the 18 top-level categories that are used to segment the large ontology into disjointly categorized hierarchies. Undesirably, however, there are a few concept names that belong to more than one top-level category.

Relations or roles in SNOMED CT are not constrained in the sense that certain roles should only be applied to certain concepts (i.e., domain and range restrictions). Even though no observable glitches may be detected in the ontology itself, the use of the ontology as a background knowledge base where users may author additional definitions (incorporate post-coordinated concepts into an extension of SNOMED CT) could violate this implicit modeling discipline and thus introduce errors. Without an appropriate logical restriction on the applicability of a role, such errors are hard or may even be impossible to detect.

Part-of relation is not in use in the current versions of SNOMED CT. Instead, it formulates a part-whole relation, for instance, between Finger and Hand by means

[3] Though this is employed in a rather incomplete way.

of the subsumption relation and the so-called SEP-triplet. This method unnecessarily overloads the is-a hierarchy, increases the size of the anatomy portion of the ontology, and causes confusion for ontology developers [Pat06, SBSS07].

Right-identity is used in Snomed ct [Spa00] to express the fact that "an 'active ingredient' of a 'direct substance' is a 'direct substance'." The role inclusion:

$$\mathsf{direct\text{-}substance} \circ \mathsf{has\text{-}active\text{-}ingredient} \sqsubseteq \mathsf{direct\text{-}substance}$$

could correctly capture the intuition of the fact if the role grouping technique was not employed. With role grouping, existential quantifications in Snomed ct are preceded by an existential quantification over roleGroup, for instance, the definitions of AmputationOfFinger and TetralogyOfFallot. Particularly, the role has-active-ingredient will always be preceded by roleGroup in any model of the ontology, blocking a potential role chain as in the left-hand side of the right-identity rule from happening. As a consequence, three subsumption relationships that are supposed to hold in the presence of the right-identity rule cannot be inferred from the ontology.

For the rest of the present section, we will discuss SEP-triplet encoding and its use in Snomed ct.

3.2.1 SEP-Triplet encoding

The example of primitive concept definition above is part of an SEP-triplet. Precisely, an SEP-triplet for A is composed of three concepts: the entity A, the part A_P and the structure A_S, which are linked to each other by is-a and part-of relationships as shown below:

$$\begin{tikzcd} & A_S & \\ A \arrow[ur] & & A_P \arrow[ul] \arrow[ll, dashed, "\mathsf{part\text{-}of}"] \end{tikzcd}$$

$$\begin{aligned} A &\sqsubseteq A_S \\ A_P &\sqsubseteq A_S \sqcap \exists \mathsf{part\text{-}of}.A \end{aligned}$$

The SEP-triplet can be expressed by two (primitive) concept definitions as shown on the right of the graph representation. Intuitively, the E-concept denotes the whole *entity* of A, the P-concept denotes any proper *part* of A, and finally the S-concept denotes the *structure* (which includes both the entity and any part) of A.

This encoding technique has been introduced by Schulz et al. in [SRH98] to address the problem of logical reasoning without transitivity on roles. They proposed a workaround method by embedding the aggregation hierarchy (part-of relation) in the generalization hierarchy (is-a or subsumption relation) with the help of auxiliary concepts.[4] In order to specify, for instance, that finger is part of hand, an is-a link from the S-concept of Finger to the P-concept of Hand is added. To illustrate the transitivity effect over the part-of relation, another such relationship

[4]The subsumption relation is inherently transitive and could be handled by the early knowledge base reasoner LOOM [MB87] which was used in the experiments described in [SH01].

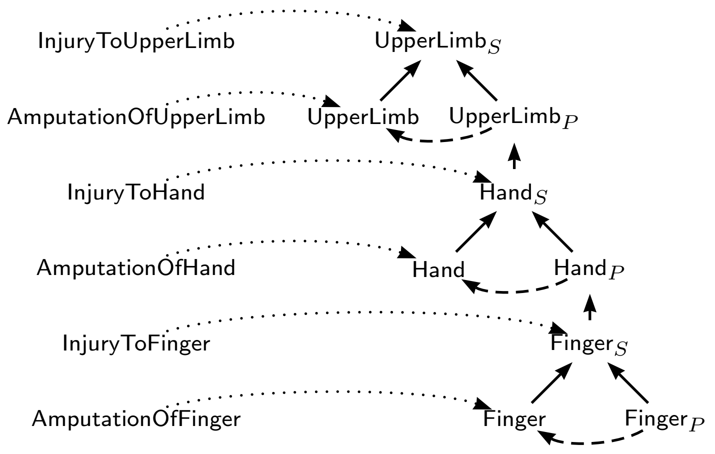


Figure 3.1: Examples of *idealized* SEP-triplets in Snomed ct.

is considered: hand is part of upper limb. Figure 3.1 visualizes the three complete SEP-triplets (representing three anatomical concepts) and the aggregation relationships among them. In the figure, the solid and dashed edges denote subsumption and part-of relationship, respectively; while the dotted edges represent the relation has-location that link either clinical findings or medical procedures to body parts. Again, the graph representation can be formulated in an $\mathcal{EL}$ TBox shown in Figure 3.2. It is easy to see that transitivity of part-of can be simulated through the intra-triplet part-of relationships and the intrinsic transitivity of (both intra- and inter-triplet) subsumption relationships. Formally, it can be inferred that $\mathsf{Finger} \sqsubseteq \mathsf{Finger}_S \sqsubseteq \mathsf{Hand}_P \sqsubseteq \mathsf{Hand}_S \sqsubseteq \mathsf{UpperLimb}_P \sqsubseteq \exists \mathsf{part\text{-}of}.\mathsf{UpperLimb}$.

Nonetheless, there are several problems with this transitivity workaround from both ontological and practical viewpoints. First of all, resolving the aggregation hierarchy (i.e., the hierarchy induced by part-of relationships) by means of the generalization hierarchy (i.e., the hierarchy induced by subsumption relationships) is dangerous since the two relationships are completely different by nature [Pat06]. In general, characteristics are not inherited along the aggregation hierarchy, but are so along the generalization hierarchy. There are, however, some characteristics for which propagation along the part-of hierarchy is desirable. In our example, the concept InjuryToFinger is supposed to be classified as InjuryToHand and thus InjuryToUpperLimb, therefore it is sensible to allow for propagation of the property 'Injury to' along the aggregation hierarchy. On the contrary, the property 'Amputation of' should never propagate in this way because it would be incorrect to say that 'an amputation of finger is an amputation of upper limb.' In [SRH98], it has been proposed to choose between the S-concepts whenever such inheritance is desired, and E-concepts otherwise (see Figure 3.1). Even though this approach works, it is rather confusing and error-prone as indicated by a number of problems found in Snomed ct.

First of all, it has indeed turned out that is-a links in Snomed ct can be ambiguous, i.e., it is not always clear whether they are introduced as part of the SEP-triplet

$\mathsf{UpperLimb}$	$\sqsubseteq$	$\mathsf{UpperLimb}_S$
$\mathsf{UpperLimb}_P$	$\sqsubseteq$	$\mathsf{UpperLimb}_S \sqcap \exists \mathsf{part\text{-}of}.\mathsf{UpperLimb}$
Hand_S	$\sqsubseteq$	$\mathsf{UpperLimb}_P$
Hand	$\sqsubseteq$	Hand_S
Hand_P	$\sqsubseteq$	$\mathsf{Hand}_S \sqcap \exists \mathsf{part\text{-}of}.\mathsf{Hand}$
Finger	$\sqsubseteq$	Hand_P
Finger	$\sqsubseteq$	Finger_S
Finger_P	$\sqsubseteq$	$\mathsf{Finger}_S \sqcap \exists \mathsf{part\text{-}of}.\mathsf{Finger}$
$\mathsf{InjuryToUpperLimb}$	$\equiv$	$\mathsf{Injury} \sqcap \mathsf{has\text{-}location}.\mathsf{UpperLimb}_S$
$\mathsf{InjuryToHand}$	$\equiv$	$\mathsf{Injury} \sqcap \mathsf{has\text{-}location}.\mathsf{Hand}_S$
$\mathsf{InjuryToFinger}$	$\equiv$	$\mathsf{Injury} \sqcap \mathsf{has\text{-}location}.\mathsf{Finger}_S$
$\mathsf{AmputationOfUpperLimb}$	$\equiv$	$\mathsf{Amputation} \sqcap \mathsf{has\text{-}location}.\mathsf{UpperLimb}$
$\mathsf{AmputationOfHand}$	$\equiv$	$\mathsf{Amputation} \sqcap \mathsf{has\text{-}location}.\mathsf{Hand}$
$\mathsf{AmputationOfFinger}$	$\equiv$	$\mathsf{Amputation} \sqcap \mathsf{has\text{-}location}.\mathsf{Finger}$

Figure 3.2: An $\mathcal{EL}$ TBox representing the SEP-triplets and referring concepts.

approach or are supposed to represent a genuine generalization relationship. Second, the SEP-triplet approach is error-prone since it works correctly only if it is employed with a very strict modeling discipline. In Snomed, triplets are often modeled in an incomplete way, in particular, the P-concept and the part-of link to it from the E-concept are missing in most cases. In addition, the auxiliary S-concept is often used as if it were a proper entity class. This mistake has resulted in many *false positive* subsumption relationships that hold in the ontology due to unintended inheritance of all characteristics along the aggregation hierarchy. Third, the approach introduces for every proper concept in (the anatomy portion of) the ontology two auxiliary concepts which results in a drastic increase in the ontology size. To add to the confusion, the same term (essentially a string attached to each concept name) often ambiguously denotes both Snomed S-concepts and E-concepts, e.g., the terms 'Finger' and 'Finger structure' are being allowed synonyms for the concept Finger_S.

3.3 A Modeling Paradigm for Snomed ct Using $\mathcal{EL}^+$

One of the most prominent decisions to be made when modeling an ontology is the concept description and ontology language. As presented in Chapter 2, there are numerous DL dialects ranging from inexpressive sub-Boolean logics to highly expressive ones with complex constructors like quantified number restriction and inverse roles. Different DLs are suitable for different application domains and have different complexity levels. Usually, the more expressive the logic gets, the higher the complexity goes [Zol07]. For instance, the DL $\mathcal{ALC}$ has an ExpTime-complete subsumption problem when GCIs are admitted [Sch91] (see also [Don07]), while subsumption in $\mathcal{EL}^+$ is polynomial (shown for its superlogic $\mathcal{EL}^{++}$ in [BBL05]). This means that reasoning with the full machinery of $\mathcal{ALC}$ can potentially take time exponential in the size of the input—the ontology. On the one hand, it is important to take good care of complexity of reasoning when the language is chosen, especially given the typically large scale of biomedical ontologies. On the other hand, it is not less important to ensure that essential aspects of the ontology can be properly expressed in the language chosen. Most of the issues raised in the previous section are examples of underspecification of concept definitions which stem from using the overly inexpressive DL $\mathcal{EL}$.

To this end, we present a modeling paradigm for Snomed ct using the DL $\mathcal{EL}^+$ for two reasons:

- $\mathcal{EL}^+$ is proved tractable both from the theoretical [BBL05] and practical [BLS07] points of view, with readily available DL systems (see, e.g., [BLS06]). Tractability of the underlying ontology language is a key to scalability of reasoning.
- The machinery in $\mathcal{EL}^+$ is sufficient to address the issues in Section 3.2 (particularly, the SEP-triplet encoding). By appropriately modeling constraints (such as pairwise disjointness of top-level categories), previously invisible modeling errors can now be automatically detected by a reasoner.

In the next subsection, a re-engineering approach for Snomed ct is proposed with emphasis on removing SEP-triplets. In Subsection 3.3.2, a number of usage scenarios of reasoning techniques presented in this thesis will be presented which shall demonstrate usability and usefulness of reasoning support in the process of design and maintenance of Snomed ct.

3.3.1 Re-engineering Snomed ct

The main tenets of the SEP-triplet encoding are twofold: first, it helps mimic reasoning with transitivity on the part-of relation; and second, it allows certain properties to propagate along the aggregation hierarchy. Now, we propose an alternative approach to these reasoning patterns by specifying them in a more direct and concise way [SBSS07]. In what follows, let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology (see Definition 6 on page 22) illustrating the transformation from the ontology in Figure 3.2 to one without SEP-triplets.

Since the DL $\mathcal{EL}^+$ directly supports transitivity on roles (see Table 2.3 on page 23), reasoning with transitivity of the part-of role can be done by simply adding the axiom transitive(part-of) to $\mathcal{O}$. Moreover, we suggest to model a subrole proper-part-of $\sqsubseteq$ part-of which is also transitive. The distinction between the two partitive roles is that part-of is reflexive, while proper-part-of is not. With this distinction, the P-concept and S-concept of a triplet can be simply reformulated, respectively, by the following concept definitions:

$$\begin{aligned}\mathsf{Hand}_P &\equiv \exists\mathsf{proper\text{-}part\text{-}of.Hand}\\ \mathsf{Hand}_S &\equiv \exists\mathsf{part\text{-}of.Hand}\end{aligned}$$

whenever they are required.[5] Though no explicit intra-triplet subsumption relationships are specified, the triplet structure is retained, namely, $\mathsf{Hand}_P \sqsubseteq_{\mathcal{O}} \mathsf{Hand}_S$ (since

[5]Note that our modeling technique does not require the triplets, but this is presented in case that backward compatibility becomes an issue.

$\mathsf{proper\text{-}part\text{-}of} \sqsubseteq \mathsf{part\text{-}of} \in \mathcal{O}$) and $\mathsf{Hand} \sqsubseteq_{\mathcal{O}} \mathsf{Hand}_S$ (since $\mathsf{Hand} \sqsubseteq_{\mathcal{O}} \exists\mathsf{part\text{-}of.Hand}$ due to reflexivity of part-of). Optionally, a role inclusion axiom (or a left-identity rule)[6]

$$\mathsf{part\text{-}of} \circ \mathsf{proper\text{-}part\text{-}of} \sqsubseteq \mathsf{proper\text{-}part\text{-}of}$$

can be used to retain inter-triplet structures, e.g., the subsumption relationships $\mathsf{Finger}_S \sqsubseteq \mathsf{Hand}_P$ and $\mathsf{Hand}_S \sqsubseteq \mathsf{UpperLimb}_P$.

To state that "finger is part of hand" and "hand is part of upper limb," the concept inclusions $\mathsf{Finger} \sqsubseteq \exists\mathsf{proper\text{-}part\text{-}of.Hand}$ and $\mathsf{Hand} \sqsubseteq \exists\mathsf{proper\text{-}part\text{-}of.UpperLimb}$ are added to the ontology $\mathcal{O}$.[7] By the semantics of transitivity and role hierarchy (see Table 2.2 on page 20), it implies that $\mathsf{Finger} \sqsubseteq_{\mathcal{O}} \exists\mathsf{proper\text{-}part\text{-}of.UpperLimb}$ and $\mathsf{Finger} \sqsubseteq_{\mathcal{O}} \exists\mathsf{part\text{-}of.UpperLimb}$ as required.

The other rationale of SEP-triplets in SNOMED CT is to allow certain properties to propagate along the aggregation hierarchy [SRH98]. It is realized in SEP-triplet encoding by using two target concepts for a single anatomical organ, for instance, Finger_S and Finger for finger. The same property (e.g., has-location) may point to either S or E concepts depending on whether it is to propagate along the aggregation hierarchy or not. This is rather not intuitive and error-prone as confirmed by the fact that AmputationOfFinger is linked to Finger_S in SNOMED CT.

For getting rid of SEP-triplets, we propose to use two 'has location' relations: has-location which is allowed to propagate along the aggregation hierarchy and used in the definition of, e.g., InjuryToFinger; and has-exact-location which is not allowed to propagate that way and is used in the definition of, e.g., AmputationOfFinger. To enable the effect of propagation along the aggregation hierarchy, we use the following right-identity rule [Spa00, SBSS07]:

$$\mathsf{has\text{-}location} \circ \mathsf{proper\text{-}part\text{-}of} \sqsubseteq \mathsf{has\text{-}location}$$

Here, proper-part-of is said to be a right-identity of has-location. Intuitively, any occurrence of proper-part-of is absorbed to the right of the role has-location in the sense that, e.g., $\exists\mathsf{has\text{-}location}.\exists\mathsf{proper\text{-}part\text{-}of.Hand}$ is subsumed by $\exists\mathsf{has\text{-}location.Hand}$. Putting together we obtain a re-engineered portion of SNOMED CT shown in Figure 3.3 that corresponds to the cumbersome modeling method with SEP-triplets (see Figure 3.2 on page 38).

In addition to using role inclusions to replace SEP-triplets, some other machinery of $\mathcal{EL}^+$ could be used to alleviate difficulties during the modeling process of SNOMED CT. Some of the more important issues are discussed in order:

- The eighteen top-level categories are intended to be pairwise disjoint. The disjointness however is not logically specified as axioms in the ontology. Thus, it cannot be automatically detected if supposedly disjoint hierarchies happen to

[6]This, together with $\mathsf{proper\text{-}part\text{-}of} \sqsubseteq \mathsf{part\text{-}of}$, forms a cycle over role inclusions which is not allowed in $\mathcal{SROIQ}$ and thus OWL 2. Fortunately, such a cyclic dependency does not cause any trouble in our setting with the DL $\mathcal{EL}^+$.

[7]These inclusions are indeed primitive concept definitions and thus can be incorporated into existing definitions of Finger and Hand, respectively, if any.

$$
\begin{array}{rcll}
\mathsf{Finger} & \sqsubseteq & \mathsf{BodyPart} \sqcap \exists \mathsf{proper\text{-}part\text{-}of}.\mathsf{Hand} & (1) \\
\mathsf{Hand} & \sqsubseteq & \mathsf{BodyPart} \sqcap \exists \mathsf{proper\text{-}part\text{-}of}.\mathsf{UpperLimb} & (2) \\
\mathsf{UpperLimb} & \sqsubseteq & \mathsf{BodyPart} & (3) \\
\mathsf{AmputationOfFinger} & \equiv & \mathsf{Amputation} \sqcap \exists \mathsf{has\text{-}exact\text{-}location}.\mathsf{Finger} & (4) \\
\mathsf{AmputationOfHand} & \equiv & \mathsf{Amputation} \sqcap \exists \mathsf{has\text{-}exact\text{-}location}.\mathsf{Hand} & (5) \\
\mathsf{AmputationOfUpperLimb} & \equiv & \mathsf{Amputation} \sqcap \exists \mathsf{has\text{-}exact\text{-}location}.\mathsf{UpperLimb} & (6) \\
\mathsf{InjuryToFinger} & \equiv & \mathsf{Injury} \sqcap \exists \mathsf{has\text{-}location}.\mathsf{Finger} & (7) \\
\mathsf{InjuryToHand} & \equiv & \mathsf{Injury} \sqcap \exists \mathsf{has\text{-}location}.\mathsf{Hand} & (8) \\
\mathsf{InjuryToUpperLimb} & \equiv & \mathsf{Injury} \sqcap \exists \mathsf{has\text{-}location}.\mathsf{UpperLimb} & (9) \\
\mathsf{proper\text{-}part\text{-}of} \circ \mathsf{proper\text{-}part\text{-}of} & \sqsubseteq & \mathsf{proper\text{-}part\text{-}of} & (10) \\
\mathsf{proper\text{-}part\text{-}of} & \sqsubseteq & \mathsf{part\text{-}of} & (11) \\
\mathsf{part\text{-}of} \circ \mathsf{part\text{-}of} & \sqsubseteq & \mathsf{part\text{-}of} & (12) \\
\epsilon & \sqsubseteq & \mathsf{part\text{-}of} & (13) \\
\mathsf{part\text{-}of} \circ \mathsf{proper\text{-}part\text{-}of} & \sqsubseteq & \mathsf{proper\text{-}part\text{-}of} & (14) \\
\mathsf{has\text{-}exact\text{-}location} & \sqsubseteq & \mathsf{has\text{-}location} & (15) \\
\mathsf{has\text{-}location} \circ \mathsf{proper\text{-}part\text{-}of} & \sqsubseteq & \mathsf{has\text{-}location} & (16)
\end{array}
$$

Figure 3.3: A re-engineered portion of SNOMED CT dispensing with SEP-triplets.

overlap. Our empirical analysis with the January/2005 release of SNOMED CT demonstrates evidence of this problem. In fact, the classification of the ontology extended with pairwise disjointness axioms, e.g.,

$$\mathsf{ClinicalFinding} \sqcap \mathsf{BodyStructure} \sqsubseteq \bot$$

reveals that more than one hundred concepts become unsatisfiable due to the fact that they belong to more than one top-level category.

- Certain roles in SNOMED CT are designed to be used to connect concepts in a certain category to those in another, as observed by Zhang et al. in [ZPTI06]. An example for this modeling discipline is that concepts in the category ClinicalFinding have a relationship findingSite with concepts in category BodyStructure. This can directly be translated to domain and range restrictions as follows:

$$
\begin{array}{rcl}
\mathsf{domain}(\mathsf{findingSite}) & \sqsubseteq & \mathsf{ClinicalFinding} \\
\mathsf{range}(\mathsf{findingSite}) & \sqsubseteq & \mathsf{BodyStructure}
\end{array}
$$

  Domain and range restrictions, in conjunction with the disjointness assertions above, help to automatically detect any indisciplined role usage.[8] Unlike the case with disjointness constraints, adding domain and range restrictions did not detect any modeling flaw in SNOMED CT. However, such restrictions are helpful when a user (who may not be aware of the implicit modeling discipline) extends the ontology or post-coordinates concept descriptions.

[8]The syntactic restriction on range restrictions and role inclusions (see Definition 8 on page 22) needs to be satisfied in order to ensure tractability of reasoning.

- If role grouping is indispensable, it has to be taken into account when modeling role inclusion axioms. More precisely, the only right-identity rule in Snomed ct has to be rewritten to:

  $$\mathsf{direct\text{-}substance} \circ \mathsf{roleGroup} \circ \mathsf{has\text{-}active\text{-}ingredient} \sqsubseteq \mathsf{direct\text{-}substance}$$

  which revives the missing three subsumptions and possibly other subsumptions between post-coordinated concept descriptions. Likewise, additional role inclusion axioms proposed in the new modeling paradigm (i.e., the axioms 10, 12, 14 and 16 in Figure 3.3 on page 41) need to be rewritten accordingly.

3.3.2 Usage scenarios of reasoning support

One of the most prominent benefits of using a Description Logic as the underlying modeling language is its automated reasoning. This subsection discusses usage scenarios of the reasoning services introduced in Section 2.3 and 2.4 that hopefully support the tedious and error-prone process of designing and maintaining large-scale ontologies like Snomed ct.

Satisfiability and consistency check whether there are any *logical* inconsistencies in the ontology. An unsatisfiable concept reveals a modeling error since it is equivalent to the bottom concept which cannot be instantiated. In $\mathcal{EL}^+$, the only source of concept unsatisfiability is by violating disjointness axioms which has been shown to be useful in the previous subsection when pairwise disjointness is asserted among the top-level categories in Snomed ct.

If the ontology is inconsistent, it does not admit any model and anything logically follows from it. Obviously, this is undesirable and needs to be detected and removed before any subsequent design and maintenance tasks can be carried out. In $\mathcal{EL}^+$, it is possible though uncommon to encounter such a global inconsistency. In most cases, inconsistencies involve the assertional component (ABox) of the ontology in such a way that an unsatisfiable concept is instantiated by an individual.

Subsumption and classification automatically infer specialization–generalization relationships between two or more concepts. Unlike unsatisfiability and inconsistency, subsumption (except special cases $A \sqsubseteq_{\mathcal{O}} \bot$ and $\top \sqsubseteq_{\mathcal{O}} A$) does not indicate a logical error within the ontology. It however may indicate unintuitive or unintended relationships that should not hold. An example that does actually happen in Snomed ct is the subsumption $\mathsf{AmputationOfFinger} \sqsubseteq_{\mathcal{O}} \mathsf{AmputationOfHand}$ [BS08]. Without automated reasoning support, it is not easy to detect such a faulty subsumption relationship.

Classification computes subsumption relationships between all pairs of concept names occurring in the ontology. Moreover, all concept names are arranged in a lattice (i.e., directed acyclic graph) based on their subsumption relationships. Such a concept lattice can not only be used for ontology navigation and visualization but potentially also for the ontology 'distribution normal form,' in which only the most specific subsuming concept names are given in concept definitions [Spa01].

Instance retrieval and realization involve scenarios where SNOMED CT is used as a background knowledge in data-rich applications. An example is a knowledge-aware health information system in a hospital in which patient data, treatment records, etc. are stored in an ABox. Instance checking and retrieval can then be used to semantically retrieve relevant information from the data in the ABox and the background knowledge in the TBox. For example, a general practitioner currently treating pulmonary heart disease may be interested in retrieving related previous treatment records, for instance, retrieval of instances of the concept $\mathsf{PatientRecord} \sqcap \exists \mathsf{has\text{-}symptom.Dyspnoea} \sqcap \exists \mathsf{has\text{-}symptom.ChestPain} \sqcap \exists \mathsf{has\text{-}family\text{-}record.HeartDisease}$. On the other hand, given a patient record with symptoms, family background and social context information, realization could help in medical diagnoses in the sense that it automatically computes potential diseases the patient may have.

Justification helps the ontology developer to understand *why*, for instance, one concept is subsumed by another. The 'amputation' example given above obviously is counter-intuitive and thus should be removed. In order to do that, the developer has to find the culprit for it, i.e., the minimal set of axioms that are responsible for the subsumption. Given almost four hundred thousand axioms in SNOMED CT, it is practically impossible to find the culprit by hand. Automated reasoning support of explanation can help by computing a minimal subset (all minimal subsets) of the ontology that have the consequence of interest. The ontology developer then has to revise under scrutiny only a few relevant axioms instead of the whole ontology to eliminate the unwanted consequence.

Modularization is particularly useful for ontology re-use and segmentation. Consider the scenario where a medical company wants to model an ontology about its pharmaceutical products. To reduce modeling time and to reuse existing knowledge, it decides to import a *relevant* portion of SNOMED CT. Before the import actually takes place, SNOMED CT has to be modularized based on the concepts (and possibly also roles) of interest—in this case, those pharmaceutical products the company has and wants to model. The extracted module of the ontology is guaranteed to be relatively small while still capturing all relevant information required.

Besides, modularization of all concept names appearing in the ontology helps reveal semantic dependencies in and structures of the ontology.

More usage scenarios of logical reasoning support for design and maintenance of ontologies have been discussed in [LBF$^+$06, LLS06, CGG$^+$07].

Chapter 4

Techniques for Standard Reasoning

In this chapter, we describe several algorithms for deciding standard DL inference problems as defined in Section 2.3. As pointed out before, satisfiability can be reduced to subsumption and subsumption is by definition a sub-problem of classification, so it suffices to have a classification algorithm. Section 4.1 describes a polynomial-time classification algorithm that, given an ontology, makes explicit the subsumption relationships between all pairs of concept names occurring in it. Since the core algorithm works on an ontology in normal form that does not contain range restrictions, we explain how an $\mathcal{EL}^+$ ontology can be transformed into a subsumption-preserving one in the required format in Subsection 4.1.1 and 4.1.2. The last two subsections discuss several techniques employed in the implementation and optimization of the algorithm in the CEL reasoner. Although the classification algorithm can be used to answer subsumption queries, it is not efficient since it also identifies all other subsumption relationships. We develop a decision procedure that is directed by the target subsumption in Section 4.2 and investigate a few methods for constructing a concept hierarchy in Section 4.3. Then, we propose in Section 4.4 a pragmatic approach to incremental classification that is useful not only for certain scenario of incremental reasoning but also for querying complex subsumptions. Finally, techniques for reasoning with individuals (thus, ABox) are discussed in Section 4.5.

4.1 Classifying an $\mathcal{EL}^+$ Ontology

A polynomial-time algorithm for classification in $\mathcal{EL}$ with general concept inclusions and role hierarchy axioms has been first proposed in [Bra04a], and shortly afterward this algorithm has been enhanced and extended to the much more powerful DL $\mathcal{EL}^{++}$ in the "Pushing the $\mathcal{EL}$ Envelope" paper [BBL05] by Baader et al. Very recently, the border of the tractable DL has been pushed yet further to support reflexivity and range restriction on roles [BBL08] (the syntactic restriction in Definition 8 on page 22 applied). Here, we consider the slightly less expressive DL $\mathcal{EL}^+$ introduced in the previous chapter and describe a *refined* version of the polytime classification algorithm for implementation purposes. The queue techniques used in the refined classification algorithm are mainly inspired by the algorithm for satisfiability of propositional Horn formulas proposed in [DG84].

Both in tableau-based DL systems and in earlier DL systems based on structural subsumption algorithms, classification and the concept hierarchy is computed by performing multiple subsumption tests. In addition to optimizing the algorithm for testing individual subsumption, such systems can also be optimized by minimizing the number of subsumption tests needed to classify the ontology and to construct the concept hierarchy [BHN+92, BHN+94, Hor97]. In contrast, our classification algorithm simultaneously computes the subsumption relationships between *all* pairs of concept names occurring in the input ontology.

The classification algorithm presented below does not treat range restrictions as universal quantifiers in the way the standard tableau-based algorithm for expressive DLs does,[1] but rather eliminates them in quadratic time in such a way that all (non)subsumptions are preserved. Only after the elimination is completed, the core algorithm (i.e., the version that does not support range restrictions) can be applied. Also, the core classification algorithm accepts an ontology in normal form, so at first the input ontology has to be transformed in linear time into a (non)subsumption preserving one in normal form. Normalization and elimination of range restrictions can naturally be seen as preprocessing steps of the overall classification procedure.[2]

4.1.1 Normalization

Given an $\mathcal{EL}^+$ ontology $\mathcal{O}$, we write $\mathsf{CN}(\mathcal{O})$ to denote the set of concept names occurring in $\mathcal{O}$, $\mathsf{CN}^\top(\mathcal{O})$ to denote $\mathsf{CN}(\mathcal{O})$ augmented with the top concept, and $\mathsf{CN}^\bot(\mathcal{O})$ to denote $\mathsf{CN}(\mathcal{O})$ augmented with the bottom concept. Likewise, $\mathsf{RN}(\mathcal{O})$ denotes the set of role names occurring in $\mathcal{O}$.

Definition 24 (Normal form). An $\mathcal{EL}^+$ ontology $\mathcal{O}$ is in *normal form* if the following three conditions are satisfied:

1. all concept inclusions in $\mathcal{O}$ have one of the following forms:

$$\begin{array}{lrcl} \textbf{GCI1} & A_1 \sqcap \cdots \sqcap A_n & \sqsubseteq & B, \\ \textbf{GCI2} & A_1 & \sqsubseteq & \exists r.A_2, \\ \textbf{GCI3} & \exists r.A_1 & \sqsubseteq & B \end{array}$$

   where $A_i \in \mathsf{CN}^\top(\mathcal{O})$ and $B \in \mathsf{CN}^\bot(\mathcal{O})$;

2. all role inclusions in $\mathcal{O}$ have one of the following forms:

$$\begin{array}{lrcl} \textbf{RI1} & \epsilon & \sqsubseteq & r, \\ \textbf{RI2} & r & \sqsubseteq & s, \\ \textbf{RI3} & r \circ s & \sqsubseteq & t \end{array}$$

   where $r, s, t \in \mathsf{RN}(\mathcal{O})$;

[1] In expressive DLs, the range restriction $\mathsf{range}(r) \sqsubseteq C$ can be viewed as a GCI $\top \sqsubseteq \forall r.C$.

[2] This is precisely what the CEL reasoner does when an ontology is loaded to the system. See [Sun05a] for the CEL reference manual.

3. there are neither reflexivity statements, transitivity statements nor domain restrictions, and all range restrictions are of the form $\mathsf{range}(r) \sqsubseteq A$ with A a concept name.

◇

Each domain restriction can be rewritten as a concept inclusion, while reflexivity and transitivity statements are simply special cases of role inclusions. By introducing new concept and role names, any $\mathcal{EL}^+$ ontology $\mathcal{O}$ can be turned into a normalized ontology $\widetilde{\mathcal{O}}$ that is a *conservative extension* of $\mathcal{O}$, i.e., every model of $\widetilde{\mathcal{O}}$ is also a model of $\mathcal{O}$, and every model of $\mathcal{O}$ can be extended to a model of $\widetilde{\mathcal{O}}$ by appropriately choosing the interpretations of the additional concept and role names. As a consequence of conservativity, we have that $C \sqsubseteq_{\mathcal{O}} D$ if, and only if, $C \sqsubseteq_{\widetilde{\mathcal{O}}} D$ for all concept descriptions C, D that are constructed from concept and role names occurring in $\mathcal{O}$.

The transformation can be performed in linear time in the size of $\mathcal{O}$ and introduces linearly many new concept and role names. Figure 4.1 shows the *normalization rules* that are to be repeatedly applied to axioms in the ontology to obtain new axioms in normal form according to Definition 24. Here, a *rule application* means that the axiom on the left-hand side of the rule is replaced by the axiom(s) on the right-hand side. Observe that Rule **NR1-6** simply removes the left-hand concept inclusion by replacing it with nothing. We need to partition the rule applications into three phases:

1. exhaustive applications of Normalization Rule **NR1-1** to **NR1-7**;
2. exhaustive applications of Normalization Rule **NR2-1** to **NR2-3**;
3. exhaustive applications of Normalization Rule **NR3-1** to **NR3-3**,

in order to ensure a linear termination of the transformation.[3] If all the rules are exhaustively applied in an arbitrary order, the size of the resulting normalized ontology may blow up quadratically in the size of the original ontology (see, e.g., Example 39 in [Sun05b]).

Lemma 25 (Normalization is linear and preserves subsumption). *Normalization of an $\mathcal{EL}^+$ ontology $\mathcal{O}$ runs in linear time in the size of $\mathcal{O}$ and yields another $\mathcal{EL}^+$ ontology $\widetilde{\mathcal{O}}$ in normal form whose size is linear in that of $\mathcal{O}$. Moreover, $\widetilde{\mathcal{O}}$ is a conservative extension of $\mathcal{O}$.*

Proof.

Each of Normalization Rule **NR1-1** to **NR1-5** can be applied at most once for each axiom of the forms shown on the left-hand side, and each application generates at most two axioms of linear size. Analogously, the last two rules in the first phase may be applied at most once to each concept inclusion in which the bottom concept occurs. Let $\mathcal{O}'$ be the result of exhaustive rule applications in the first phase. It is readily shown that the first phase requires linear number of rule applications, that each rule application takes linear time, and that $|\mathcal{O}'|$ is linear in $|\mathcal{O}|$.

[3]In fact, the first two phases could be combined without affecting termination in linear time. The resulting normalized ontology however may contain some irrelevant axioms due to tardy elimination of axioms of the form $C^{\perp} \sqsubseteq D$.

NR1-1	$\mathsf{domain}(r) \sqsubseteq C$	$\leadsto$	$\exists r.\top \sqsubseteq C$
NR1-2	$\mathsf{range}(r) \sqsubseteq C$	$\leadsto$	$\mathsf{range}(r) \sqsubseteq A,\ A \sqsubseteq C$
NR1-3	$\mathsf{reflexive}(r)$	$\leadsto$	$\epsilon \sqsubseteq r$
NR1-4	$\mathsf{transitive}(r)$	$\leadsto$	$r \circ r \sqsubseteq r$
NR1-5	$C \equiv D$	$\leadsto$	$C \sqsubseteq D,\ D \sqsubseteq C$
NR1-6	$C^{\perp} \sqsubseteq D$	$\leadsto$	
NR1-7	$C \sqsubseteq D^{\perp}$	$\leadsto$	$C \sqsubseteq \perp$
NR2-1	$r_1 \circ \cdots \circ r_k \sqsubseteq s$	$\leadsto$	$r_1 \circ \cdots \circ r_{k-1} \sqsubseteq u,\ u \circ r_k \sqsubseteq s$
NR2-2	$C_1 \sqcap \cdots \sqcap \hat{C} \sqcap \cdots \sqcap C_n \sqsubseteq D$	$\leadsto$	$\hat{C} \sqsubseteq A,\ C_1 \sqcap \cdots \sqcap A \sqcap \cdots \sqcap C_n \sqsubseteq D$
NR2-3	$\exists r.\hat{C} \sqsubseteq D$	$\leadsto$	$\hat{C} \sqsubseteq A,\ \exists r.A \sqsubseteq D$
NR3-1	$\hat{C} \sqsubseteq \hat{D}$	$\leadsto$	$\hat{C} \sqsubseteq A,\ A \sqsubseteq \hat{D}$
NR3-2	$B \sqsubseteq \exists r.\hat{C}$	$\leadsto$	$B \sqsubseteq \exists r.A,\ A \sqsubseteq \hat{C}$
NR3-3	$B \sqsubseteq C \sqcap D$	$\leadsto$	$B \sqsubseteq C,\ B \sqsubseteq D$

where r, r_i, s denote role names, $k > 2$, C, C_i, D arbitrary concept descriptions, $C^{\perp}$ a concept description in which $\perp$ occurs, $D^{\perp}$ a complex concept description in which $\perp$ occurs, $\hat{C}, \hat{D}$ complex concept descriptions, B an atomic concept description. Moreover, we write u to denote a *fresh* role name, and A a *fresh* concept name, respectively.

Figure 4.1: Normalization rules.

Rule **NR2-1** is applied at most once for each occurrence of "$\circ$" in the role composition. Such an application increases the size of $\mathcal{O}'$ by a constant, introducing a new role name u and splitting the role inclusion to two. Similarly, the numbers of applications of Rule **NR2-2** and **NR2-3** are bounded by the number of occurrences of "$\sqcap$" and "$\exists$", respectively. Again, such an application increases the ontology size by a constant, introducing a new concept name A and splitting the concept inclusion to two. Therefore, the second phase runs in linear time and produces an ontology $\mathcal{O}''$ of size linear in $|\mathcal{O}'|$.

Rule **NR3-1** is applicable at most once for each concept inclusion in $\mathcal{O}''$, and an application of this rule takes a constant time and increases the size by a constant. Analogous to the second phase, **NR3-2** and **NR3-3** can be applied at most once for each occurrence of "$\exists$" and "$\sqcap$" in $\mathcal{O}''$. Each application of **NR3-2** increases the size by a constant, and so does an application of **NR3-2** (since B is atomic). So, the last phase of normalization runs in linear time and produces an ontology $\widetilde{\mathcal{O}}$ in normal form whose size is linear in $|\mathcal{O}''|$, thus also linear in $|\mathcal{O}|$.

To prove the second part of the lemma (i.e., to show that $\widetilde{\mathcal{O}}$ is a conservative extension of $\mathcal{O}$), it suffices to show that each normalization rule is model-preserving in the sense that any model of the rhs axiom(s) is also a model of the lhs axiom, and that any model of the lhs axiom can be extended to a model of the rhs axioms. This is obvious for the case of **NR1-1**, **NR1-3**, **NR1-4** and **NR1-5**, since they are simply syntactic variants. For **NR1-2**, let $\mathcal{I}$ be a model of $\{\mathsf{range}(r) \sqsubseteq A,\ A \sqsubseteq C\}$. It implies immediately by the semantics that $\mathcal{I} \models \mathsf{range}(r) \sqsubseteq C$. The other direction also holds by interpreting A as the interpretation of C. For **NR1-6** and **NR1-7**, it suffices to show that $C^{\perp}$ and $D^{\perp}$ are equivalent to $\perp$. But, this is vacuously the case by the semantics of concept constructs in $\mathcal{EL}^+$ and the fact that $\perp$ occurs in $C^{\perp}$ and $D^{\perp}$.

Model conservativity of all the other rules, apart from **NR3-3**, can be shown in the same fashion as for **NR1-2** by properly interpreting the new role or concept name. The lhs and rhs of Rule **NR3-3** are equivalent by the semantics of conjunction. ❏

It immediately follows that subsumption (thus, classification) w.r.t. an $\mathcal{EL}^+$ ontology can be reduced in linear time to the same problem w.r.t. the normalized ontology. Therefore, we may assume without loss of generality that a given $\mathcal{EL}^+$ ontology is in normal form. Before the core classification algorithm can be described, we need to perform one additional step to the normalized ontology, that is, to eliminate the range restrictions in normal form.

4.1.2 Reducing out range restrictions

As mentioned earlier in this section, our $\mathcal{EL}^+$ algorithm does not handle a range restriction $\mathsf{range}(r) \sqsubseteq A$ as an equivalent inclusion $\top \sqsubseteq \forall r.A$ like the way the standard tableau-based algorithm does. Instead, it reduces out the range assertion $\mathsf{range}(r) \sqsubseteq A$ by asserting the concept name A directly into each existential quantification $\exists r.C$ on the right-hand side of a concept inclusion. In the following, we formally describe this reduction and state a lemma.

Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology in normal form. For each role name $r \in \mathsf{RN}(\mathcal{O})$, $\mathsf{range}_{\mathcal{O}}(r)$ denotes the set of concept names A such that $\mathcal{O} \models \mathsf{range}(r) \sqsubseteq A$, i.e., there exists a superrole s of r w.r.t. $\mathcal{O}$ such that $\mathsf{range}(s) \sqsubseteq A \in \mathcal{O}$. To assert range restrictions on r into existential quantifications, we introduce a fresh concept name $X_{r,D}$ for every *normalized* GCI $C \sqsubseteq \exists r.D$ in $\mathcal{O}$. Intuitively, $X_{r,D}$ denotes those individuals in the model that satisfy both D and the range of r. We define the new ontology $\mathcal{O}'$ from $\mathcal{O}$ by removing all range restrictions and performing the following operations:

1. exchange each GCI $C \sqsubseteq \exists r.D$ in $\mathcal{O}$ for the following GCIs
$$C \sqsubseteq \exists r.X_{r,D}, \quad X_{r,D} \sqsubseteq D, \quad X_{r,D} \sqsubseteq A \text{ for all } A \in \mathsf{range}_{\mathcal{O}}(r);$$
2. for each role name r with $\epsilon \sqsubseteq r \in \mathcal{O}$, add the GCI $\top \sqsubseteq A$ for all $A \in \mathsf{range}_{\mathcal{O}}(r)$.

In [BBL08], the following result has been shown:

Lemma 26 (Range elimination preserves subsumption). *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology in normal form, and $\mathcal{O}'$ the resulting ontology from the abovementioned method. For all concept names $A_0, B_0 \in \mathsf{CN}(\mathcal{O})$, we have*
$$A_0 \sqsubseteq_{\mathcal{O}} B_0 \quad \textit{if, and only if,} \quad A_0 \sqsubseteq_{\mathcal{O}'} B_0.$$

Proof. Both directions are proved by showing the contrapositive. The "$\Leftarrow$" direction is trivial since every (counter) model $\mathcal{I}$ of $\mathcal{O}$ can be extended to a (counter) model $\mathcal{I}'$ of $\mathcal{O}'$ by interpreting each of the new concept names $X_{r,D}$ as $\{d \in D^{\mathcal{I}} \mid$ there exists an e such that $(e,d) \in r^{\mathcal{I}}$ $\}$. Obviously, $\mathcal{I}'$ satisfies all the new GCIs.

For the "$\Rightarrow$" direction, let $A_0 \not\sqsubseteq_{\mathcal{O}'} B_0$. Then, there is a model $\mathcal{I}'$ of $\mathcal{O}'$ with a witness individual $w \in A_0^{\mathcal{I}'} \backslash B_0^{\mathcal{I}'}$. Since $\mathcal{I}'$ need not satisfy the range restrictions in $\mathcal{O}$, $\mathcal{I}'$ is not necessarily a model of $\mathcal{O}$. To fix this problem, r-edges that violate the range restrictions are removed from $\mathcal{I}'$. Precisely, we construct a new interpretation $\mathcal{I}$ such that $\mathcal{I}$ and $\mathcal{I}'$ share the same domain and agree on the interpretation of concept names A and role names r with $\mathsf{range}_{\mathcal{O}}(r) = \emptyset$. It modifies the interpretation of all other roles as follows:

$$r^{\mathcal{I}} := \left\{ (d,e) \in r^{\mathcal{I}'} \mid e \in \bigcap_{A \in \mathsf{range}_{\mathcal{O}}(r)} A^{\mathcal{I}'} \right\}.$$

Since we do not change the interpretation of concept names, it is the case that $w \in A_0^{\mathcal{I}} \backslash B_0^{\mathcal{I}}$. It remains to show that $\mathcal{I}$ is indeed a model of $\mathcal{O}$. We consider only axioms in $\mathcal{O}$ that are potentially affected by the modification:

- Let $C \sqsubseteq \exists r.D \in \mathcal{O}$. Then, $\mathcal{O}'$ contains the GCIs $C \sqsubseteq \exists r.X_{r,D}$ and $X_{r,D} \sqsubseteq D$. Let $d \in C^{\mathcal{I}}$. Since C is a concept name, $d \in C^{\mathcal{I}'} \subseteq (\exists r.X_{r,D})^{\mathcal{I}'}$. Hence, there must be an $e \in \Delta^{\mathcal{I}}$ such that $(d,e) \in r^{\mathcal{I}'}$ and $e \in X_{r,D}^{\mathcal{I}'} \subseteq D^{\mathcal{I}'}$. Since D is a concept name, $e \in D^{\mathcal{I}}$. By reduction, there is a GCI $X_{r,D} \sqsubseteq A$ in $\mathcal{O}'$ for all $A \in \mathsf{range}_{\mathcal{O}}(r)$. Thus, $e \in A^{\mathcal{I}'}$ which together with $(d,e) \in r^{\mathcal{I}'}$ implies that $(d,e) \in r^{\mathcal{I}}$. It follows that $d \in (\exists r.D)^{\mathcal{I}}$ as required.

- Let $\exists r.C \sqsubseteq D \in \mathcal{O}$. Then, this GCI is also in $\mathcal{O}'$. Since $(\exists r.C)^{\mathcal{I}'} \subseteq D^{\mathcal{I}'}$ and $r^{\mathcal{I}} \subseteq r^{\mathcal{I}'}$, we have $(\exists r.C)^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ as required.

- Let $\epsilon \sqsubseteq r \in \mathcal{O}$. Then, it is also in $\mathcal{O}'$. Let $d \in \Delta^{\mathcal{I}}$. Then, $(d,d) \in r^{\mathcal{I}'}$. By reduction, there is a GCI $\top \sqsubseteq A$ in $\mathcal{O}'$ for all $A \in \mathsf{range}_{\mathcal{O}}(r)$. Thus, we have that $d \in A^{\mathcal{I}'}$ for all concept names A, and the definition of $\mathcal{I}$ yields that $(d,d) \in r^{\mathcal{I}}$.

- Let $r \sqsubseteq s \in \mathcal{O}$. Then, it is also in $\mathcal{O}'$. Since $r \sqsubseteq s \in \mathcal{O}$, we have $\mathsf{range}_{\mathcal{O}}(s) \subseteq \mathsf{range}_{\mathcal{O}}(r)$. By definition of $\mathcal{I}$, this together with $r^{\mathcal{I}'} \subseteq s^{\mathcal{I}'}$ yields $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$.

- Let $r \circ s \sqsubseteq t \in \mathcal{O}$. Then, it is also in $\mathcal{O}'$. Let $(d,e) \in r^{\mathcal{I}}$ and $(e,f) \in s^{\mathcal{I}}$. Then, $(d,e) \in r^{\mathcal{I}'}$, $(e,f) \in s^{\mathcal{I}'}$, and thus $(d,f) \in s^{\mathcal{I}'}$. The syntactic restriction (see Definition 8 on page 22) on $\mathcal{O}$ ensures that $\mathsf{range}_{\mathcal{O}}(t) \subseteq \mathsf{range}_{\mathcal{O}}(s)$. By the definition of $\mathcal{I}$, $(e,f) \in s^{\mathcal{I}}$ implies that $f \in \bigcap_{A \in \mathsf{range}_{\mathcal{O}}(s)} A^{\mathcal{I}'}$. But, it follows that $f \in \bigcap_{A \in \mathsf{range}_{\mathcal{O}}(t)} A^{\mathcal{I}'}$. This together with $(d,f) \in t^{\mathcal{I}'}$ implies that $(d,f) \in t^{\mathcal{I}}$ as required.

- Let $\mathsf{range}(r) \sqsubseteq A \in \mathcal{O}$ and $(d,e) \in r^{\mathcal{I}}$. By the definition of $\mathcal{I}$, $e \in A^{\mathcal{I}'} = A^{\mathcal{I}}$.

❏

Observe that the reduction induces a quadratic blowup in the size of the original ontology in normal form. Precisely, for each GCI $C \sqsubseteq \exists r.D$, the reduction potentially adds n new GCIs of the form $X_{r,D} \sqsubseteq A$ with n the number of all concept names A in $\mathcal{O}$. This however is the worst-case scenario which, we believe, does not happen in biomedical ontology applications.

Recall that we require the syntactic restriction to be fulfilled for an $\mathcal{EL}^+$ ontology (see Definition 8 on page 22). The elimination proposed in this subsection does not work in the general case where the syntactic restriction is lifted. In fact, subsumption w.r.t. an unrestricted $\mathcal{EL}^+$ ontology (see Definition 6 on page 22) is undecidable as shown in [BBL08]. Essentially, the machinery of range restrictions and role inclusions suffices to express the emptiness problem of the intersection of two context-free grammars, which is known to be undecidable [HU79].

Since the range elimination procedure generates only new concept inclusions of the form **GCI1** or **GCI2**, the resulting ontology remains in normal form and contains *no* range restrictions. For the rest of this chapter, we assume without loss of generality that a given $\mathcal{EL}^+$ ontology is a finite set of concept and role inclusions of the forms specified in Point 1 and 2 in Definition 24 on page 46.

4.1.3 An abstract classification algorithm

The $\mathcal{EL}$ family of Description Logics enjoys the nice algorithmic property that the main inference problem of subsumption can be decided in polynomial time in the size of the input ontology and concept descriptions. Speaking of the model-theoretic semantics, the tractability is a consequence of the following properties:

- there always exists a canonical model $\mathcal{I}$ of size *polynomial* in the size of the ontology $\mathcal{O}$, i.e., $A \sqsubseteq_{\mathcal{O}} B$ if, and only if, there is a model $\mathcal{I}$ of $\mathcal{O}$ with $|\mathcal{I}| \leq p(|\mathcal{O}|)$ and p a polynomial such that $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$; and
- such a canonical model can be constructed in a *deterministic* manner, i.e., no nondeterminism is involved in the model construction.

In other words, to decide subsumption in a DL in the $\mathcal{EL}$ family, we deterministically construct a polynomial-size canonical model and then simply check against this model whether or not the subsumption in question holds. Obviously, such the model construction can be performed in *deterministic polynomial* time.

The subsumption algorithm for $\mathcal{EL}^{++}$ proposed in [BBL05], which is an improved and extended version of the similar one for $\mathcal{ELH}$ with GCIs in [Bra04a], essentially constructs two mappings (one for subsumer sets and the other for role relations). These mappings form an edge-labeled directed graph that purposefully corresponds to the canonical model. In the scope of this thesis, we present the algorithm from [BBL05] modulo the DL $\mathcal{EL}^+$ and later refine it for efficient implementation purposes. To distinguish the former from the refined version, we call it the *abstract* classification algorithm. The refinement is discussed in the next subsection.

The abstract classification takes as input an $\mathcal{EL}^+$ ontology $\mathcal{O}$ (in normal form without range restrictions) and computes

CR1	If $A_1, \dots, A_n \in S(X)$, $A_1 \sqcap \dots \sqcap A_n \sqsubseteq B \in \mathcal{O}$, and $B \notin S(X)$ then $S(X) := S(X) \cup \{B\}$
CR2	If $A \in S(X)$, $A \sqsubseteq \exists r.B \in \mathcal{O}$, and $(X, B) \notin R(r)$ then $R(r) := R(r) \cup \{(X, B)\}$
CR3	If $(X, Y) \in R(r)$, $A \in S(Y)$, $\exists r.A \sqsubseteq B \in \mathcal{O}$, and $B \notin S(X)$ then $S(X) := S(X) \cup \{B\}$
CR4	If $(X, Y) \in R(r)$, $\bot \in S(Y)$, and $\bot \notin S(X)$ then $S(X) := S(X) \cup \{\bot\}$
CR5	If $(X, Y) \in R(r)$, $r \sqsubseteq s \in \mathcal{O}$, and $(X, Y) \notin R(s)$ then $R(s) := R(s) \cup \{(X, Y)\}$
CR6	If $(X, Y) \in R(r)$, $(Y, Z) \in R(s)$, $r \circ s \sqsubseteq t \in \mathcal{O}$, and $(X, Z) \notin R(t)$ then $R(t) := R(t) \cup \{(X, Z)\}$

Figure 4.2: Completion rules.

- a mapping S assigning to each element A of $\mathsf{CN}^\top(\mathcal{O})$ a subset $S(A)$ of $\mathsf{CN}(\mathcal{O}) \cup \{\top, \bot\}$, and
- a mapping R assigning to each element r of $\mathsf{RN}(\mathcal{O})$ a binary relation $R(r)$ over $\mathsf{CN}^\top(\mathcal{O})$.

Alternatively, we can view the mappings collectively as a *completion graph* $\mathcal{CG} = (V, E, \ell)$ with the set of vertices $V = \mathsf{CN}^\top(\mathcal{O})$, the set of edges $E = \{(A, r, B) \mid (A, B) \in R(r)\}$, and the vertex-labeling function $\ell = S$.

The intuition is that these mappings make implicit subsumption relationships explicit in the sense that

- $B \in S(A)$ implies $A \sqsubseteq_{\mathcal{O}} B$, and
- $(A, B) \in R(r)$ implies $A \sqsubseteq_{\mathcal{O}} \exists r.B$.

We write $\mathcal{O} \models \epsilon \sqsubseteq r$ if, and only if, there is an $s \in \mathsf{RN}(\mathcal{O})$ such that $\mathcal{O} \models s \sqsubseteq r$ and $\epsilon \sqsubseteq s \in \mathcal{O}$. The mappings are initialized by setting, for each $A \in \mathsf{CN}^\top(\mathcal{O})$, $S(A) := \{A, \top\}$; and, for each $r \in \mathsf{RN}(\mathcal{O})$,

$$R(r) := \begin{cases} \{(A, A) \mid A \in \mathsf{CN}^\top(\mathcal{O})\} & \text{if } \mathcal{O} \models \epsilon \sqsubseteq r; \\ \emptyset & \text{otherwise.} \end{cases}$$

Then the sets $S(\cdot)$ and $R(\cdot)$ are extended by exhaustively applying the completion rules shown in Figure 4.2 until no more rule applies. The following result has been shown in [BBL05]:

Theorem 27 (Correctness of the abstract algorithm). *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology in normal form without range restrictions. The abstract algorithm applied to $\mathcal{O}$*

terminates in time polynomial in the size of $\mathcal{O}$ and, after termination, we have, for all concept names A, B in $\mathcal{O}$, that

$$\{B, \bot\} \cap S(A) \neq \emptyset \textit{ if, and only if, } A \sqsubseteq_{\mathcal{O}} B.$$

In fact, we can straightforwardly extend this theorem to cover those subsumptions that involve the top and bottom concepts. While $A \sqsubseteq_{\mathcal{O}} \top$ and $\bot \sqsubseteq_{\mathcal{O}} A$ vacuously hold true in any ontology, subsumptions of the forms $\top \sqsubseteq_{\mathcal{O}} A$ and $A \sqsubseteq_{\mathcal{O}} \bot$ denote that A is a tautology and unsatisfiable concept, respectively. Moreover, the subsumption $\top \sqsubseteq_{\mathcal{O}} \bot$ implies that the ontology $\mathcal{O}$ admits no model, which means that it is inconsistent. To this end, it is not difficult to prove that Theorem 27 is still true if A ranges over $\mathsf{CN}^{\top}(\mathcal{O})$ and B over $\mathsf{CN}^{\bot}(\mathcal{O})$.

4.1.4 Refining the classification algorithm

The abstract algorithm presented in the previous subsection can classify an $\mathcal{EL}^+$ ontology in polynomial time by exhaustively applying the completion rules. One of the main problems to be addressed when implementing the described algorithm is to develop a good strategy for finding the next completion rule to be applied. If this is realized by a naïve brute-force search, then one cannot expect an acceptable runtime behavior on large inputs, despite being polynomial. As a solution to this problem, we propose a refined version of the algorithm, which is inspired by the linear-time algorithm for testing satisfiability of propositional Horn formulas proposed in [DG84].

The refined algorithm employs a set of queues—one for each concept name appearing in the input ontology—to guide the application of completion rules. Intuitively, the queues list modifications to the data structures (i.e., to the sets $S(A)$ and $R(r)$) that still have to be carried out. The possible entries of the queues are of the form

$$\mathbf{B} \rightarrow B' \quad \text{and} \quad \exists r.B$$

with B, B' concept names, r a role name, and $\mathbf{B}$ a set of concept names $B_1, \ldots, B_n$. To simplify the notation, we simply write the first type of queue entry as B' for the case $\mathbf{B} = \emptyset$ and $B_1 \rightarrow B'$ for the case where $\mathbf{B} = \{B_1\}$ is a singleton set. Intuitively,

- an entry $\mathbf{B} \rightarrow B'$ in $\mathsf{queue}(A)$ means that B' has to be added to $S(A)$ if $\mathbf{B} \subseteq S(A)$, i.e., if all elements in $\mathbf{B}$ subsume A, then so does B'; and,
- $\exists r.B \in \mathsf{queue}(A)$ means that (A, B) has to be added to $R(r)$.

Observe that an augmentation made to either $S(\cdot)$ or $R(\cdot)$ may trigger applicability of other completion rules. Such potentially applicable rules are taken into account by appropriately extending the respective queues immediately after the augmentation takes place.

To facilitate describing the manipulation of the queues, we view the input ontology $\mathcal{O}$ (in normal form without range restrictions) as a mapping $\widehat{\mathcal{O}}$ from concept descriptions to sets of queue entries as follows: for each atomic concept description $A \in \mathsf{CN}^{\top}(\mathcal{O})$, $\widehat{\mathcal{O}}(A)$ is the minimal set of queue entries such that

- if $B_1 \sqcap \cdots \sqcap B_n \sqsubseteq B \in \mathcal{O}$ and $A = B_i$ for $1 \leq i \leq n$, then
$$(\{B_j \mid 1 \leq j \leq n \text{ with } B_j \neq A\} \to B) \in \widehat{\mathcal{O}}(A);$$
- if $A \sqsubseteq \exists r.B \in \mathcal{O}$, then $\exists r.B \in \widehat{\mathcal{O}}(A)$.

Likewise, for each concept $\exists r.A$, $\widehat{\mathcal{O}}(\exists r.A)$ is the minimal set of queue entries such that, if $\exists r.A \sqsubseteq B \in \mathcal{O}$, then $B \in \widehat{\mathcal{O}}(\exists r.A)$. Intuitively, $\widehat{\mathcal{O}}$ maps a premise (i.e., a concept name or an existential quantification occurring on the left-hand side of a concept inclusion) to its *immediate* consequences (i.e., queue entries) as specified by the concept inclusions in $\mathcal{O}$. An immediate consequence is *conditional* if it is of the form $\mathbf{B} \to B'$ and $\mathbf{B}$ is not empty.

Since the queues in the refined algorithm are tightly coupled with the values in the core mappings $R(\cdot)$ and $S(\cdot)$, we need to take them into account when the queues are initialized. Recall that $R(r)$ is initialized with $\{(A, A) \mid A \in \mathsf{CN}^\top(\mathcal{O})\}$ when the role r is reflexive (i.e., $\mathcal{O} \models \epsilon \sqsubseteq r$) and the empty set otherwise; whereas $S(A)$ is initialized with $\{A, \top\}$. Therefore, we initialize $\mathsf{queue}(A)$ with

$$\underbrace{\widehat{\mathcal{O}}(A)}_{A \in S(A)} \quad \cup \quad \underbrace{\widehat{\mathcal{O}}(\top)}_{\top \in S(A)} \quad \cup \quad \underbrace{\{x \in \widehat{\mathcal{O}}(\exists r.A) \cup \widehat{\mathcal{O}}(\exists r.\top) \mid \mathcal{O} \models \epsilon \sqsubseteq r\}}_{(A,A) \in R(r)}$$

In other words, we initialize the queue of A by adding to it the immediate consequences of being an instance of A, $\top$, and—in the case that r is reflexive—$\exists r.A$ and $\exists r.\top$.

Then, we repeatedly fetch (and thereby remove) entries from the queues and process them using the procedure $\mathsf{process}$ displayed in Algorithm 1. To be more precise, the refined classification procedure dispatches the next available queue entry of $\mathsf{queue}(A)$, together with the concept name A itself, to the procedure $\mathsf{process}$. The queues are being shrunk and expanded throughout classification, but eventually they become all empty.

Observe that the first outer if-clause (line 1) of the procedure $\mathsf{process}$ implements **CR1**, part of **CR3**, and part of **CR4**; whereas the second outer if-clause (line 12) implements **CR2**, the rest of **CR3** and **CR4**, as well as **CR5** and **CR6**. The recursive procedure $\mathsf{process\text{-}new\text{-}edge}(A, r, B)$ is called by $\mathsf{process}$ to handle the effects of adding a new pair (A, B) to $R(r)$. Recall that the notation $\mathcal{O} \models r \sqsubseteq s$ used in its top-most **for**-loop stands for the reflexive-transitive closure of the role hierarchy axioms in $\mathcal{O}$. Similarly, the recursive procedure $\mathsf{process\text{-}bottom}(A)$ is called whenever the bottom concept is involved in the rule application. In accordance with Rule **CR4**, the procedure not only adds $\bot$ to $S(A)$ but also propagates it to $S(A')$ for all predecessors A' of A.

Observe also that the refined algorithm need not perform *any* search to check which completion rules are applicable. Once all the queues are empty, the classification is done in the sense that no more completion rule is applicable.

The soundness and completeness of the refined algorithm will be demonstrated by showing its conformity with the abstract algorithm and by the virtue that the abstract algorithm is sound and complete. First of all, however, we demonstrate termination of the algorithm in polynomial time.

Algorithm 1 The refined classification algorithm.

```
Procedure process($A, X$)
Input: $A$: concept name; $X$: queue entry
1: if $X = \mathbf{B} \rightarrow B$, $\mathbf{B} \subseteq S(A)$ and $B \notin S(A)$ then
2:   if $B = \bot$ then
3:     process-bottom($A$)
4:   else
5:     $S(A) := S(A) \cup \{B\}$
6:     queue($A$) := queue($A$) $\cup\ \widehat{\mathcal{O}}(B)$
7:     for all concept names $A'$ and role names $r$ with $(A', A) \in R(r)$ do
8:       queue($A'$) := queue($A'$) $\cup\ \widehat{\mathcal{O}}(\exists r.B)$
9:     end for
10:  end if
11: end if
12: if $X = \exists r.B$ and $(A, B) \notin R(r)$ then
13:   if $\bot \in S(B)$ and $\bot \notin S(A)$ then
14:     process-bottom($A$)
15:   end if
16:   process-new-edge($A, r, B$)
17: end if

Procedure process-new-edge($A, r, B$)
Input: $A, B$: concept names; $r$: role name
1: for all role names $s$ with $\mathcal{O} \models r \sqsubseteq s$ do
2:   $R(s) := R(s) \cup \{(A, B)\}$
3:   queue($A$) := queue($A$) $\cup \bigcup_{\{B' \mid B' \in S(B)\}} \widehat{\mathcal{O}}(\exists s.B')$
4:   for all concept names $A'$ and role names $t, u$ with $t \circ s \sqsubseteq u \in \mathcal{O}$ such that
     $(A', A) \in R(t)$ and $(A', B) \notin R(u)$ do
5:     process-new-edge($A', u, B$)
6:   end for
7:   for all concept names $B'$ and role names $t, u$ with $s \circ t \sqsubseteq u \in \mathcal{O}$ such that
     $(B, B') \in R(t)$ and $(A, B') \notin R(u)$ do
8:     process-new-edge($A, u, B'$)
9:   end for
10: end for

Procedure process-bottom($A$)
Input: $A$: concept name
1: $S(A) := S(A) \cup \{\bot\}$
2: for all concept names $A'$ and role names $r$ with $(A', A) \in R(r)$ such that $\bot \notin$
   $S(A')$ do
3:   process-bottom($A'$)
4: end for
```

Lemma 28 (Termination). *If the refined algorithm is applied to an ontology $\mathcal{O}$ with $|\mathcal{O}| = n$, then it terminates after $O(n^4)$ additions to the data structures $S(\cdot)$, $R(\cdot)$ and $\mathsf{queue}(\cdot)$.*

Proof. To show termination of the refined algorithm, it suffices to show that there are at most n^4 additions to the data structure since every infinite run of the algorithm must definitely perform infinitely many additions to the data structures. For the sets $S(\cdot)$ and $R(\cdot)$, each element can be added at most once and after being added no element is ever deleted. In the case of $\mathsf{queue}(\cdot)$, however, entries are eventually deleted (by means of fetching), and the same entry can be added to a specific queue (e.g., $\mathsf{queue}(A)$) several times.

Consider additions to the sets $S(\cdot)$. Since there are at most n such sets, each set can contain at most n elements, and elements are never deleted, there can be at most n^2 additions. Analogously, it can be demonstrated that there can be at most n^3 additions for the sets $R(\cdot)$. Now, consider additions to $\mathsf{queue}(\cdot)$. These are only made together with an addition to $S(\cdot)$ or $R(\cdot)$. More precisely, together with a single addition to $S(A)$ come at most n additions (bounded by $|\widehat{\mathcal{O}}(B)|$) to $\mathsf{queue}(A)$ and at most n additions (bounded by $|\widehat{\mathcal{O}}(\exists r.B)|$) to $\mathsf{queue}(A')$ for all predecessors A' of A—at most n^2 additions for every addition to $S(\cdot)$. Each single addition to $R(\cdot)$ is followed by at most n additions (bounded by the number of concept names) to $\mathsf{queue}(\cdot)$.

Overall, this implies the bound of $O(n^4)$ for the total number of additions to the data structures. ❏

Lemma 29 (Soundness). *After the refined algorithm terminates on an ontology $\mathcal{O}$, the following holds for all $A, B \in \mathsf{CN}^\top(\mathcal{O})$:*

$$\text{if } \{B, \bot\} \cap S(A) \neq \emptyset, \text{ then } A \sqsubseteq_\mathcal{O} B.$$

Proof. To show soundness, we introduce a number of invariants on the three interdependent data structures used in the refined algorithm and then show that the invariants hold throughout the computation.

INV1 If $B \in S(A)$, then $A \sqsubseteq_\mathcal{O} B$.

INV2 If $(A, B) \in R(r)$, then $A \sqsubseteq_\mathcal{O} \exists r.B$

INV3 If $\exists r.B \in \mathsf{queue}(A)$, then $A \sqsubseteq_\mathcal{O} \exists r.B$

INV4 If $\mathbf{B} \rightarrow B \in \mathsf{queue}(A)$, then

$$\textstyle\bigwedge_{B_i \in \mathbf{B}} (A \sqsubseteq_\mathcal{O} B_i) \text{ implies } A \sqsubseteq_\mathcal{O} B.$$

Observe, in case that $\mathbf{B} = \emptyset$, that **INV4** degenerates to "if $B \in \mathsf{queue}(A)$, then $A \sqsubseteq_\mathcal{O} B$". As a special case of **INV1**, if $\bot \in S(A)$, then $A \sqsubseteq_\mathcal{O} \bot$. But, this vacuously implies $A \sqsubseteq_\mathcal{O} B$ for any concept name B. So, satisfaction of **INV1** throughout (and after) the computation implies Lemma 29.

We start with demonstrating that the invariants hold after the initialization of the refined algorithm. Since each of the sets $S(A)$ is initialized with $\{A, \top\}$, **INV1** is clearly satisfied. Each of the sets $R(r)$ is initialized with $\{(A, A) \mid A \in \mathsf{CN}^\top(\mathcal{O})\}$ if r is reflexive and $\emptyset$ otherwise. In the latter case, **INV2** is trivially satisfied. In the former case, since $(d, d) \in r^\mathcal{I}$ for every model $\mathcal{I}$ of $\mathcal{O}$, $d \in A^\mathcal{I}$ implies $d \in (\exists r.A)^\mathcal{I}$, satisfying **INV2**. Now suppose that $\exists r.B \in \mathsf{queue}(A)$. Since queue entries of this type (existential quantification) are derived from $\widehat{\mathcal{O}}(A) \cup \widehat{\mathcal{O}}(\top)$, either $A \sqsubseteq \exists r.B$ or $\top \sqsubseteq \exists r.B$ must belong to $\mathcal{O}$. Either case implies that $A \sqsubseteq_\mathcal{O} \exists r.B$, and thus **INV3** is satisfied. For the last invariant, suppose that $\mathbf{B} \to B \in \mathsf{queue}(A)$. There are two possibilities for $\mathbf{B}$: $\mathbf{B} = \{B_1, \cdots, B_n\}$ and $\mathbf{B} = \emptyset$. In the first case, there must exist (up to commutativity of $\sqcap$) a GCI $A \sqcap B_1 \sqcap \cdots \sqcap B_n \sqsubseteq B$ in $\mathcal{O}$. Due to the presence of this GCI and $A \sqsubseteq_\mathcal{O} A$, **INV4** is satisfied. In the second case, $B \in \mathsf{queue}(A)$ may be derived either from $A \sqsubseteq B$ or $\top \sqsubseteq B$ in $\mathcal{O}$ or—in the case that r is reflexive—from $\exists r.A \sqsubseteq B$ or $\exists r.\top \sqsubseteq B$ in $\mathcal{O}$. For the first two GCIs, it vacuously holds that $A \sqsubseteq_\mathcal{O} B$, thus satisfying **INV4**. Since r is reflexive in the last two cases, $(d, d) \in r^\mathcal{I}$ for every model $\mathcal{I}$ of $\mathcal{O}$ and $d \in \Delta^\mathcal{I}$. If $d \in A^\mathcal{I}$, then $d \in (\exists r.A)^\mathcal{I}$. The third GCI implies that $d \in B^\mathcal{I}$ and thus satisfies **INV4**. Since $d \in \top^\mathcal{I}$, **INV4** is also satisfied w.r.t. the last GCI.

After the initialization, the data structures are manipulated by the procedures process (in lines 5, 6 and 8), process-new-edge (in lines 2 and 3) and process-bottom (in line 1) in Algorithm 1. For the sake of succinctness, we refer to these lines as **L1**–**L6**, respectively. Now, it remains to be shown that each manipulation preserves the invariants.

L1 We need to show that **INV1** (the only invariant that could be invalidated by **L1**) is preserved. This line adds B to $S(A)$ when $\mathsf{process}(A, \mathbf{B} \to B)$ is invoked with $\mathbf{B} \subseteq S(A)$ and $B \neq \bot$. The fact that $\mathbf{B} \to B$ is processed w.r.t. A means that it was in the queue of A. By **INV4**, we have that $\bigwedge_{B_i \in \mathbf{B}}(A \sqsubseteq_\mathcal{O} B_i)$ implies $A \sqsubseteq_\mathcal{O} B$. For each $B_i \in \mathbf{B}$, we have $A \sqsubseteq_\mathcal{O} B_i$ due to **INV1** and $B_i \in S(A)$. This, together with the implication above, yields $A \sqsubseteq_\mathcal{O} B$. Hence, **INV1** is preserved as required.

L2 We need to show that **INV3** and **INV4** are preserved. This line adds all the elements of $\widehat{\mathcal{O}}(B)$ to $\mathsf{queue}(A)$ when process is invoked with arguments $(A, \mathbf{B} \to B)$ such that $\mathbf{B} \subseteq S(A)$ and $B \neq \bot$. We know that prior to this B has been added to $S(A)$ (in line **L1**), which implies $A \sqsubseteq_\mathcal{O} B$ by **INV1**. Thus, we may argue as for the initialization step that **INV3** and **INV4** are preserved.

L3 We need to show that **INV3** and **INV4** are preserved. This line adds all the elements of $\widehat{\mathcal{O}}(\exists r.B)$ to $\mathsf{queue}(A')$ for all r-predecessors A' of A if $B \in S(A)$. By **INV1** and **INV2**, respectively, we have $A \sqsubseteq_\mathcal{O} B$ and $A' \sqsubseteq_\mathcal{O} \exists r.A$, which implies $A' \sqsubseteq_\mathcal{O} \exists r.B$. The existence of an element B' in $\widehat{\mathcal{O}}(\exists r.B)$ implies that the GCI $\exists r.B \sqsubseteq B'$ belongs to $\mathcal{O}$. Hence, we have $A' \sqsubseteq_\mathcal{O} B'$ and adding B' to $\mathsf{queue}(A')$ preserves (the $\mathbf{B} = \emptyset$ case of) **INV4**. Note that **INV3** is intact since only atomic concepts can be elements of $\widehat{\mathcal{O}}(\exists r.B)$.

L4 We need to show that **INV2** is preserved. This line adds (A, B) to $R(s)$ when $\mathsf{process\text{-}new\text{-}edge}(A, r, B)$ is invoked and $\mathcal{O} \models r \sqsubseteq s$. We make a case distinction according to the three possible reasons for which process-new-edge may be invoked.

First, assume that process-new-edge(A, r, B) is called by the process procedure. In this case, $\exists r.B$ was in the queue of A, and **INV3** yields $A \sqsubseteq_{\mathcal{O}} \exists r.B$. Together with $\mathcal{O} \models r \sqsubseteq s$, this implies $A \sqsubseteq_{\mathcal{O}} \exists s.B$, and thus **INV2** is preserved.

Second, assume that process-new-edge(A, r, B) is called recursively by itself in line 5. Then, there are role names t, u and a concept name A' such that $t \circ u \sqsubseteq r \in \mathcal{O}$, $(A, A') \in R(t)$ and $(A', B) \in R(u)$. By **INV2**, we have $A \sqsubseteq_{\mathcal{O}} \exists t.A'$ and $A' \sqsubseteq_{\mathcal{O}} \exists u.B$. Together with $t \circ u \sqsubseteq r \in \mathcal{O}$ and $\mathcal{O} \models r \sqsubseteq s$, we get $A \sqsubseteq_{\mathcal{O}} \exists s.B$ as required to preserve **INV2**.

Third, assume that process-new-edge(A, r, B) is called recursively by itself in line 8. It can be shown in analogy to the previous case that **INV2** is preserved.

L5 We need to show that **INV3** and **INV4** are preserved. This line adds the elements of $\widehat{\mathcal{O}}(\exists s.B')$ to queue(A) only if $(A, B) \in R(s)$ and $B' \in S(B)$. With the same arguments as in case of **L3**, we have $A \sqsubseteq_{\mathcal{O}} \exists s.B \sqsubseteq_{\mathcal{O}} \exists s.B' \sqsubseteq_{\mathcal{O}} B''$ for all $B'' \in \widehat{\mathcal{O}}(\exists s.B')$. Thus, (the $\mathbf{B} = \emptyset$ case of) **INV4** is preserved. Again, **INV3** is intact since only atomic concepts can be elements of $\widehat{\mathcal{O}}(\exists r.B)$.

L6 We need to show that **INV1** is preserved. This line adds the bottom concept to $S(A)$ when process-bottom(A) is invoked. Again, we make a case distinction according to the three sources of invocation.

First, assume that process-bottom(A) is called by line 3 of the process procedure. In this case, $\mathbf{B} \rightarrow \bot$ was in the queue of A, and $\mathbf{B} \subseteq S(A)$. With the same arguments as in the case of **L1**, we have that $A \sqsubseteq_{\mathcal{O}} \bot$. Thus, **INV1** is preserved.

Second, assume that process-bottom(A) is called by line 14 of the process procedure. In this case, $\exists r.B$ was in the queue of A, and $\bot \in S(B)$. The former implies $A \sqsubseteq_{\mathcal{O}} \exists r.B$ by **INV3**, while the latter implies $B \sqsubseteq_{\mathcal{O}} \bot$ by **INV1**. Putting together, we have that $A \sqsubseteq_{\mathcal{O}} \exists r.\bot \sqsubseteq_{\mathcal{O}} \bot$, which preserves **INV1**.

Third, assume that process-bottom(A') is called recursively by itself in line 3. Then, there is an r-successor A of A' with $\bot \in S(A)$. By **INV1**, **INV2**, and the semantics, we have $A' \sqsubseteq_{\mathcal{O}} \exists r.A \sqsubseteq_{\mathcal{O}} \exists r.\bot \sqsubseteq_{\mathcal{O}} \bot$. Thus, **INV1** is preserved.

❑

Lemma 30 (Completeness). *After the refined algorithm terminates on an ontology $\mathcal{O}$, the following holds for all $A, B \in \mathsf{CN}^{\top}(\mathcal{O})$:*

$$\text{if } A \sqsubseteq_{\mathcal{O}} B, \text{ then } \{B, \bot\} \cap S(A) \neq \emptyset.$$

Proof. Instead of proving completeness of the refined algorithm from scratch, we demonstrate it by reusing the known completeness result of the abstract algorithm given in [BBL05]. To recall, the completeness proofs of the abstract algorithm shows the following: if $\mathcal{O}$ is an ontology, S, R are mappings as defined in Section 4.1.3 such that

- for every $A \in \mathsf{CN}^{\top}(\mathcal{O})$, $S(A)$ contains A and $\top$;

- for every $r \in \mathsf{RN}(\mathcal{O})$, $R(r)$ contains (A, A) for all $A \in \mathsf{CN}^\top(\mathcal{O})$ if $\mathcal{O} \models \epsilon \sqsubseteq r$;
- none of the completion rules in Figure 4.2 on page 52 is applicable to $S, R, \mathcal{O}$;

then $A \sqsubseteq_\mathcal{O} B$ implies $\{B, \bot\} \subseteq S(A)$.

In order to show Lemma 30, we demonstrate that the three itemized properties hold after the refined algorithm terminates. The first two properties hold since the refined algorithm initializes the sets $S(\cdot)$ and $R(\cdot)$ to precisely contain those elements. Moreover, it never removes elements from $S(\cdot)$ and $R(\cdot)$. It remains to be shown that, after termination of the refined algorithm, no completion rule is applicable.

Assume to the contrary of what has to be proven that there exists an applicable completion rule after termination of the refined algorithm. We make a case distinction according to the kind of completion rule and show that in each case our assumption leads to a contradiction.

CR1 If this rule is applicable, then there exists an $X \in \mathsf{CN}^\top(\mathcal{O})$ and a concept inclusion $A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B$ in $\mathcal{O}$ such that $A_1, \ldots, A_n \in S(X)$ and $B \notin S(X)$. Let $\ell \leq n$ be such that A_ℓ is the last element among $A_1, \ldots, A_n$ to have been added to $S(X)$. When A_ℓ was added, the entry $(\mathbf{A}\backslash\{A_\ell\} \rightarrow B) \in \widehat{\mathcal{O}}(A_\ell)$ was also put into $\mathsf{queue}(X)$ due to the presence of the concept inclusion. Since $B \notin S(X)$, the conditional queue entry has not yet been processed w.r.t. X. This implies that $\mathsf{queue}(X)$ is not empty and that the refined algorithm has not terminated, contradicting the initial assumption.

CR2 If this rule is applicable, there exists an $X \in \mathsf{CN}^\top(\mathcal{O})$ and a GCI $A \sqsubseteq \exists r.B$ in $\mathcal{O}$ such that $A \in S(X)$ and $(X, B) \notin R(r)$. When A was added to $S(X)$, the entry $\exists r.B \in \widehat{\mathcal{O}}(A)$ was added to $\mathsf{queue}(X)$. If this queue entry had been processed, $\mathsf{process\text{-}new\text{-}edge}(X, r, B)$ would have been invoked, having added (X, B) to $R(r)$. Since $(X, B) \notin R(r)$, the entry has not yet been processed and the queue is not empty, contradicting the initial assumption.

CR3 If this rule is applicable, there are $X, Y \in \mathsf{CN}^\top(\mathcal{O})$ and a concept inclusion $\exists r.A \sqsubseteq B$ in $\mathcal{O}$ such that $(X, Y) \in R(r)$, $A \in S(Y)$ and $B \notin S(X)$. We handle the following two cases separately:

First, assume that (X, Y) had been added to $R(r)$ before A was added to $S(Y)$. Then, the entry $B \in \widehat{\mathcal{O}}(\exists r.A)$ was added to $\mathsf{queue}(X)$ when A was added to $S(Y)$. However, $B \notin S(X)$ means that this entry has not yet been processed and the queue is not empty, contradicting the initial assumption.

Second, assume that A had been added to $S(Y)$ before (X, Y) was added to $R(r)$. Then, the entry $B \in \widehat{\mathcal{O}}(\exists r.A)$ was added to $\mathsf{queue}(X)$ when (X, Y) was added to $R(r)$. Now, we can continue as in the first case.

CR4 If this rule is applicable, there are $X, Y \in \mathsf{CN}^\top(\mathcal{O})$ such that $(X, Y) \in R(r)$, $\bot \in S(Y)$ and $\bot \notin S(X)$. Again, we distinguish the following two cases:

First, assume that (X, Y) had been added to $R(r)$ before $\bot$ was added to $S(Y)$. When the latter took place, the procedure $\mathsf{process\text{-}bottom}$ recursively propagated

$\bot$ to all predecessors of Y. Since X is a predecessor of Y, the recursive call ensures that $\bot$ is eventually added to $S(X)$. This contradicts the applicability of Completion Rule **CR4**.

Second, assume that $\bot$ had been added to $S(Y)$ before (X, Y) was added to $R(r)$. When the latter took place, the procedure process-bottom is invoked immediately, adding $\bot$ to $S(X)$. This contradicts the applicability of Completion Rule **CR4**.

CR5 If this rule is applicable, there are $X, Y \in \mathsf{CN}^\top(\mathcal{O})$ and a role inclusion $r \sqsubseteq s$ in $\mathcal{O}$ such that $(X, Y) \in R(r)\backslash R(s)$. The addition of (X, Y) to $R(r)$ took place in the procedure process-new-edge that was invoked with arguments (X, u, Y) for some role name u with $\mathcal{O} \models u \sqsubseteq r$. Since $r \sqsubseteq s \in \mathcal{O}$, we also have that $\mathcal{O} \models u \sqsubseteq s$ and that (X, Y) was also added to $R(s)$ in the same procedure call. This contradicts the applicability of Completion Rule **CR5**.

CR6 If this rule is applicable, there are $X, Y, Z \in \mathsf{CN}^\top(\mathcal{O})$ and a role inclusion $r \circ s \sqsubseteq t$ in $\mathcal{O}$ such that $(X, Y) \in R(r)$, $(Y, Z) \in R(s)$ and $(X, Z) \notin R(t)$. We distinguish the following two cases:

First, assume that (X, Y) had been added to $R(r)$ before (Y, Z) was added to $R(s)$. When the pair (Y, Z) was added to $R(r)$ in the procedure process-new-edge, the first inner **for**-loop and the recursive call therein ensure that (X, Z) is eventually added to $R(t)$. This contradicts the applicability of Completion Rule **CR6**.

Second, assume that (Y, Z) had been added to $R(s)$ before (X, Y) was added to $R(r)$. This case is parallel to the previous case with the second inner **for**-loop in place of the first.

❏

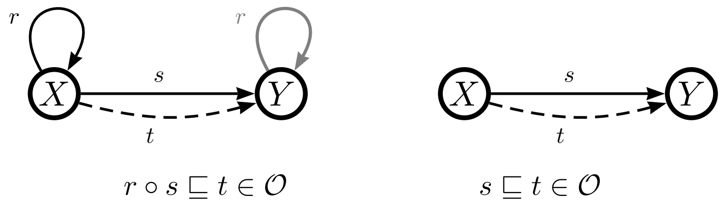
Combining Lemma 29, 30 and 28, correctness of the refined $\mathcal{EL}^+$ classification algorithm as stated by the following theorem is readily proved.

Theorem 31 (Correctness of the refined algorithm). *The refined algorithm runs in polynomial time, and it is sound and complete, i.e., after it terminates on the input ontology $\mathcal{O}$ we have, for all concept names A, B in $\mathcal{O}$, that*

$$\{B, \bot\} \cap S(A) \neq \emptyset \textit{ if, and only if, } A \sqsubseteq_{\mathcal{O}} B.$$

4.1.5 Optimization techniques

The main advantage of the refined classification algorithm over the abstract one is essentially the fact that it need not perform search for applicable completion rules. To enable acceptable classification time for large-scale ontologies, however, additional optimizations are inevitable. This subsection describes some of the optimization techniques which have been employed in the CEL reasoner (see [BLS06, Sun05a]).



r
r
s
t
X
Y
$r \circ s \sqsubseteq t \in \mathcal{O}$
s
t
X
Y
$s \sqsubseteq t \in \mathcal{O}$

Figure 4.3: Dispensing with the r reflexive loops.

Avoid populating the reflexive role relation with identity pairs. The initialization adds all the identity pair (A, A), for all $A \in \mathsf{CN}^\top(\mathcal{O})$, into $R(r)$ whenever r is reflexive. Instead of explicitly putting the identity pairs into the role relation $R(r)$, handling of reflexivity can be hardcoded in the algorithm. For example, in addition to the queue extension in lines 7–9 of process in Algorithm 1 on page 55, the following must be added:

```
for all role names $r$ with $\mathcal{O} \models \epsilon \sqsubseteq r$ do
    $\mathsf{queue}(A) := \mathsf{queue}(A) \cup \widehat{\mathcal{O}}(\exists r.B)$
end for
```

Additionally, we need to add a role hierarchy axiom $s \sqsubseteq t$ to $\mathcal{O}$ for each role inclusion $r \circ s \sqsubseteq t$ (or $s \circ r \sqsubseteq t$) in $\mathcal{O}$ with r reflexive. As a result, an application of **CR6** with $(X, X) \in R(r), (X, Y) \in R(s), r \circ s \sqsubseteq t \in \mathcal{O}$ boils down to one of **CR5** with $(X, Y) \in R(s), s \sqsubseteq t \in \mathcal{O}$ (see Figure 4.3).

Domain and range restrictions on reflexive roles. Similar to the second operation for elimination of range restrictions (see Subsection 4.1.2), a domain restriction imposed on a reflexive role is a tautology. To be precise, we can exchange a concept inclusion of the form $\exists r.\top \sqsubseteq A$ (equivalent to $\mathsf{domain}(r) \sqsubseteq A$) in $\mathcal{O}$ for $\top \sqsubseteq A$ if r is reflexive (i.e., $\mathcal{O} \models \epsilon \sqsubseteq r$).

Moreover, the range elimination in Subsection 4.1.2 can be simplified by skipping the first operation if the role under consideration is reflexive. Precisely, exchange $\mathsf{range}(r) \sqsubseteq A$ in $\mathcal{O}$ for the concept inclusion $\top \sqsubseteq A$ if $\mathcal{O} \models \epsilon \sqsubseteq r$.

Isolation of $S(\top)$. The top concept and all its consequences of the forms $X \in S(\top)$ and $(\top, X) \in R(r)$ are also consequences of other concept names. This is the reason why $\top$ is put into $S(A)$ alongside A during initialization. However, this makes computation w.r.t. consequences of $\top$ unnecessarily redundant in the sense that an identical pattern of computation is carried out once for each set $S(A)$. To avoid this, we isolate $S(\top)$ from $S(A)$ by initializing $S(A)$ with the singleton set $\{A\}$, for every concept name $A \in \mathsf{CN}(\mathcal{O})$. Membership testing operation such as $B \in^? S(A)$ now have to explicitly incorporate consequences from the top concept: $B \in^? S(A) \cup S(\top)$.

Told synonyms and structural equivalences Two concept names $A, B \in \mathsf{CN}^\top(\mathcal{O})$ are said to be told synonyms in $\mathcal{O}$ if there are $A_0, \ldots, A_{n-1}$, for $n > 1$, such that $A = A_0$, $B = A_{n-1}$ and $A_i \equiv A_{(i+1) \bmod n} \in \mathcal{O}$, for all $i < n$. After the classification algorithm terminates, any told synonyms A, B have the same set of subsumers, i.e., $S(A) = S(B)$. Thus, it is more economical to maintain and compute only one such set for each synonymous class. Told synonyms could be extended to structural equivalences. For instance, the concept description $A \sqcap B$ is structurally equivalent to $B \sqcap A$. By recognizing structural equivalence, we can minimize the number of newly introduced concept names during normalization. For instance, $\exists r.(A \sqcap B)$ and $\exists r.(B \sqcap A)$ are both normalized to $\exists r.X$ with X the only new concept name that represents both $A \sqcap B$ and $B \sqcap A$.

Additionally, a fully defined concept name B with $B \equiv \hat{C} \in \mathcal{O}$ is used to replace the concept description $\hat{C}$ during normalization instead of introducing a new name A and adding a GCI. For instance, **NR2-3** simply rewrites $\exists r.\hat{C} \sqsubseteq D$ to $\exists r.B \sqsubseteq D$.

Multiple queues and types of entries In the procedure process, the type of queue entry first has to be checked and then processed accordingly. There are three types of queue entries: $\exists r.B$, $\mathbf{B} \rightarrow B$ and B (the special case of $\mathbf{B} \rightarrow B$ where $\mathbf{B}$ is empty). Type checking (line 1 and 12 in process) of queue entries can be avoided by making distinction from the start, i.e., instead of viewing the ontology $\mathcal{O}$ as a single mapping $\widehat{\mathcal{O}}$, we could view it as three mappings:

- $\widehat{\mathcal{O}}_1$ maps concepts to queue entries of type B;
- $\widehat{\mathcal{O}}_2$ maps concepts to queue entries of type $\mathbf{B} \rightarrow B$;
- $\widehat{\mathcal{O}}_1$ maps concepts to queue entries of type $\exists r.B$;

Also, we use three queues for each concept name, one for each type of queue entry. Queues queue_1,queue_2 and queue_3 are extended w.r.t. $\widehat{\mathcal{O}}_1$, $\widehat{\mathcal{O}}_2$ and $\widehat{\mathcal{O}}_3$, respectively. We extract three execution paths from process and therewith define three variants of the procedure. Queue processing is done in a similar way, but now the right execution path is determined by which queue is being processed. Thus, not only can we avoid a number of **if**-statements, queues can also be prioritized based on their type. Finally, finer grained data structures may be chosen which help to reduce storage requirement, for instance, $\widehat{\mathcal{O}}_1(\cdot)$ and $\mathsf{queue}_1(\cdot)$ are sets of *atomic* concepts in contrast to existential quantifications and conditional queue entries.

Pruning of the completion graph. Since $\bot \in S(A)$ implies that $A \sqsubseteq_{\mathcal{O}} B$ for any concept $B \in \mathsf{CN}^{\top}(\mathcal{O})$, subsequent rule applications that add B' to $S(A)$ do not provide new information. Thus, we need not perform further rule applications involving $S(A)$ as soon as it is known to be unsatisfiable. To minimize storage requirement, $S(A)$ is reduced to $\{\bot\}$. Furthermore, due to **CR4** unsatisfiability propagates to all ancestors in the completion graph. Also, any path $(A_1, A_2) \in R(r_1), \cdots, (A_k, A_{k+1}) \in R(r_k)$ with $\bot \in S(A_{k+1})$ can be pruned from the completion graph.

Disabling $R(\cdot)$ for primitive TBoxes. A *primitive TBox* merely contains primitive concept definitions in which $\bot$ does not occur. Since no full concept definitions nor concept inclusions occur in such a specialized case of $\mathcal{EL}^+$ ontology, normalized

GCIs are either of the form **GCI1** or **GCI2** where $\bot$ does not occur (see Definition 24 on page 46). Obviously, Completion Rule **CR3** and **CR4** can never be applied. Since explicit information in $R(\cdot)$ is used by **CR3** and **CR4** to infer implicit subsumption involving a GCI of the form **GCI3** or $\bot$, we can disable $R(\cdot)$ and the related completion rules, namely, all the rules in Figure 4.2 on page 52 except for **CR1**.

In fact, classifying such a primitive TBox boils down to computation of the reflexive-transitive closure of told subsumption relationships.

4.2 Querying Subsumption

Unlike structural subsumption algorithms for inexpressive DLs [Küs00, BN07] and tableau-based algorithms for expressive DLs [BS01, BN07], the algorithm presented above not only tests a particular subsumption but classifies the ontology. Since the problem of classification can be viewed as a set of subsumption queries, we can in fact use the classification algorithm as a decision procedure for subsumption in $\mathcal{EL}^+$. This approach is however not directed by the subsumption in question, in the sense that it computes all positive subsumption relationships and simply checks if the subsumption in question is among them.

To avoid computation of irrelevant subsumption relationships, we modify (the refined version of) the classification algorithm from Subsection 4.1.4 to obtain a *goal-directed* subsumption algorithm. In a nutshell, the goal-directed subsumption algorithm uses information in the subsumption query $A_0 \sqsubseteq^?_{\mathcal{O}} B_0$ to guide where the computation starts and when it stops. It takes as input not only an $\mathcal{EL}^+$ ontology but also two concept names A_0, B_0 between which subsumption is to be queried. Similar to the refined classification algorithm, it normalizes the ontology (see Subsection 4.1.1) and, if any, eliminates range restrictions (see Subsection 4.1.2) in it before the core procedure can be started. In particular, it views the ontology in normal form without range restrictions as the mapping $\widehat{\mathcal{O}}$ and uses the same data structures queue, R and S as before. Algorithm 2 outlines the modified core procedure goal-directed-process to replace process of Algorithm 1 on page 55. The recursive procedures process-new-edge process-bottom are not shown here since both of them are intact.

The main difference is the initialization of R and S, thus of queue. Since we are not interested in all subsumption relationships between all pairs of concept names, we need not initialize $S(\cdot)$ and $R(\cdot)$ w.r.t. all concept names. Initialization of the mappings, thus of the queues, is done on demand when a concept name becomes 'activated.' Since we are interested in the particular subsumption $A_0 \sqsubseteq B_0$, we activate only the concept name A_0 at the beginning by initializing $S(A_0)$ with $\{A_0, \top\}$, $R(r)$ with $\{(A_0, A_0)\}$ when the role r is reflexive (i.e., $\mathcal{O} \models \epsilon \sqsubseteq r$) and the empty set otherwise, and queue(A_0) with

$$\widehat{\mathcal{O}}(A_0) \quad \cup \quad \widehat{\mathcal{O}}(\top) \quad \cup \quad \bigcup_{r \text{ with } \mathcal{O}\models\epsilon\sqsubseteq r} \Big(\widehat{\mathcal{O}}(\exists r.A_0) \ \cup \ \widehat{\mathcal{O}}(\exists r.\top)\Big)$$

Another concept name B is activated *only* when it becomes the second component of a tuple added to some $R(r)$ and has not previously been activated (see line 15-16 in

Algorithm 2 The goal-directed subsumption algorithm.

Procedure subsumes?($A_0 \sqsubseteq B_0$)
Input: ($A_0 \sqsubseteq B_0$): target subsumption
Output: 'positive' or 'negative' answer to the subsumption

```
1: activate($A_0$)
2: while not empty(queue($A$)) for some $A \in \mathsf{CN}^\top(\mathcal{O})$ do
3:   $X \leftarrow$ fetch(queue($A$))
4:   if goal-directed-process($A, X, A_0, B_0$) = 'positive' then
5:     return 'positive'
6:   end if
7: end while
8: return 'negative'
```

Procedure goal-directed-process(A, X, A_0, B_0)
Input: A: concept name; X: queue entry ($A_0 \sqsubseteq B_0$): target subsumption
Output: 'positive' or 'unknown' answer to the subsumption

```
1: if $X = \mathbf{B} \rightarrow B$, $\mathbf{B} \subseteq S(A)$ and $B \notin S(A)$ then
2:   if $B = \bot$ then
3:     return 'positive'
4:   else
5:     if $A = A_0$ and $B = B_0$ then
6:       return 'positive'
7:     end if
8:     $S(A) := S(A) \cup \{B\}$
9:     queue($A$) := queue($A$) $\cup\ \widehat{\mathcal{O}}(B)$
10:    for all concept names $A'$ and role names $r$ with $(A', A) \in R(r)$ do
11:      queue($A'$) := queue($A'$) $\cup\ \widehat{\mathcal{O}}(\exists r.B)$
12:    end for
13:  end if
14: end if
15: if $X = \exists r.B$ and $(A, B) \notin R(r)$ then
16:   activate($B$)
17:   process-new-edge($A, r, B$)
18: end if
19: return 'unknown'
```

goal-directed-process of Algorithm 2). Thereby, $S(B)$ is initialized with $\{B, \top\}$, while $R(r)$ is augmented with the pair (B, B) in case that r is reflexive and stays the same otherwise. We initialize queue(B) accordingly in the same fashion as for A_0 above. Observe that this activation strategy is well-defined in relation with the usage of these data structures throughout the algorithm, in the sense that relevant data structures are referred to only after they have been properly initialized.

Queues are processed in the same fashion as in the classification algorithm except that the goal subsumption $A_0 \sqsubseteq B_0$ is now being monitored (line 5), so that immediately after B_0 is added to $S(A_0)$, the algorithm terminates with the positive answer (line 6). The positive answer is also returned once the bottom concept is encountered. Otherwise, goal-directed-process terminates normally, and the next queue entry will be fetched (line 3 in subsumes? of Algorithm 2) and processed (line 4). Unless 'positive' is returned, queue processing is continued until they are all empty. In this case, the algorithm returns 'negative.'

It is important to note that the goal-directed algorithm activates only concept names relevant to the target subsumption $A_0 \sqsubseteq B_0$, i.e., those nodes reachable from A_0 in the completion graph. Those that are not reachable from A_0 obviously cannot induce subsumption ramifications for A_0 and are thus irrelevant. This tenet of the goal-directed subsumption algorithm is closely related to reachability-based modularization which is introduced and discussed in detail in Section 5.1. Also, an interesting connection between these two notions is presented in Theorem 42 on page 86.

Theorem 32 (Correctness of the goal-directed algorithm). *The goal-directed subsumption algorithm runs in polynomial time, and it is sound and complete, i.e., after it terminates on the input ontology $\mathcal{O}$ and the concept names A_0, B_0, we have that:*

1. *if* subsumes$(A_0 \sqsubseteq B_0)$ *returns 'positive,' then* $A_0 \sqsubseteq_{\mathcal{O}} B_0$.
2. *if* subsumes$(A_0 \sqsubseteq B_0)$ *returns 'negative,' then* $A_0 \not\sqsubseteq_{\mathcal{O}} B_0$.

Proof. Termination follows immediately from Lemma 28. Point 1 (soundness) can be demonstrated along the same line as the proof of Lemma 29 with an additional invariant to address Point 1 of the theorem:

INV5 If goal-directed-process returns 'positive,' then $A_0 \sqsubseteq_{\mathcal{O}} B_0$.

Other invariants have been shown to hold throughout the computation of the refined algorithm, and they obviously remain so w.r.t. the goal-directed algorithm. For the new invariant, only lines 3 and 6 of goal-directed-process in Algorithm 2 are relevant. To avoid ambiguity, we denote these lines as **L7** and **L8**, respectively. Now, we show that, in these two cases, **INV5** holds.

L7 This line returns 'positive' when process$(A, \mathbf{B} \to B)$ is invoked with $\mathbf{B} \subseteq S(A)$ and $B = \bot$. The fact that $\mathbf{B} \to B$ is processed w.r.t. A means that it was in the queue of A. By **INV4**, we have that $\bigwedge_{B_i \in \mathbf{B}}(A \sqsubseteq_{\mathcal{O}} B_i)$ implies $A \sqsubseteq_{\mathcal{O}} B$. For each $B_i \in \mathbf{B}$, we have $A \sqsubseteq_{\mathcal{O}} B_i$ due to **INV1** and $B_i \subseteq S(A)$. This, together with the implication above, yields $A \sqsubseteq_{\mathcal{O}} B$. Since A has been activated, it must be reachable from A_0 via some series of $R(\cdot)$. By **INV2**, it holds that $A_0 \sqsubseteq_{\mathcal{O}} \exists r_1 \ldots \exists r_k.A$ for some $r_1, \ldots, r_k \in \mathsf{RN}(\mathcal{O})$. Since $B = \bot$, we have that $A_0 \sqsubseteq_{\mathcal{O}} \bot$ and thus $A_0 \sqsubseteq_{\mathcal{O}} B_0$, justifying **INV5**.

L8 This line returns 'positive' when process$(A, \mathbf{B} \to B)$ is invoked with $\mathbf{B} \subseteq S(A)$, $A = A_0$ and $B = B_0$. With the same arguments as in the case of **L7**, we have that $A \sqsubseteq_{\mathcal{O}} B$. Since $A = A_0$ and $B = B_0$, **INV5** is readily justified.

In order to show Point 2 (completeness), it suffices to use completeness of the abstract algorithm and demonstrate that, after termination of the goal-directed subsumption algorithm on A, B and $\mathcal{O}$, we have that

- for every $A \in \mathsf{CN}^\top(\mathcal{O})$ such that A is activated, $S(A)$ contains A and $\top$.
- for every $r \in \mathsf{RN}(\mathcal{O})$ such that $\mathcal{O} \models \epsilon \sqsubseteq r$, $R(r)$ contains (A, A) for all activated concept name A.
- if A is activated, no applicable completion rules may add (A, B) to $R(r)$ or B to $S(A)$ for $B \in \mathsf{CN}^\top(\mathcal{O})$ and $r \in \mathsf{RN}(\mathcal{O})$.

The first two properties hold since we initialize the sets $R(\cdot)$ and $S(\cdot)$ to precisely contain those elements whenever a new concept name becomes activated. These elements are never removed throughout the course of execution. Since the algorithm returns 'negative,' it is the case that neither **L7** nor **L8** is executed, implying that $\{B_0, \bot\} \cap S(A_0) = \emptyset$. By completeness of the abstract algorithm and the last property above, this implies that $A_0 \not\sqsubseteq_{\mathcal{O}} B_0$.

It remains to show that the last property holds. Assume to the contrary that, after Algorithm 2 has terminated, there exists an applicable completion rule that adds either (A, B) to $R(r)$ or B to $S(A)$ where A has already been activated. We make a case distinction according to the kind of completion rule and show that in each case our assumption leads to a contradiction.

CR1–CR3, CR5, CR6 If one of these rules is applicable, a contradiction can be demonstrated in an analogous way as in Lemma 30 on page 58, provided that X—as well as Y in the case of **CR3** and **CR5**; Y, Z in the case of **CR6**—is activated. By assumption, X is activated. Since Y, Z have become second components of $R(\cdot)$, they are also activated.

CR4 There are $X, Y \in \mathsf{CN}^\top(\mathcal{O})$ such that X is activated, $(X, Y) \in R(r)$, $\bot \in S(Y)$ and $\bot \notin S(X)$. Since the goal-directed algorithm never adds $\bot$ to $S(\cdot)$, this immediately contradicts the applicability of **CR4**.

❏

It is obvious to see that positive subsumption benefits directly from this modification since the algorithm terminates as soon as the subsumption is known to be true. Nonetheless, negative subsumption also benefits from the goal-directed approach, since the subsumer sets of concept names that do not become activated are not populated, thus saving space and time for computation.[4]

4.3 Computing the Concept Hierarchy

The innate output of the classification algorithm presented in Section 4.1 is a collection of the sets $S(A)$ for all concept names A, henceforth called *subsumer sets*. In contrast,

[4]This algorithm is implemented in the CEL reasoner and used whenever subsumption is queried prior to classification. See [Sun05a] for the CEL reference manual.

$$\begin{tikzcd}
 & & & & & \top & & & & & & \\
\mathsf{BP} \arrow[urrrrr] & \mathsf{Ti} \arrow[urrrr] & \mathsf{H} \arrow[urrr] & \mathsf{In} \arrow[urr] & & \mathsf{D} \arrow[u] & & \mathsf{V} \arrow[ull] & \mathsf{VD} \arrow[ulll] & \mathsf{NT} \arrow[ullll] & \mathsf{BV} \arrow[ulllll] & \mathsf{BW} \arrow[ullllll] \\
\mathsf{Ax} \arrow[u] & \mathsf{Em} \arrow[u] & & & \mathsf{Im} \arrow[ur] & & \mathsf{HD} \arrow[ul] & & & & \mathsf{HV} \arrow[u] & \mathsf{HW} \arrow[u] \\
 & & & \mathsf{As} \arrow[ur] & \mathsf{Es} \arrow[u] \arrow[urr] & & \mathsf{Ps} \arrow[u] \arrow[ull] & & & & &
\end{tikzcd}$$

Figure 4.4: The concept hierarchy (DAG representation) of $\mathcal{O}_{\mathsf{med}}$.

tableau-based DL reasoners usually generate a *directed acyclic graph (DAG)* describing the *direct* subsumption relation, i.e., for every concept name A they compute the sets of its direct and strict subsumers and subsumees, which are the sets of concept names B such that $A \sqsubset_{\mathcal{O}} B$ ($B \sqsubset_{\mathcal{O}} A$) and there is no concept name $X \in \mathsf{CN}(\mathcal{O}) \setminus \{A, B\}$ with $A \sqsubset_{\mathcal{O}} X \sqsubset_{\mathcal{O}} B$ ($B \sqsubset_{\mathcal{O}} X \sqsubset_{\mathcal{O}} A$).

Such a directed acyclic graph is called the *concept hierarchy* (see Definition 14 on page 27). Since the subsumption relation is a quasi-order rather than a partial order (i.e., in general not antisymmetric), one node of the DAG actually corresponds to an equivalence class of concept names rather than a single concept name. Figure 4.4 depicts the concept hierarchy of our example ontology in Figure 2.2 on page 24. Note that every node in the hierarchy (including the root) represents a *singleton* equivalence class, so we write the element (e.g., $\top$) instead of the equivalence class (e.g., $\{\top\}$). For the sake of readability, the equivalence class for $\bot$ and edges starting from it are omitted, and obvious abbreviations for concept names are used. An edge in the DAG such as $\mathsf{Es} \rightarrow \mathsf{HD}$ represents a *direct and strict* subsumption relationship between $\mathsf{Endocarditis}$ and $\mathsf{HeartDisease}$, i.e., $\mathsf{Es} \sqsubset_{\mathcal{O}_{\mathsf{med}}} \mathsf{HD}$ and there is *no* $X \in \mathsf{CN}(\mathcal{O}_{\mathsf{med}})$ such that $\mathsf{Es} \sqsubset_{\mathcal{O}_{\mathsf{med}}} X$ and $X \sqsubset_{\mathcal{O}_{\mathsf{med}}} \mathsf{HD}$.

The advantage of using the concept hierarchy over subsumer sets is that this format is more compact, and it directly supports browsing the subsumption hierarchy by going from a concept name to its direct subsumers or subsumees. The disadvantage is that answering a subsumption question $A \sqsubseteq^?_{\mathcal{O}} B$ then requires to compute the reflexive-transitive closure of the direct and strict subsumption relation.

Two methods are discussed in this section. The next subsection discusses the well-known *enhanced traversal* method [BHN$^+$92, BHN$^+$94] for constructing the concept hierarchy by means of repeated calls to the subsumption testing procedure, while Subsection 4.3.2 proposes a modified technique based on the complete subsumption information from the subsumer sets.

4.3.1 Enhanced traversal method

Most DL systems for expressive DLs use a tableau-based subsumption testing algorithm to compute the concept hierarchy. Apart from optimizing the subsumption testing algorithm itself, one can also optimize computation of the concept hierarchy by reducing the number of required tests to the minimum. In [BHN$^+$92, BHN$^+$94], an efficient algorithm called *enhanced traversal* method for computing the concept hierarchy has been introduced and experimented in the KRIS system.

The enhanced traversal method initializes the DAG to contain $\top, \bot$ and the only edge $\bot \to \top$. For each concept name $A \in \mathsf{CN}(\mathcal{O})$, it inserts A into the DAG by identifying A's immediate and strict subsumers and subsumees in the graph. Thereby edges from each of the subsumees to A and from A to each of the subsumers are added, while existing edges between A's subsumees and subsumers are removed. The enhanced traversal method performs two searches for each insert: the *top search* sweeps down the hierarchy from the $\top$ concept until direct subsumers are identified; the *bottom search* sweeps up the hierarchy from the $\bot$ concept until direct subsumees are identified.

Since we have an algorithm for testing individual subsumption (the goal-directed subsumption algorithm in Section 4.2), we could directly use it in the enhanced traversal method with known optimization techniques from [BHN$^+$92, BHN$^+$94] as well as [Hor97]. These include the following:

- While inserting a concept A, the top search exploits the transitivity of the subsumption relation by propagating negative subsumption results from the preceding tests down the hierarchy. To make this optimization effective, before performing any subsumption test $A \sqsubseteq^?_{\mathcal{O}} B$, a test $A \sqsubseteq^?_{\mathcal{O}} B'$ for all subsumers B' of B need to be tested. If there is a subsumer B' of B such that $A \not\sqsubseteq_{\mathcal{O}} B'$, then it obviously holds that $A \not\sqsubseteq_{\mathcal{O}} B$. That is, a breadth first search is employed.

- The bottom search uses a dual technique to the previous case by propagating negative subsumption results up the hierarchy.

- When the top search is finished, information concerning A's direct subsumers is effectively used to narrow down the candidates for being A's direct subsumees. Precisely, the bottom search can be confined to those concepts already known to be subsumed by a subsumer of A.

- Told and structural subsumption information is also used to avoid certain subsumption tests. To get the full advantage of this, concepts are inserted into the DAG in the definition order, i.e., a concept name A can be inserted into the DAG only after all A's told subsumers have been.

- If the ontology is unfoldable, the bottom search phase can be totally omitted when inserting a primitive concept. A primitive concept has no subsumees in the current DAG (apart from $\bot$) since it can only subsume concepts for which it is a told subsumer, and since all those subsumees will be inserted later due to the definition order.

4.3.2 Using the complete subsumption information

In contrast to the previous approach, the classification algorithm can be used directly to produce the complete subsumption information in the form of subsumer sets. The concept hierarchy can then be extracted from them. In principle, converting subsumer sets into a concept hierarchy is straightforward. For each concept name A, the following can be computed:

- the set $SS(A) := \{B \in S(A) \mid A \notin S(B)\}$ of strict subsumers of A, i.e., subsumers of A that are not equivalent to A;
- the set $DS(A) := SS(A) \setminus \left(\bigcup_{B \in SS(A)} SS(B)\right)$ of direct subsumers of A;
- the set $DS^{-}(A) := \{B \mid A \in DS(B)\}$ of direct subsumees of A.

Clearly, the sets $DS(A)$ and $DS^{-}(A)$, for all concept names A, yield a DAG representation of the concept hierarchy.

However, we do not use this naïve construction since computing the sets $DS(A)$ and $DS^{-}(A)$ is unnecessarily expensive (it needs quadratic time for each A and thus cubic for the whole computation). In fact, it is possible to avoid the direct computation of these sets according to the above definition by using an approach that is inspired by the enhanced traversal method [BHN+94]. Another virtue of our alternative approach is that the potentially costly set operations in the computation of $DS(A)$ are replaced by an efficient marking algorithm.

In order to explain the main idea underlying our algorithm, assume that a partial concept hierarchy with respect to some subset of the concept names has already been computed, and that a concept name A is to be inserted into the DAG. We start by computing the set $SS(A)$ of strict subsumers according to the definition given above. The elements of $S(A) \setminus SS(A)$ are the concepts that are equivalent to A. To find all the *direct* subsumers of A among the elements of $SS(A)$, we proceed as follows. If all elements of $SS(A)$ belong to the already computed DAG, we can find the direct subsumers by using a simple graph traversal algorithm to mark all the strict subsumers of elements of $SS(A)$ in the DAG. The direct subsumers of A are then those elements of $SS(A)$ that are not marked. If there are elements of $SS(A)$ that do not belong to the already computed DAG, then we first insert these elements into the DAG (by issuing recursive calls of the insertion procedure) before inserting A. By following this strategy, we ensure that, when inserting a concept name A into the DAG, all subsumers of A are already in the DAG, but no subsumee of A is. Hence, our algorithm need not compute the direct subsumees explicitly. Instead, it is enough to extend the set of direct subsumees of B by A in case B is found to be a direct subsumer of A. Algorithm 3 shows a pseudo code representation of our algorithm. The sets $\mathsf{parents}(A)$ and $\mathsf{children}(A)$ are used to store the direct subsumers and direct subsumees of A, respectively; while the sets $\mathsf{equivalents}(A)$ are used to store the concepts that are equivalent to A.[5]

A similar algorithm can be obtained if we view the collection of the subsumer sets as a primitive TBox, that is, a set of primitive concept definitions:

$$A \sqsubseteq \bigsqcap_{B_i \in S(A)} B_i$$

[5]Note that the description of the algorithm is a bit sloppy in that we do not distinguish between a concept name and the node in the DAG representing (the equivalence class of) this name.

Algorithm 3 Computing the concept hierarchy from the subsumer sets.

```
Procedure compute-dag
 1: $\mathsf{classified}(\top) := \mathbf{true}$
 2: for each concept name $A \in \mathsf{CN}(\mathcal{O})$ do
 3:   if not $\mathsf{classified}(A)$ then
 4:     $\mathsf{dag\text{-}classify}(A)$
 5:   end if
 6: end for
 7: for each concept $A \in \mathsf{CN}^{\top}(\mathcal{O})$ with $\mathsf{children}(A) = \emptyset$ do
 8:   $\mathsf{children}(A) := \{\bot\}$
 9: end for
Procedure $\mathsf{dag\text{-}classify}(A)$
Input: $A$: concept name
 1: $\mathsf{candidates} := \{\top\}$
 2: for all strict subsumers $B \in S(A) \setminus \{A, \top\}$ do
 3:   if $A \in S(B)$ then
 4:     $\mathsf{classified}(B) := \mathbf{true}$
 5:     $\mathsf{equivalents}(A) := \mathsf{equivalents}(A) \cup \{B\}$
 6:   else
 7:     if not $\mathsf{classified}(B)$ then
 8:       $\mathsf{dag\text{-}classify}(B)$
 9:     end if
10:     $\mathsf{candidates} := \mathsf{candidates} \cup \{B\}$
11:   end if
12: end for
13: $\mathsf{dag\text{-}insert}(A, \mathsf{candidates})$
14: $\mathsf{classified}(A) := \mathbf{true}$
Procedure $\mathsf{dag\text{-}insert}(A, \mathsf{candidates})$
Input: $A$: concept name; candidates: set of concept names
 1: $\mathsf{marked}(X) := \mathbf{false}$ for all $X \in \mathsf{CN}^{\top}(\mathcal{O})$
 2: for all $X \in \mathsf{CN}^{\top}(\mathcal{O})$ and $B \in \mathsf{candidates}$ such that $X \in \mathsf{parents}(B)$ do
 3:   $\mathsf{marked}(X) := \mathbf{true}$
 4: end for
 5: $\mathsf{parents}(A) := \{B \in \mathsf{candidates} \mid \mathsf{marked}(B) = \mathbf{false}\}$
 6: for all $B \in \mathsf{parents}(A)$ do
 7:   $\mathsf{children}(B) := \mathsf{children}(B) \cup \{A\}$
 8: end for
```

for each concept name A. Then, a simplified version of the enhanced traversal method [BHN+94] is used to construct the DAG. Since the TBox is primitive and insertion is done in the definition order, the bottom search can be dispensed with. With a specialized optimization technique, called 'completely defined concepts' in [HT05], we can do away with subsumption testing during the top search phase.

It is not a priori clear which of the two approaches presented in this section is more efficient in practice, given the classification and subsumption algorithms in Section 4.1 and 4.2, respectively. Therefore, we have implemented and evaluated both strategies in CEL, and the empirical evidence shows that the second strategy is at least 50% more efficient than the first. The time required to compute the DAG (Algorithm 3) is negligible compared to the time required to compute the subsumer sets (Algorithm 1 on page 55). For instance, the time ratio is less than 1% in the case of Snomed ct (see Subsection 6.2.1 for the details and results for other ontologies). For this reason, Algorithm 3 is used by the CEL reasoner whenever (part of) the concept hierarchy is demanded. For instance, when direct subsumers or subsumees of a given concept name is queried, or when the whole hierarchy is to be output. (See [Sun05a] for the CEL reference manual.)

4.4 Incremental Classification

Although the classification (thus subsumption) algorithm can be used to query subsumption between concept names, complex subsumptions such as

$$\mathsf{Inflammation} \sqcap \exists \mathsf{has\text{-}location}.\mathsf{Heart} \sqsubseteq^?_{\mathcal{O}_{\mathsf{med}}} \mathsf{HeartDisease} \sqcap \exists \mathsf{has\text{-}state}.\mathsf{NeedsTreatment}$$

cannot be answered directly. First, the ontology $\mathcal{O}_{\mathsf{med}}$ (from Figure 2.2 on page 24) has to be augmented to

$$\begin{aligned}\mathcal{O}'_{\mathsf{med}} \;:= \mathcal{O}_{\mathsf{med}} \cup \{ \;& A \sqsubseteq \mathsf{Inflammation} \sqcap \exists \mathsf{has\text{-}location}.\mathsf{Heart}, \\ & \mathsf{HeartDisease} \sqcap \exists \mathsf{has\text{-}state}.\mathsf{NeedsTreatment} \sqsubseteq B \;\}\end{aligned}$$

with A, B new concept names, and then the subsumption test $A \sqsubseteq^?_{\mathcal{O}'_{\mathsf{med}}} B$ can be carried out to decide the original complex subsumption. Since A, B are new names not occurring in $\mathcal{O}_{\mathsf{med}}$, our complex subsumption holds if, and only if, $A \sqsubseteq_{\mathcal{O}'_{\mathsf{med}}} B$. This approach is effective but inefficient unless only one such complex subsumption is queried for each ontology. Constructing and normalizing the augmented ontology every time each subsumption is tested is not likely to be acceptable in practice, especially when the background ontology is large. For instance, normalization of Snomed ct takes more than one minute.

In this section, the refined algorithm (henceforth referred to as *the original algorithm*) from Subsection 4.1.4 is extended to cater for a *duo-ontology* $\mathcal{O} = (\mathcal{O}_p \cup \mathcal{O}_t)$ with $\mathcal{O}_p$ a *permanent* $\mathcal{EL}^+$ ontology and $\mathcal{O}_t$ a set of *temporary* GCIs. Intuitively, $\mathcal{O}_p$ is the input ontology of which axioms have been read in and processed before, while $\mathcal{O}_t$ contains temporary GCIs that are asserted later. The main purpose is to reuse the information readily made available by the preprocess and classification of $\mathcal{O}_p$. Once $\mathcal{O}_p$ has been classified, the classification of $\mathcal{O}_p \cup \mathcal{O}_t$ should not start from scratch, but rather use the existing classification information w.r.t. $\mathcal{O}_p$ together with the new GCIs from $\mathcal{O}_t$ to do incremental classification.

In our extension, we use *two* sets of the core data structures $\widehat{\mathcal{O}}(\cdot), R(\cdot), S(\cdot)$, but retain a single set of queues $\mathsf{queue}(\cdot)$.[6] The mappings $\widehat{\mathcal{O}}_p, R_p, S_p$ are initialized and populated exactly as in the original algorithm, i.e., $\widehat{\mathcal{O}}_p$ encodes axioms in $\mathcal{O}_p$, and R_p, S_p store subsumption relationships inferred from $\mathcal{O}_p$. Similarly, the mapping $\widehat{\mathcal{O}}_t$ encodes axioms in $\mathcal{O}_t$, but R_t, S_t represent additional inferred subsumptions drawn from $\mathcal{O}_p \cup \mathcal{O}_t$ that are not already present in R_p, S_p, respectively. The extended algorithm is based on the tenet that DLs are monotonic, i.e., $\mathcal{O}_p \models \alpha$ implies $\mathcal{O}_p \cup \mathcal{O}_t \models \alpha$. There may be an additional consequence β such that $\mathcal{O}_p \not\models \beta$ but $\mathcal{O}_p \cup \mathcal{O}_t \models \beta$. The extended algorithm stores such a consequence β in the separate set of data structures R_t, S_t. Analogously to the original algorithm, queue entries are repeatedly fetched and processed until all queues are empty. Instead of the procedures process, process-new-edge and process-bottom, we use the extended versions for duo-ontology classification as outlined in Algorithm 4.

The extended algorithm's behavior is identical to that of Algorithm 1 if $\mathcal{O}_p$ has not been classified. In particular, $\widehat{\mathcal{O}}_p(\cdot) \cup \widehat{\mathcal{O}}_t(\cdot)$ here is equivalent to $\widehat{\mathcal{O}}(\cdot)$ in Algorithm 1 given that $\mathcal{O} = (\mathcal{O}_p \cup \mathcal{O}_t)$. Since no classification has taken place, $S_p(A) = R_p(r) = \emptyset$ for every concept name A and role name r. Initialization and processing of queues are done in the same manner with the only difference that inferred consequences are now stored in R_t and S_t.

If $\mathcal{O}_p$ has been classified (thus, S_p, R_p have been populated), then proper initialization has to be done w.r.t. previously inferred consequences (i.e., S_p, R_p) and new GCIs (i.e., $\widehat{\mathcal{O}}_t$). To this end, we initialize the data structures by setting:

- for each role name $r \in \mathsf{RN}(\mathcal{O})$, $R_t(r) := \emptyset$;
- for each *old* concept name $A \in \mathsf{CN}(\mathcal{O}_p)$, $S_t(A) := \emptyset$ and $\mathsf{queue}(A) := \bigcup_{X \in S_p(A)} \widehat{\mathcal{O}}_t(X) \ \cup \ \bigcup_{\{(A,B) \in R_p(r), X \in S_p(B)\}} \widehat{\mathcal{O}}_t(\exists r.X)$;
- for each *new* concept name $A \in \mathsf{CN}(\mathcal{O}_t) \backslash \mathsf{CN}(\mathcal{O}_p)$, $S_t(A) := \{A, \top\}$ $\mathsf{queue}(A) := \widehat{\mathcal{O}}_t(A) \cup \widehat{\mathcal{O}}(\top) \cup \bigcup_{r \text{ with } \mathcal{O} \models \epsilon \sqsubseteq r} (\widehat{\mathcal{O}}_t(\exists r.A) \ \cup \ \widehat{\mathcal{O}}(\exists r.\top))$.

After initialization, queue processing is carried out by Algorithm 4 until all the queues are empty. Observe the structural analogy between these procedures and the original ones in Algorithm 1 on page 55. Observe also the key difference: information is always retrieved from both sets of data structures, e.g., $S_p(A) \cup S_t(A)$ in line 1, while modifications are only made to the temporary set of data structures, e.g., $S_t(A) := S_t(A) \cup \{B\}$ in line 2.

The correctness of this algorithm can be shown following the correctness proof's structures of the original algorithm w.r.t. additional subsumption consequences obtained during incremental classification.

Theorem 33 (Correctness of Algorithm 4). *Let $\mathcal{O} = (\mathcal{O}_p \cup \mathcal{O}_t)$ be a duo-ontology, and S_p, R_p be the results after the original algorithm terminates on $\mathcal{O}_p$. Then, the duo-ontology classification algorithm (Algorithm 4), applied to $\mathcal{O}_t$, incrementally classifies*

[6]Here, the optimization technique in Subsection 4.1.5 that uses multiple sets of queues is not taken into account.

Algorithm 4 The duo-ontology classification algorithm.

```
Procedure process-duo($A, X$)
Input: $A$: concept name; $X$: queue entry;
 1: if $X = \mathbf{B} \rightarrow B$, $\mathbf{B} \subseteq S_p(A) \cup S_t(A)$ and $B \notin S_p(A) \cup S_t(A)$ then
 2:   if $B = \bot$ then
 3:     process-bottom-duo($A$)
 4:   else
 5:     $S_t(A) := S_t(A) \cup \{B\}$
 6:     queue($A$) := queue($A$) $\cup\ \widehat{\mathcal{O}}_p(B) \cup \widehat{\mathcal{O}}_t(B)$
 7:     for all concept names $A'$ and role names $r$ with $(A', A) \in R_p(r) \cup R_t(r)$ do
 8:       queue($A'$) := queue($A'$) $\cup\ \widehat{\mathcal{O}}_p(\exists r.B) \cup \widehat{\mathcal{O}}_t(\exists r.B)$
 9:     end for
10:   end if
11: end if
12: if $X = \exists r.B$ and $(A, B) \notin R_p(r) \cup R_t(r)$ then
13:   if $\bot \in S(B)$ and $\bot \notin S(A)$ then
14:     process-bottom-duo($A$)
15:   end if
16:   process-new-edge-duo($A, r, B$)
17: end if

Procedure process-new-edge-duo($A, r, B$)
Input: $A, B$: concept names; $r$: role name;
 1: for all role names $s$ with $\mathcal{O}_p \models r \sqsubseteq s$ do
 2:   $R_t(s) := R_t(s) \cup \{(A, B)\}$
 3:   queue($A$) := queue($A$) $\cup \bigcup_{\{B' \mid B' \in S_p(B) \cup S_t(B)\}}(\widehat{\mathcal{O}}_p(\exists s.B') \cup \widehat{\mathcal{O}}_t(\exists s.B'))$
 4:   for all concept name $A'$ and role names $u, v$ with $u \circ s \sqsubseteq v \in \mathcal{O}_p$ and
      $(A', A) \in R_p(u) \cup R_t(u)$ and $(A', B) \notin R_p(v) \cup R_t(v)$ do
 5:     process-new-edge-duo($A', v, B$)
 6:   end for
 7:   for all concept name $B'$ and role names $u, v$ with $s \circ u \sqsubseteq v \in \mathcal{O}_p$ and
      $(B, B') \in R_p(u) \cup R_t(u)$ and $(A, B') \notin R_p(v) \cup R_t(v)$ do
 8:     process-new-edge-duo($A, v, B'$)
 9:   end for
10: end for

Procedure process-bottom-duo($A$)
Input: $A$: concept name;
 1: $S_t(A) := S_t(A) \cup \{\bot\}$
 2: for all concept names $A'$ and role names $r$ with $(A', A) \in R_p(r) \cup R_t(r)$ such that
    $\bot \notin S_p(A') \cup S_t(A')$ do
 3:   process-bottom-duo($A'$)
 4: end for
```

$\mathcal{O}_t$ against $\mathcal{O}_p$ (i.e., classifies $\mathcal{O}$) in time polynomial in the size of $\mathcal{O}$. That is, after termination of the algorithm, we have that, for all concept names A, B in $\mathcal{O}$:

$$\{B, \bot\} \cap (S_p(A) \cup S_t(A)) \neq \emptyset \text{ if, and only if, } A \sqsubseteq_{\mathcal{O}} B.$$

Proof. *(Sketch)*
Termination in polynomial time is straightforward and follows immediately from Lemma 28 on page 56. In fact, the duo-ontology classification algorithm adds elements to $S_t(\cdot)$ and $R_t(\cdot)$ and never deletes existing elements. Additionally, each such addition induces only polynomially many additions to the queues.

Soundness is shown along the same line as in Lemma 29 on page 56 where **INV1** and **INV2** are modified to state the properties w.r.t. $S_p(\cdot)$ and $R_p(\cdot)$, respectively; and where two additional invariants are introduced:

INV6 If $B \in S_t(A)$, then $A \sqsubseteq_{\mathcal{O}} B$.

INV7 If $(A, B) \in R_t(r)$, then $A \sqsubseteq_{\mathcal{O}} \exists r.B$

to capture the new set of data structures. Observe that **INV1** and **INV6** together form the "only if" part (soundness) of the theorem. Since Algorithm 4 never manipulates $S_p(\cdot)$ and $R_p(\cdot)$, **INV1** and **INV2** are preserved throughout the computation. Modifications to $S_t(\cdot)$, $R_t(\cdot)$ and queue$(\cdot)$ can be shown to preserve all the other invariants in an analogous way as in Lemma 29. Here, we only demonstrate the most interesting case: initialization of queue(A) with A an old concept name in $\mathsf{CN}(\mathcal{O}_p)$. We make a case distinction according to the source of the new queue entries.

- An element $\mathbf{B} \rightarrow B$ from $\widehat{\mathcal{O}}_t(X)$ with $X \in S_p(A)$ is added to queue(A). Since $A \sqsubseteq_{\mathcal{O}} X$ due to $X \in S_p(A)$ and **INV1**, it suffices to show that $\bigwedge_{B_i \in \mathbf{B}}(X \sqsubseteq_{\mathcal{O}} B_i)$ implies $X \sqsubseteq_{\mathcal{O}} B$. But, this is the case because there exists (up to commutativity of $\sqcap$) a GCI $X \sqcap B_1 \sqcap \cdots \sqcap B_n \sqsubseteq B$ in $\mathcal{O}_t \subseteq \mathcal{O}$ and $X \sqsubseteq_{\mathcal{O}} X$. Thus, **INV4** is preserved.

- An element $\exists r.B$ from $\widehat{\mathcal{O}}_t(X)$ with $X \in S_p(A)$ is added to queue(A). Thus, there must exist a GCI $X \sqsubseteq \exists r.B$ in $\mathcal{O}_t \subseteq \mathcal{O}$. Since $A \sqsubseteq_{\mathcal{O}} X$ due to $X \in S_p(A)$ and **INV1**, it holds that $A \sqsubseteq_{\mathcal{O}} \exists r.B$, preserving **INV3**.

- An element Y from $\widehat{\mathcal{O}}_t(\exists r.X)$ with $X \in S_p(B)$ and $(A, B) \in R_p(r)$ is added to queue(A). Thus, **INV1** and **INV2**, alongside $X \in S_p(B)$ and $(A, B) \in R_p(r)$, imply that $B \sqsubseteq_{\mathcal{O}} X$ and $A \sqsubseteq_{\mathcal{O}} \exists r.B$, respectively. There must also exist a GCI $\exists r.X \sqsubseteq Y$ in $\mathcal{O}_t \subseteq \mathcal{O}$. Putting together, it holds that $A \sqsubseteq_{\mathcal{O}} \exists r.B \sqsubseteq_{\mathcal{O}} \exists r.X \sqsubseteq_{\mathcal{O}} Y$, thus preserving (the $\mathbf{B} = \emptyset$ case of) **INV4**.

The completeness part of the theorem can be shown in the same way as in Lemma 30 on page 58 using the completeness result of the abstract algorithm given in [BBL05].

❑

In our example, we set $\mathcal{O}_p$ to $\mathcal{O}_{\mathsf{med}}$ and $\mathcal{O}_t$ to the set of the two new GCIs. We can run the extended algorithm on $\mathcal{O}_p \cup \mathcal{O}_t$ and reuse existing information in S_p and R_p, if any. After termination, our complex subsumption boils down to the set membership test $B \in^? S_p(A) \cup S_t(A) = S_t(A)$. (Recall that A is a new concept name not occurring in $\mathcal{O}_p$, and thus $S_p(A) = \emptyset$.) To answer next subsumption queries, only $\mathcal{O}_t, R_t, S_t$ and queue need to be initialized, leaving the background ontology $\mathcal{O}_p$ and possibly its classification information R_t, S_t intact.

Interestingly, this algorithm can be used effectively in certain scenarios of incremental classification. Consider $\mathcal{O}_p$ as a well-developed, permanent ontology, and $\mathcal{O}_t$ as a small set of temporary axioms currently being authored. Obviously, if the permanent ontology is large, it would be impractical to reclassify from scratch every time some new axioms are to be added. Algorithm 4 incrementally classifies $\mathcal{O}_t$ against $\mathcal{O}_p$ and its classification information. If the inferred consequences are satisfactory, the temporary axioms can be committed to the permanent ontology by merging the two sets of data structures. Otherwise, axioms in $\mathcal{O}_t$ and their inferred consequences can be easily retracted, since these are segregated from $\mathcal{O}_p$ and its consequences. To be precise, we simply dump the values of $\mathcal{O}_t(\cdot), R_t(\cdot)$ and $S_t(\cdot)$, when the temporary axioms are retracted. This incremental reasoning scenario has also been considered in [Law08], where Snomed is used as a reference knowledge base, and where additional definitions representing mappings from concepts in another terminology to concepts in Snomed are defined and incorporated into the reference knowledge base. Since Snomed is large and non-versatile in such a scenario, it is sensible to classify it only once and reuse its classification results for subsequent reasoning w.r.t. a relatively small and versatile user-defined ontology.

4.5 Reasoning with Individuals

In [Bra04b], Brandt proposed a polynomial classification algorithm for $\mathcal{ELH}$ with GCIs and later extended it in [Bra04a] to deal with instance checking problem.[7] The idea is to extend the approach to deciding subsumption by means of completing the subsumption graph to ABox individuals. The algorithm completes the extended mapping S which also maps every individual a to a set $S(a)$ of concepts. The intuition is that A belongs to $S(a)$ if, and only if, the instance checking $A(a)$ holds in $\mathcal{O}$. In this section, we exploit this technique in the abstract classification algorithm.

Recall from Definition 5 on page 21 that an ABox $\mathcal{A}$ (the assertional part of an ontology $\mathcal{O}$) is a set of assertions of the forms $C(a)$ and $r(a, b)$ with C a concept description, r a role name, and a, b individuals. Before the core algorithm can be started, we need to perform a couple of preprocessing steps to the ontology.

[7]The algorithm in [Bra04a, Bra04b] differs from the one presented in Subsection 4.1.3 in that it maintains only $S(\cdot)$ but not $R(\cdot)$ mapping. Role relations are not made explicit in the same way as in **CR2**, whereas interaction between axioms of the forms $A \sqsubseteq \exists r.B$ and $\exists s.A' \sqsubseteq B'$ is taken care of by a rule that is essentially the combination of **CR2**, **CR3** and **CR5** in Figure 4.2 on page 52.

4.5.1 Preprocessing the ABox

We extend the notion of ontology normal form to ABox assertions by admitting only atomic concept assertions $A(a)$ and role assertions $r(a, b)$, where $A \in \mathsf{CN}(\mathcal{O})$, $r \in \mathsf{RN}(\mathcal{O})$ and $a, b \in \mathsf{Ind}(\mathcal{O})$. Normalization rules depicted in Figure 4.1 on page 48 can be extended by adding the following rule:

NR1-8 $\quad C(a) \quad \rightsquigarrow \quad A(a),\ A \sqsubseteq C$

to deal with ABox concept assertions. It is not hard to see that an arbitrary $\mathcal{EL}^+$ ontology $\mathcal{O}$ can be transformed in linear time to $\widetilde{\mathcal{O}}$ in normal form without complex concept assertions such that (i) the size of $\widetilde{\mathcal{O}}$ is linear in that of $\mathcal{O}$, (ii) subsumption $A \sqsubseteq B$ with $A, B \in \mathsf{CN}(\mathcal{O})$ is preserved, and (iii) instance checking $A(a)$ with $A \in \mathsf{CN}(\mathcal{O})$ and $a \in \mathsf{Ind}(\mathcal{O})$ is preserved.

By reducing out range restrictions, role assertions in the ABox also needs to be taken into account in addition to the reduction described in Subsection 4.1.2. For each role assertion $r(a, b)$ in $\mathcal{O}$ such that $\mathsf{range}_{\mathcal{O}}(r) \neq \emptyset$, we add a new concept assertion $A(b)$ for each $A \in \mathsf{range}_{\mathcal{O}}(r)$. Again, it is not difficult to see that the extended reduction is correct, i.e., it preserves subsumption and instance checking.

In what follows, we assume without loss of generality that the TBox is in normal form (Definition 24 on page 46) without range restrictions and the ABox contains no complex concept assertion.

4.5.2 Extending the algorithm with individuals

In order to reason with individuals, we extend the S and R mappings to take into account ABox individuals as follows:

- a mapping S assigning to each element x of $\mathsf{CN}^\top(\mathcal{O}) \cup \mathsf{Ind}(\mathcal{O})$ a subset $S(x)$ of $\mathsf{CN}(\mathcal{O}) \cup \{\top, \bot\}$, and
- a mapping R assigning to each element r of $\mathsf{RN}(\mathcal{O})$ a binary relation $R(r)$ over $\mathsf{CN}^\top(\mathcal{O}) \cup \mathsf{Ind}(\mathcal{O})$.

In terms of completion graph, the set of vertices V is extended to $\mathsf{CN}^\top(\mathcal{O}) \cup \mathsf{Ind}(\mathcal{O})$. Intuitively, the additional elements in the mappings represent inferred instance checking statements in the sense that

- $A \in S(a)$ implies $\mathcal{O} \models A(a)$,
- $(a, A) \in R(r)$ implies $\mathcal{O} \models \exists r.A(a)$, and
- $(a, b) \in R(r)$ implies $\mathcal{O} \models r(a, b)$.

When the algorithm terminates, the first statement is guaranteed to be complete, i.e., if $\mathcal{O} \models A(a)$, then we have $A \in S(a)$.

Similar to the classification algorithm, it starts by initializing the two mappings and then exhaustively applying the completion rules until no more rule applies. Initialization of $S(A)$ with $A \in \mathsf{CN}(\mathcal{O})$ remains unchanged, while $S(a)$ is initialized with

Algorithm 5 The goal-directed instance checking algorithm.

Procedure instance?($B_0(a)$)
Input: a: individual; B_0: concept name
Output: 'positive' or 'negative' answer to the query

```
1: activate($a$)
2: while not empty(queue($A$)) for some $A \in \mathsf{CN}^\top(\mathcal{O}) \cup \mathsf{Ind}(\mathcal{O})$ do
3:    $X \leftarrow$ fetch(queue($A$))
4:    if goal-directed-process($A, X, a, B_0$) = 'positive' then
5:       return 'positive'
6:    end if
7: end while
8: return 'negative'
```

$\{A|A(a) \in \mathcal{A}\} \cup \{\top\}$, for each individual $a \in \mathsf{Ind}(\mathcal{O})$. For each role name $r \in \mathsf{RN}(\mathcal{O})$, let $r^\mathcal{A} = \{(a,b) \mid r(a,b) \in \mathcal{A}\}$. Then, we initialize the R mapping by setting:

$$R(r) := \begin{cases} r^\mathcal{A} \cup \{(x,x) \mid x \in \mathsf{CN}^\top(\mathcal{O}) \cup \mathsf{Ind}(\mathcal{O})\} & \text{if } \mathcal{O} \models \epsilon \sqsubseteq r; \\ r^\mathcal{A} & \text{otherwise.} \end{cases}$$

The completion rules shown in Figure 4.2 on page 52 are intact and can be used directly, where the variables X, Y, Z now range over $\mathsf{CN}^\top(\mathcal{O}) \cup \mathsf{Ind}(\mathcal{O})$. Observe that the pairs in $R(\cdot)$ can be in one of the forms (a,b), (A,B) or (a,A), where a,b are individuals and A,B concept names. The last type, which links an individual to a concept name, can be generated by **CR2** and also reproduced by **CR5** and **CR6**.

Theorem 34 (Correctness). *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology in normal form without range restrictions and complex concept assertions. The extended algorithm applied to $\mathcal{O}$ terminates in time polynomial in the size of $\mathcal{O}$ and, after termination, we have that, for all individuals a and concept names A in $\mathcal{O}$:*

$$\{A, \bot\} \cap S(a) \neq \emptyset \textit{ if, and only if, } \mathcal{O} \models A(a).$$

Techniques developed in the refined classification algorithm (see Subsection 4.1.4) can also be applied here with queue($\cdot$) extended to ABox individuals. While queue(A) remains intact for all concept names $A \in \mathsf{CN}(\mathcal{O})$, we initialize queue($a$) with

$$\bigcup_{A \in S(a)} \widehat{\mathcal{O}}(A) \quad \cup \quad \bigcup_{(a,b)\in R(r),\, X \in S(b)} \widehat{\mathcal{O}}(\exists r.X)$$

Correctness of the refined version of the extended algorithm with individuals can be proved in a parallel way as in Theorem 33, where information made explicit by the initialization (i.e., $A \in S(a)$ and $(a,b) \in R(r)$) correspond to classification information of the permanent ontology (i.e., S_p and R_p).

Given a specific instance checking query $\mathcal{O} \models^? A(a)$, one can employ the goal-directed approach (see Section 4.2) to the algorithm presented in this section to answer the query in a goal-directed fashion. More precisely, the procedure goal-directed-process (see Algorithm 2 on page 64) is extended such that the variables A, A_0 not only range over concept names but also individuals. To this end, we outline the procedure instance? in Algorithm 5 which is based on the procedure subsumes? (see Algorithm 2 on page 64).[8]

Similarly, the simplified version of the enhanced traversal method (see Subsection 4.3.2) can be used to compute the realization hierarchy (see Definition 16 on page 28). Observe that $S : \mathsf{CN}^{\top}(\mathcal{O}) \cup \mathsf{Ind}(\mathcal{O}) \rightarrow \mathsf{CN}(\mathcal{O}) \cup \{\top, \bot\}$ implies that individuals never occur in any subsumer set. As a result, all the individuals occur in the bottom most above $\bot$ in the hierarchy, i.e., $\mathsf{children}(a) = \{\bot\}$ for all individuals a. Moreover, given an individual a, $\mathsf{parents}(a)$ represents the set of most specific concept names to which it belongs, whereas, given a concept name A, those individuals underneath A in the hierarchy are all its instances.

[8]Like the subsumption algorithm, this is implemented in the CEL reasoner and used whenever instance checking is asked prior to realization. See [Sun05a] for the CEL reference manual.

Chapter 5

Techniques for Supplemental Reasoning

In this chapter, techniques for supplemental reasoning (i.e., extracting small modules and finding justifications) are presented. Modularization has several potential applications both in optimization of automated reasoning and in ontology engineering and usage. A usage scenario of modularization in ontology reuse and segmentation has been given in Subsection 3.3.2, while its use in optimization of reasoning will be discussed later in this chapter. Section 5.1 presents a new type of module based on reachability and a number of interesting properties. In particular, we show that modularization in fact does not help speed up classical reasoning of subsumption if the goal-directed subsumption algorithm in Section 4.2 is employed.

The second half of the chapter is dedicated to axiom pinpointing. Section 5.2.1 shows some inherent complexity in the problem of finding all justifications. Subsection 5.2.2 and 5.2.3 discuss the black-box and glass-box approaches, respectively. Several algorithms for computing a single and all justifications are described. Finally, we propose in Subsection 5.2.4 two combined methods for computing a single justification and a method for computing all justifications. In particular, we propose a *modularization-based* approach to axiom pinpointing, in which module extraction plays an important role.

In what follows, we focus attention on the terminological component (TBox) of an $\mathcal{EL}^+$ ontology. This is not a real limitation since individuals can be encoded as special least concepts, and ABox assertions as GCIs in the TBox part of an ontology (see Section 4.5). By doing so, all the techniques presented in the following immediately extend to ontologies with an assertional component as well.

It is worth mentioning other techniques for supplemental reasoning that are beyond the scope of the present dissertation. These include the computation of least common subsumers (lcs) and most specific concepts (msc) [Küs00], which have proven useful in supporting bottom-up construction of the knowledge base. The problem of computing lcs has been studied thoroughly in [Tur07, Küs00] and that of mcs in [Bra06, Küs00].

5.1 Modularization

By Definition 21 on page 30, a module $\mathcal{O}'$ is any subset of the ontology $\mathcal{O}$ that preserves the statement σ of interest or all statements under the signature $\mathbf{S}$ of interest. Obviously, the whole ontology $\mathcal{O}$ is a module for any statement or signature in itself, but it is vacuously uninteresting since it does not give us additional information concerning the ontology relative to a specified context (i.e., through a statement or a signature) of interest.

The notion of (deductive) conservative extension [GLWZ06] could be used as a sufficient condition for extracting a module, since if $\mathcal{O}$ is a conservative extension of $\mathcal{O}'$ w.r.t. a signature $\mathbf{S}$, then $\mathcal{O}'$ is an $\mathbf{S}$-module in $\mathcal{O}$. Unfortunately, it is too expensive to decide conservativity (e.g., ExpTime-complete already in $\mathcal{EL}$ with GCIs [LW07]). Therefore, semantic modularization is highly complex in general with the only exception of the DL $\mathcal{EL}$ with acyclic TBox, in which it is tractable to extract a semantic module [KLWW08]. For this reason, several techniques for syntactic module extraction have been proposed [NM03, SR06, CHKS07, Sun08], some of which are logic-based and some are not.

In the first subsection, we introduce a new kind of module, called *reachability-based module*, that is motivated by known optimization techniques adopted by DL reasoners, including CEL. Also, we propose an algorithm for extracting such a module given an ontology and a signature as input and show some interesting properties of modules of this kind which are required to prove subsequent theorems. In Subsection 5.1.2, we establish some connections between reachability-based modules and subsumption. In particular, it is shown that the modules of this kind are *strong subsumption modules* (see Definition 23 on page 31) which is essential in our modularization-based axiom pinpointing (see Subsection 5.2.4). In Subsection 5.1.3, other extraction methods are discussed and compared to the reachability-based approach. Particularly, we show that the reachability-based module is equivalent to the minimal module based on syntactic locality modulo the DL $\mathcal{EL}^+$.

Also, experiments on realistic biomedical ontologies have been performed. The experimental results not only prove practical usability (time required to extract the module) and usefulness (size of the extracted module) of our module extraction algorithm but also reveal an insight into the ontology structure that reflects its complexity in terms of classification. These empirical results are described in Subsection 6.2.4.

5.1.1 Reachability-based modules

To facilitate defining the forthcoming notions of reachability w.r.t. an ontology, a *module normal form* is introduced.

Definition 35 (Module normal form). An $\mathcal{EL}^+$ ontology $\mathcal{O}$ is in *module normal form* if all of its axioms are either concept inclusions $C \sqsubseteq D$, role inclusions $r_1 \circ \cdots \circ r_n \sqsubseteq s$ or range restrictions $\mathsf{range}(r) \sqsubseteq D$, where $\bot$ does not occur in C. ◇

Note that an ontology in the $\mathcal{EL}^+$ normal form defined in Section 4.1.1 is also in module normal form, but not vice versa. Moreover, part of the normalization rules in Figure 4.1 on page 48 (precisely, **NR1-1**, **NR1-3**, **NR1-4**, **NR1-5** and **NR1-6**) can be reused here to linearly transform any ontology into a subsumption-preserving one in module normal form. In ontology re-use scenario, it is an understandable requirement that a module consists only of *original* axioms. That is, axioms in the module in the normal form have to be mapped to their sources, but this can be easily done by keeping the source axioms and the mapping between the source and the normalized axioms. For the rest of this subsection, we thus assume without loss of generality that $\mathcal{EL}^+$ ontologies are in module normal form.

The mentioned optimization techniques are used to heuristically determine obvious subsumption and non-subsumption relationships. They can be understood as the reachability problem in a directed graph, considering concept names as nodes and explicit subsumption relationships as edges in the graph.

Definition 36 (Tight/loose reachability). Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology, A, B concept names in $\mathcal{O}$. The tight (loose) reachability graph $\mathcal{G}_t(\mathcal{O})$ ($\mathcal{G}_l(\mathcal{O})$) for $\mathcal{O}$ is a tuple (V_t, E_t) ((V_l, E_l)) with $V_t = \mathsf{CN}(\mathcal{O})$ ($V_l = \mathsf{CN}(\mathcal{O})$) and E_t (E_l) the smallest set containing an edge $\langle A, B\rangle$ if $A \sqsubseteq D \in \mathcal{O}$ s.t. B is a conjunct in D (if $C \sqsubseteq D \in \mathcal{O}$ s.t. A occurs in C and B occurs in D).

We say that B is *tightly reachable* (*loosely reachable*) from A in $\mathcal{O}$ if there is a path from A to B in $\mathcal{G}_t(\mathcal{O})$ ($\mathcal{G}_l(\mathcal{O})$). $\diamond$

It is easy to see that, given an $\mathcal{EL}^+$ ontology $\mathcal{O}$, $\mathcal{G}_t(\mathcal{O})$ and $\mathcal{G}_l(\mathcal{O})$ can be constructed in linear time in the size of $\mathcal{O}$. Observe that B is tightly reachable from A in $\mathcal{O}$ implies $A \sqsubseteq_{\mathcal{O}} B$, while $A \sqsubseteq_{\mathcal{O}} B$ implies that B is loosely reachable from A in $\mathcal{O}$.[1] The reachability heuristic for positive subsumption is also known as *told subsumers*, which essentially is the reflexive and transitive closure of the explicitly stated subsumption. DL systems benefit from this information to completely avoid certain kind of subsumption test, as well as to classify concepts in the subsumption order.

In the DL $\mathcal{EL}^+$, the reachability heuristic for negative subsumptions can be exploited in module extraction. To achieve this, the loose reachability graph $\mathcal{G}_l(\mathcal{O})$ for $\mathcal{O}$ needs to be extended in a straightforward way to cover all the symbols in $\mathcal{O}$, i.e., also role names. Precisely, we define the extension as $\mathcal{G}'_l(\mathcal{O}) := (V'_l, E'_l)$ with $V'_l = \mathsf{Sig}(\mathcal{O})$ and $\langle x, y\rangle \in E'_l$ if there is an axiom $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}$ (i.e., a concept or role inclusion, or a range restriction) such that $x \in \mathsf{Sig}(\alpha_L)$ and $y \in \mathsf{Sig}(\alpha_R)$. The *module based on the extended loose reachability* for $\mathbf{S}$ in an ontology $\mathcal{O}$ (denoted by $\mathcal{O}^{\mathsf{loose}}_{\mathbf{S}}$) can be extracted as follows: construct $\mathcal{G}'_l(\mathcal{O})$, extract all the paths from a symbol $x \in \mathbf{S}$ in $\mathcal{G}'_l(\mathcal{O})$, and finally, accumulate axioms responsible for the edges in those paths. However, this kind of module is relatively large, and many axioms are often irrelevant. In our example ontology $\mathcal{O}_{\mathsf{med}}$, the definition for $\mathsf{HeartDisease}$ would be extracted as part of the module based on the extended loose reachability for $\mathbf{S} = \{\mathsf{Appendicitis}\}$ in $\mathcal{O}_{\mathsf{med}}$ since the concept $\mathsf{Disease}$ is reachable from $\mathsf{Appendicitis}$ via $\mathsf{Inflammation}$ in $\mathcal{G}'_l(\mathcal{O}_{\mathsf{med}})$. The fact that $\mathsf{HeartDisease}$'s definition also involves the symbols $\mathsf{has\text{-}location}$ and Heart is simply not taken into consideration here. In fact, the module for any

[1]To see the latter, assume that B is *not* loosely reachable from A in $\mathcal{O}$. Then, a counter model $\mathcal{I}$ of $\mathcal{O}$ can be constructed in a similar fashion as in the proof of Point 5 in Proposition 38 such that $A^{\mathcal{I}} \backslash B^{\mathcal{I}} \neq \emptyset$.

kind of inflammatory morphology would comprise the definition for heart disease as well.

In order to rule out such irrelevant axioms from the module, we need to make our reachability notion stronger by taking into account *all* the symbols occurring on the left-hand side of axioms. Precisely, the ontology $\mathcal{O}$ is viewed as a directed hypergraph [AFF01] where each inclusion axiom $\alpha_L \sqsubseteq \alpha_R$ in $\mathcal{O}$ essentially specifies a collection of hyperedges from the connected node $\mathsf{Sig}(\alpha_L)$ to each of the symbol in $\mathsf{Sig}(\alpha_R)$. For example, the following half of axiom α_9 in $\mathcal{O}_{\mathsf{med}}$ (see Figure 2.2 on page 24)

$$\mathsf{Disease} \sqcap \exists \mathsf{has\text{-}location}.\mathsf{Heart} \sqsubseteq \mathsf{HeartDisease}$$

represents a hyperedge $\langle\, \{\mathsf{Disease}, \mathsf{has\text{-}location}, \mathsf{Heart}\}\,,\ \mathsf{HeartDisease}\,\rangle$. Given a node x in the hypergraph, the symbol $\mathsf{HeartDisease}$ is reachable from x if *all* symbols participating in the connected nodes, i.e., $\mathsf{Disease}$, $\mathsf{has\text{-}location}$ and Heart, are reachable from x. Considering the example above, the concept $\mathsf{Appendicitis}$, as well as most inflammatory morphology concepts, does not reach $\mathsf{HeartDisease}$ since it does not reach Heart. Hence, the definition for $\mathsf{HeartDisease}$ is not extracted as part of the refined module. To this end, we formally define our *reachability-based modules* as follows:

Definition 37 (Reachability-based modules). Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology and $\mathbf{S} \subseteq \mathsf{Sig}(\mathcal{O})$ a signature. The *set of* $\mathbf{S}$*-reachable names in* $\mathcal{O}$ is inductively defined:

- x is $\mathbf{S}$-reachable in $\mathcal{O}$, for every $x \in \mathbf{S}$;
- for all inclusion axioms $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}$, if x is $\mathbf{S}$-reachable in $\mathcal{O}$ for every $x \in \mathsf{Sig}(\alpha_L)$ then y is $\mathbf{S}$-reachable in $\mathcal{O}$ for every $y \in \mathsf{Sig}(\alpha_R)$.

We call an axiom $\alpha_L \sqsubseteq \alpha_R$ $\mathbf{S}$*-reachable in* $\mathcal{O}$ if every element of $\mathsf{Sig}(\alpha_L)$ is $\mathbf{S}$-reachable in $\mathcal{O}$. The *reachability-based module for* $\mathbf{S}$ *in* $\mathcal{O}$, denoted by $\mathcal{O}^{\mathsf{reach}}_{\mathbf{S}}$, consists of all $\mathbf{S}$-reachable axioms from $\mathcal{O}$. ◇

If $\mathbf{S} = \{A\}$ ($\mathbf{S} = \{r\}$) is a singleton signature consisting only of the concept name A (role name r), we simply write A-reachable (r-reachable) and $\mathcal{O}^{\mathsf{reach}}_{A}$ ($\mathcal{O}^{\mathsf{reach}}_{r}$). Intuitively, x is y-reachable in $\mathcal{O}$ means that y syntactically refers to x, either directly or indirectly via axioms in $\mathcal{O}$. If x, y are concept names, then the reachability suggests a potential subsumption relationship $y \sqsubseteq_{\mathcal{O}} x$. Note, in particular, that axioms of the forms $\top \sqsubseteq D$ and $\epsilon \sqsubseteq r$ in $\mathcal{O}$ are vacuously reachable from any symbol in $\mathsf{Sig}(\mathcal{O})$ because $\mathsf{Sig}(\top) = \mathsf{Sig}(\epsilon) = \emptyset$; therefore, they occur in every reachability-based module.

In our example, $\mathcal{O}^{\mathsf{reach}}_{\mathsf{Appendicitis}}$ contains axioms $\alpha_1, \alpha_5, \alpha_8, \alpha_{10}$ and α_{12}–α_{14}. Observe that, in contrast to the module based on the extended loose reachability, obviously irrelevant axioms like α_9 and α_{11} (i.e., axioms related to $\mathsf{HeartDisease}$) are not part of the reachability-based module. We now show some properties of reachability and reachability-based modules that are essential for establishing the subsequent results:

Proposition 38 (Properties of reachability and $\mathcal{O}^{\mathsf{reach}}_{\mathbf{S}}$). *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology, $\mathbf{S}, \mathbf{S}_1, \mathbf{S}_2 \subseteq \mathsf{Sig}(\mathcal{O})$ signatures, x, y, z symbols in $\mathsf{Sig}(\mathcal{O})$, and A, B concept names in $\mathsf{CN}(\mathcal{O})$. Then, the following properties hold:*

1. *If* $\mathbf{S}_1 \subseteq \mathbf{S}_2$*, then* $\mathcal{O}^{\mathsf{reach}}_{\mathbf{S}_1} \subseteq \mathcal{O}^{\mathsf{reach}}_{\mathbf{S}_2}$*.*
2. *If* x *is* y*-reachable and* y *is* z*-reachable, then* x *is* z*-reachable.*
3. *If* x *is* y*-reachable in* $\mathcal{O}$*, then* $\mathcal{O}^{\mathsf{reach}}_{x} \subseteq \mathcal{O}^{\mathsf{reach}}_{y}$
4. $x \in \mathbf{S} \cup \mathsf{Sig}(\mathcal{O}^{\mathsf{reach}}_{\mathbf{S}})$ *if, and only if,* x *is* $\mathbf{S}$*-reachable in* $\mathcal{O}$*.*
5. *If* B *is* not A*-reachable in* $\mathcal{O}$*, then* $A \not\sqsubseteq_{\mathcal{O}} B$ *(unless* A *is unsatisfiable w.r.t.* $\mathcal{O}$*).*

Proof.
To show Point 1, it is enough to show, for each axiom $\alpha = (\alpha_L \sqsubseteq \alpha_R) \in \mathcal{O}$, that $\alpha \in \mathcal{O}^{\mathsf{reach}}_{\mathbf{S}_1}$ implies $\alpha \in \mathcal{O}^{\mathsf{reach}}_{\mathbf{S}_2}$. By definition, it follows from $\alpha \in \mathcal{O}^{\mathsf{reach}}_{\mathbf{S}_1}$ that x is $\mathbf{S}_1$-reachable for all $x \in \mathsf{Sig}(\alpha_L)$. Due to the monotonicity of $\mathbf{S}$-reachability and $\mathbf{S}_1 \subseteq \mathbf{S}_2$, x is $\mathbf{S}_2$-reachable for all $x \in \mathsf{Sig}(\alpha_L)$. By definition, we have $\alpha \in \mathcal{O}^{\mathsf{reach}}_{\mathbf{S}_2}$.

For Point 2, we view y-reachability of x as existence of a hyperpath from y to x. The claim is proved by induction on the length of this hyperpath. Induction Start (length 0): $y = x$. Then, x is z-reachable. Induction Step (length $n+1$): there exists a hyperedge $\langle \{x_1, \dots, x_n\}, x \rangle$ (i.e., an axiom $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}$ with $\{x_1, \dots, x_n\} = \mathsf{Sig}(\alpha_L)$ and $x \in \mathsf{Sig}(\alpha_R)$) such that x_i is y-reachable via a hyperpath of length n or less. By induction hypothesis, x_i is z-reachable for all $x_i \in \mathsf{Sig}(\alpha_L)$. Thus, x is z-reachable by definition.

Point 2 can now be used to prove Point 3. It suffices to show that $\alpha \in \mathcal{O}^{\mathsf{reach}}_{x}$ implies $\alpha \in \mathcal{O}^{\mathsf{reach}}_{y}$, for each $\alpha = (\alpha_L \sqsubseteq \alpha_R) \in \mathcal{O}$. By definition, $\alpha \in \mathcal{O}^{\mathsf{reach}}_{x}$ implies that, for all $z \in \mathsf{Sig}(\alpha_L)$, z is x-reachable. Since x is y-reachable, Point 2 implies that z is y-reachable. This means that α is y-reachable, thus $\alpha \in \mathcal{O}^{\mathsf{reach}}_{y}$.

"Only if" direction of Point 4: Trivial if $x \in \mathbf{S}$. If $x \in \mathsf{Sig}(\mathcal{O}^{\mathsf{reach}}_{\mathbf{S}})$, then there is an $\alpha = (\alpha_L \sqsubseteq \alpha_R) \in \mathcal{O}^{\mathsf{reach}}_{\mathbf{S}}$ such that $x \in \mathsf{Sig}(\alpha)$. Since such an α is $\mathbf{S}$-reachable, all $x' \in \mathsf{Sig}(\alpha_L)$ must be $\mathbf{S}$-reachable. By definition, every $x' \in \mathsf{Sig}(\alpha_R)$ is also reachable. "If" direction: Assume that x is $\mathbf{S}$-reachable. By definition, if x is $\mathbf{S}$-reachable, then $x \in \mathbf{S}$, or there is an $\alpha = (\alpha_L \sqsubseteq \alpha_R) \in \mathcal{O}$ such that $x \in \mathsf{Sig}(\alpha_R)$ and, for all $y \in \mathsf{Sig}(\alpha_L)$, y is reachable from $\mathbf{S}$. It is trivial that $x \in \mathbf{S} \cup \mathsf{Sig}(\mathcal{O}^{\mathsf{reach}}_{\mathbf{S}})$ in the first case. In the latter case, we have that α is $\mathbf{S}$-reachable, implying by definition that $\alpha \in \mathcal{O}^{\mathsf{reach}}_{\mathbf{S}}$. Thus, $x \in \mathsf{Sig}(\alpha) \subseteq \mathbf{S} \cup \mathsf{Sig}(\mathcal{O}^{\mathsf{reach}}_{\mathbf{S}})$.

To prove Point 5, we assume that B is not A-reachable and that A is satisfiable w.r.t. $\mathcal{O}$. Partition $\mathcal{O}$ into $\mathcal{O}' \cup \mathcal{O}''$ with $\mathcal{O}' := \{\alpha \in \mathcal{O} \mid \alpha \text{ is } A\text{-reachable}\}$, and $\mathcal{O}'' := \mathcal{O} \backslash \mathcal{O}'$. Since A is satisfiable w.r.t. $\mathcal{O}$ and $\mathcal{O}' \subseteq \mathcal{O}$, A is also satisfiable w.r.t. $\mathcal{O}'$. Then, there is a model $\mathcal{I}'$ of $\mathcal{O}'$ such that $A^{\mathcal{I}'}$ is not empty. Extend $\mathcal{I}'$ to a new interpretation $\mathcal{I}$ by assigning $x^{\mathcal{I}} := \emptyset$ for all symbols $x \in \mathsf{Sig}(\mathcal{O}'') \backslash \mathsf{Sig}(\mathcal{O}')$. Obviously, $A^{\mathcal{I}}$ remains non-empty, while $B^{\mathcal{I}} = \emptyset$ since B is not in $\mathsf{Sig}(\mathcal{O}')$ (otherwise, B is A-reachable contradicting the initial assumption). It remains to show that $\mathcal{I}$ is a model of $\mathcal{O}$. Since $\mathcal{I}$ and $\mathcal{I}'$ coincide on interpretation of all the symbols in $\mathsf{Sig}(\mathcal{O}')$ and $\mathcal{I}' \models \mathcal{O}'$, $\mathcal{I} \models \mathcal{O}'$. Therefore, it suffices to demonstrate that $\mathcal{I}$ is a model of $\mathcal{O}''$. By definition, every axiom $\alpha = (\alpha_L \sqsubseteq \alpha_R)$ in $\mathcal{O}''$ is not A-reachable, i.e., there exists a symbol (role or concept name) $x \in \mathsf{Sig}(\alpha_L)$ that is not A-reachable. Hence, $x \notin \mathsf{Sig}(\mathcal{O}')$ and $x^{\mathcal{I}} = \emptyset$. By semantics of $\mathcal{EL}^+$, $\alpha_L^{\mathcal{I}} = \emptyset$, implying that $\mathcal{I} \models \alpha$ as required. ❑

Algorithm 6 Extraction of a reachability-based module.

```
Procedure extract-module($\mathcal{O}, \mathbf{S}$)
Input: $\mathcal{O}$: $\mathcal{EL}^+$ ontology; $\mathbf{S}$: signature
Output: $\mathcal{O}_\mathbf{S}$: reachability-based module for $\mathbf{S}$ in $\mathcal{O}$
 1: $\mathcal{O}_\mathbf{S} := \emptyset$
 2: queue := active-axioms($\mathbf{S}$)
 3: while not empty(queue) do
 4:    $(\alpha_L \sqsubseteq \alpha_R)$ := fetch(queue)
 5:    if $\mathsf{Sig}(\alpha_L) \subseteq \mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_\mathbf{S})$ then
 6:       $\mathcal{O}_\mathbf{S} := \mathcal{O}_\mathbf{S} \cup \{\alpha_L \sqsubseteq \alpha_R\}$
 7:       queue := queue $\cup$ (active-axioms($\mathsf{Sig}(\alpha_R)$) $\setminus \mathcal{O}_\mathbf{S}$)
 8:    end if
 9: end while
10: return $\mathcal{O}_\mathbf{S}$
```

The converse of Point 5 is not true in general, for instance, Endocarditis involves Tissue in the sense of reachability, but the corresponding subsumption does not follow from the ontology. This suggests that we could use reachability in a directed hypergraph as a heuristic for answering negative subsumption, in a similar but finer way as in the loose reachability.

Algorithm 6 outlines a method for extracting the reachability-based module given as input an $\mathcal{EL}^+$ ontology $\mathcal{O}$ and a signature $\mathbf{S}$. Similar to the technique developed for the refined classification algorithm (see Section 4.1), we view the input ontology $\mathcal{O}$ as a mapping $\mathsf{active\text{-}axioms} : \mathsf{Sig}(\mathcal{O}) \rightarrow 2^{\mathcal{O}}$ with $\mathsf{active\text{-}axioms}(x)$ comprising all and only axioms $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}$ such that x occurs in α_L. The intuition is that every axiom $\alpha \in \mathsf{active\text{-}axioms}(x)$ is 'active' for x in the sense that, for some $y \in \mathsf{Sig}(\mathcal{O})$, y potentially is x-reachable via α. For convenience, we define $\mathsf{active\text{-}axioms}(\mathbf{S}) := \bigcup_{x \in \mathbf{S}} \mathsf{active\text{-}axioms}(x)$ for a signature $\mathbf{S} \subseteq \mathsf{Sig}(\mathcal{O})$.

Proposition 39 (Algorithm 6 produces $\mathcal{O}_\mathbf{S}^{\mathsf{reach}}$). *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology, n the number of axioms in $\mathcal{O}$, and $\mathbf{S} \subseteq \mathsf{Sig}(\mathcal{O})$ a signature. Algorithm 6 terminates after $O(n^2)$ steps and returns the reachability-based module for $\mathbf{S}$ in $\mathcal{O}$.*

Proof. There are at most n axioms that can be added to $\mathcal{O}_\mathbf{S}$, and once added they are never removed. After each addition, the queue is augmented by $\mathsf{active\text{-}axioms}(\cdot)$. Since $|\mathsf{active\text{-}axioms}(\cdot)|$ is bounded by n, the algorithm's runtime is $O(n^2)$.

Let $\mathcal{O}_\mathbf{S}^i$ be the value of $\mathcal{O}_\mathbf{S}$ the algorithm has computed at the ith iteration, and $\mathcal{O}_\mathbf{S}^\infty$ be the output after no more active axioms are to be processed. We prove by induction on i that all axioms in $\mathcal{O}_\mathbf{S}^i$ are $\mathbf{S}$-reachable. For the induction start, it is trivial since $\mathcal{O}_\mathbf{S}^0$ is empty. For the induction step, assume that $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}_\mathbf{S}^{i+1} \setminus \mathcal{O}_\mathbf{S}^i$ is the new axiom added at iteration $i+1$. This is possible only when the condition given in line 5 is satisfied, i.e., $\mathsf{Sig}(\alpha_L) \subseteq \mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_\mathbf{S}^i)$. By induction hypothesis, axioms in $\mathcal{O}_\mathbf{S}^i$ are $\mathbf{S}$-reachable, and thus all the symbols in $\mathsf{Sig}(\alpha_L) \subseteq \mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_\mathbf{S}^i)$ are $\mathbf{S}$-reachable. By definition, $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}_\mathbf{S}^{i+1}$ is $\mathbf{S}$-reachable as required.

It remains to show that the algorithm extracts all $\mathbf{S}$-reachable axioms. An axiom

$\alpha = (\alpha_L \sqsubseteq \alpha_R)$ is $\mathbf{S}$-reachable if all symbols in $\mathsf{Sig}(\alpha_L)$ are $\mathbf{S}$-reachable. All potential $\mathbf{S}$-reachable symbols according to Definition 37 on page 82 are considered through the way the algorithm initializes and maintains queue. In fact, it starts with all active axioms for $\mathbf{S}$ which corresponds to the base case in Definition 37. Then, it recursively extends queue with all active axioms for $\mathsf{Sig}(\alpha_R)$ modulo $\mathcal{O}_\mathbf{S}$ once $\alpha_L \sqsubseteq \alpha_R$ is known to be $\mathbf{S}$-reachable which corresponds to the the induction case in Definition 37.

❑

In fact, reachability can be reduced to propositional Horn clause implication. The idea is to translate the $\mathcal{EL}^+$ ontology $\mathcal{O}$ in module normal form into a propositional Horn formula $\phi[\mathcal{O}]$ comprising the clauses:

$$l_1 \wedge \cdots \wedge l_m \rightarrow r_1 \wedge \cdots \wedge r_n$$

if $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}$ such that $l_i \in \mathsf{Sig}(\alpha_L)$ and $r_j \in \mathsf{Sig}(\alpha_R)$, for $1 \leq i \leq m$ and $1 \leq j \leq n$. Given a signature $\mathbf{S}$ and a symbol x, x is $\mathbf{S}$-reachable in $\mathcal{O}$ if, and only if, x is implied by $\bigwedge_{y \in \mathbf{S}} y$ w.r.t. $\phi[\mathcal{O}]$. Hence, the algorithm proposed by Dowling and Gallier [DG84] can be used to check $\mathbf{S}$-reachability in linear time. Also, the counting technique used in [DG84] can be exploited to extract the reachability-based module in linear time. More precisely, a counter $\mathsf{counter}(\alpha)$ is maintained for each axiom $\alpha = (\alpha_L \sqsubseteq \alpha_R)$ in $\mathcal{O}$ and initialized with $|\mathsf{Sig}(\alpha_L)|$. Whenever a new symbol x becomes $\mathbf{S}$-reachable, the counters of all the active axioms for x are decremented by one. As soon as the counter $\mathsf{counter}(\alpha)$ reaches zero for some active axiom $\alpha = (\alpha_L \sqsubseteq \alpha_R) \in \mathsf{active\text{-}axiom}(x)$ (meaning that all symbols in $\mathsf{Sig}(\alpha_L)$ are $\mathbf{S}$-reachable), every symbol $y \in \mathsf{Sig}(\alpha_R)$ is marked $\mathbf{S}$-reachable. In addition, we mark the axiom with 'reached' to denote that it is $\mathbf{S}$-reachable, thus in the reachability-based module. The procedure carries on until a fixpoint is reached, i.e., no more symbol is becoming $\mathbf{S}$-reachable. This method improves on Algorithm 6 in the sense that it treats an axiom only when the counter reaches zero, i.e., the axiom is already known to be reachable. This avoids potential redundant tests at line 5. The complexity of the Horn-motivated method is linear in the size of the ontology.[2]

This shows that the presented algorithm is suboptimal since we can in principle do better. In practice, however, $|\mathsf{active\text{-}axioms}(\cdot)|$ is relatively small compared to the size of the ontology and thus can effectively be considered as a constant. Moreover, the left-hand side α_L of an axiom is usually so small that the counting technique does not really pay off. The experiment results described in Subsection 6.2.4 demonstrate that Algorithm 6 performs quite well on realistic biomedical ontologies.

5.1.2 Modularity and Subsumption

In this subsection, we show certain connections between the reachability-based modules and subsumption.

[2]Note that, though Algorithm 6 runs in quadratic time, it is measured w.r.t. the number of axioms in the ontology, as opposed to the size of the ontology.

Lemma 40 **($\mathcal{O}^{\mathsf{reach}}_{\mathsf{Sig}(C)}$ preserves $C \sqsubseteq_{\mathcal{O}} D$).** *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology and $\mathbf{S} \subseteq \mathsf{Sig}(\mathcal{O})$ a signature. Then, $C \sqsubseteq_{\mathcal{O}} D$ if, and only if, $C \sqsubseteq_{\mathcal{O}^{\mathsf{reach}}_{\mathbf{S}}} D$ for arbitrary $\mathcal{EL}^+$ concept descriptions C, D such that $\mathsf{Sig}(C) \subseteq \mathbf{S}$.*

Proof. "If" direction immediately follows from monotonicity of $\mathcal{EL}^+$ and the fact that $\mathcal{O}^{\mathsf{reach}}_{\mathbf{S}} \subseteq \mathcal{O}$, so only the "only if" direction remains to be shown. Assume to the contrary that $C \sqsubseteq_{\mathcal{O}} D$ but $C \not\sqsubseteq_{\mathcal{O}^{\mathsf{reach}}_{\mathbf{S}}} D$. Then, there must exist an interpretation $\mathcal{I}$ and an individual $w \in \Delta^{\mathcal{I}}$ such that $\mathcal{I} \models \mathcal{O}^{\mathsf{reach}}_{\mathbf{S}}$ and $w \in C^{\mathcal{I}} \backslash D^{\mathcal{I}}$. Modify $\mathcal{I}$ to $\mathcal{I}'$ by setting $x^{\mathcal{I}'} := \emptyset$ for all $x \in \mathsf{Sig}(\mathcal{O}) \backslash (\mathbf{S} \cup \mathsf{Sig}(\mathcal{O}^{\mathsf{reach}}_{\mathbf{S}}))$. Obviously, $\mathcal{I}'$ is a model of $\mathcal{O}^{\mathsf{reach}}_{\mathbf{S}}$ since it does not change the interpretation of any symbol in it. For each $\alpha = (\alpha_L \sqsubseteq \alpha_R) \in \mathcal{O} \backslash \mathcal{O}^{\mathsf{reach}}_{\mathbf{S}}$, we have $\alpha_L^{\mathcal{I}'} \subseteq \alpha_R^{\mathcal{I}'}$ since α is not $\mathbf{S}$-reachable and thus α_L is composed of an x with $x^{\mathcal{I}'} = \emptyset$. Therefore, $\mathcal{I}'$ is a model of $\mathcal{O}$. It remains to be shown that $w \in C^{\mathcal{I}'} \backslash D^{\mathcal{I}'}$. But, this is the case since both $\mathcal{I}$ and $\mathcal{I}'$ coincide on all the symbols $y \in \mathsf{Sig}(C) \subseteq \mathbf{S}$, and $D^{\mathcal{I}'} \subseteq D^{\mathcal{I}}$.

❏

It immediately follows that $\mathcal{O}^{\mathsf{reach}}_{\mathbf{S}}$ is an $\mathbf{S}$-module, i.e., it preserves all subsumption relationships between concept descriptions constructed out of $\mathbf{S}$. Moreover, the following corollary is obtained when we restrict attention to subsumption between concept names.

Corollary 41. *$\mathcal{O}^{\mathsf{reach}}_{A}$ is a subsumption module for A in $\mathcal{O}$.*

The property of subsumption module suggests that, to query subsumption, it is enough to extract and maintain only linearly many reachability-based modules, i.e., one for each concept name. Precisely, the module $\mathcal{O}^{\mathsf{reach}}_{A}$ can be used to correctly answer subsumption $A \sqsubseteq^?_{\mathcal{O}} B$ for every concept name B occurring in $\mathcal{O}$. Consider our example ontology from Chapter 2. It is not difficult to derive that Inflammation, Disease, and HeartDisease are the only subsumers of the concept Endocarditis and to see that all these subsumptions remain in the module $\mathcal{O}^{\mathsf{reach}}_{\mathsf{Endocarditis}}$.

In principle, it should be highly effective to use a small subsumption module to optimize standard reasoning of subsumption due to a smaller number of axioms needed to be taken into consideration. This is nevertheless not helpful if the goal-directed subsumption algorithm (see Section 4.2) is used for subsumption querying. As already mentioned, there is a connection between this algorithm and reachability-based modularity. In fact, any axiom that is involved in a rule application during the computation of subsumes?$(A \sqsubseteq B)$ (see Algorithm 2 on page 64) belongs to the reachability-based module $\mathcal{O}^{\mathsf{reach}}_{A}$ in $\mathcal{O}$. The following theorem states this correlation:

Theorem 42 **(subsumes?$(A_0 \sqsubseteq B_0)$ only requires axioms in $\mathcal{O}^{\mathsf{reach}}_{A_0}$).** *Let $\mathcal{O}$ be an ontology in $\mathcal{EL}^+$ normal form, and $\mathcal{O}^{\mathsf{reach}}_{A_0}$ the reachability-based module for $\mathbf{S} = \{A_0\}$ in $\mathcal{O}$. Then,* subsumes?$(A_0 \sqsubseteq B_0)$ *(i.e., the subsumption-testing procedure in Algorithm 2 on page 64) only requires axioms in $\mathcal{O}^{\mathsf{reach}}_{A_0}$.*

Proof. Assume that Algorithm 2 requires α, for some axiom $\alpha \in \mathcal{O}$, i.e., α is used in a rule application and thus causes addition to either $S(\cdot)$ or $R(\cdot)$. Before we can prove the proposition, we need the following invariants:

INV1: If a concept name A is activated, then A is A_0-reachable w.r.t. $\mathcal{O}$.

INV2: If $B \in S(A)$ for some concept names A, B, then B is A_0-reachable w.r.t. $\mathcal{O}$.

INV3: If $(A, B) \in R(r)$ for some role name r, then r is A_0-reachable w.r.t. $\mathcal{O}$.

INV4: If $(\mathbf{B} \to B) \in \mathsf{queue}(A)$ and, for all $B' \in \mathbf{B}$, B' is A_0-reachable, then B is A_0-reachable (a special case, if $B \in \mathsf{queue}(A)$, then B is A_0-reachable); and, if $\exists r.B \in \mathsf{queue}(A)$, then r and B are A_0-reachable.

INV5: If r is processed by process-new-edge, then r is A_0-reachable w.r.t. $\mathcal{O}$.

We start with demonstrating that all the invariants hold after initialization. The goal-directed subsumption algorithm initializes itself by activating A_0 and performing the following operations:

- $S(A_0) := \{A_0, \top\}$;
- $R(r) := \{(A_0, A_0)\}$ if $\mathcal{O} \models \epsilon \sqsubseteq r$, or $R(r) := \emptyset$ otherwise; and
- $\mathsf{queue}(A_0) := \widehat{\mathcal{O}}(A_0) \cup \widehat{\mathcal{O}}(\top) \cup \bigcup_{r \text{ with } \mathcal{O} \models \epsilon \sqsubseteq r} \Big(\widehat{\mathcal{O}}(\exists r.A_0) \ \cup \ \widehat{\mathcal{O}}(\exists r.\top)\Big)$.

It is obvious to see that **INV5** is not affected by initialization. **INV1** and **INV2** hold since A_0 is A_0-reachable w.r.t. $\mathcal{O}$ by the definition of reachable names (see Definition 37 on page 82). We make a case distinction for **INV3**. In the first case, let r be such that $\mathcal{O} \models \epsilon \sqsubseteq r$, i.e., there exists a chain of role hierarchy axioms $\epsilon \sqsubseteq r_1, r_1 \sqsubseteq r_2, \ldots, r_{n-1} \sqsubseteq r_n \in \mathcal{O}$ with $r = r_n$. Since $\mathsf{Sig}(\epsilon) = \emptyset$, all role names in the chain are x-reachable for any $x \in \mathsf{Sig}(\mathcal{O})$. In particular, r is A_0-reachable, thus preserving **INV3**. In the second case, **INV3** holds vacuously since $R(r)$ is initialized with the empty set. An entry that is added to the queue of A_0 corresponds to one of the following axioms (up to commutativity of $\sqcap$) for some concept names $B_{(i)}$ and role name s: $A_0 \sqcap B_1 \sqcap \cdots \sqcap B_n \sqsubseteq B$, $A_0 \sqsubseteq \exists s.B$, $\top \sqsubseteq B$, $\top \sqsubseteq \exists s.B$, $\exists r.A_0 \sqsubseteq B$ and $\exists r.\top \sqsubseteq B$. The first axiom induces the conditional queue entry $(\mathbf{B} \to B) \in \widehat{\mathcal{O}}(A_0)$ with $\mathbf{B} = \{B_1, \ldots, B_n\}$. By the definition of reachable names, if A_0 and B_i for $1 \leq i \leq n$ are A_0-reachable, then B is as well A_0-reachable. Hence, **INV4** is preserved w.r.t. the conditional queue entry since A_0 is A_0-reachable. In the special case where $\mathbf{B} = \emptyset$ and all other cases, adding any of these entries to $\mathsf{queue}(A_0)$ does not violate **INV4** since all the symbols on the lhs of these axioms (i.e., r, A_0) and thus those on the rhs (i.e., s, B) are A_0-reachable.

After initialization, the data structures are manipulated and concept names become activated. It remains to show that every modification preserves the invariants. We make a case distinction based on the lines of execution that can potentially violate one of the invariants above. Recall that only activated concepts A can be processed by goal-directed-process.

- In line 8 of goal-directed-process, B is added to $S(A)$ if $\mathbf{B} \subseteq S(A)$. We need to show that **INV2** (the only invariant that could be invalidated) is preserved. By **INV2**, $\mathbf{B} \subseteq S(A)$ implies that B_i is A_0-reachable for every $B_i \in \mathbf{B}$. Since $\mathbf{B} \rightarrow B$ was in queue(A), the first part of **INV4** implies that B is A_0-reachable, thus preserving **INV2** as required.

- In line 9 of goal-directed-process, elements from $\widehat{\mathcal{O}}(B)$ are added to queue(A). We need to show that **INV4** is preserved. Due to **INV2** and the fact that $B \in S(A)$, B is A_0-reachable. There are three potential kinds of axioms involved, i.e., $B \sqsubseteq B'$, $B \sqsubseteq \exists r.B'$, and $B_1 \sqcap \cdots \sqcap B_n \sqsubseteq B'$ such that $B = B_i$ and $1 \leq i \leq n$. In the first two cases, B' $(r, B'$, resp.) is A_0-reachable by the definition of reachable names, thus preserving **INV4**. In the last case, the first part of **INV4** follows immediately from the definition of reachable names where $\mathsf{Sig}(\alpha_L) = \{B_1, \ldots, B_n\}$ and $\mathsf{Sig}(\alpha_R) = \{B'\}$.

- In line 11 of goal-directed-process, elements B'' of $\widehat{\mathcal{O}}(\exists r.B)$ are added to queue(A') such that $(A', A) \in R(r)$ and $B \in S(A)$. We need to show that **INV4** is preserved. Thus, there is an axiom $\exists r.B \sqsubseteq B''$ in $\mathcal{O}$. It holds that B is A_0-reachable due to **INV2** and $B \in S(A)$, while r is A_0-reachable due to **INV3** and $(A', A) \in R(r)$. By the definition of reachable names, B'' is also A_0-reachable, preserving (the special case of) **INV4**.

- In line 16 of goal-directed-process, B is activated. We need to show that **INV1** is preserved. Since $\exists r.B$ occurred in queue(A), the second part of **INV4** guarantees that B (together with r) is A_0-reachable, thus preserving **INV1**.

- In line 17 of goal-directed-process, the sub-procedure process-new-edge is invoked. We need to show that **INV5** is preserved. Since $\exists r.B$ occurred in queue(A), the second part of **INV4** guarantees that r (together with B) is A_0-reachable, thus preserving **INV5**.

- In line 2 of process-new-edge, the pair (A, B) is added to $R(s)$ with $\mathcal{O} \models r \sqsubseteq s$. We need to show that **INV3** is preserved. It is already shown above that r is A_0-reachable. It follows by the definition of reachable names that s is also A_0-reachable. Therefore, **INV3** is preserved.

- In line 3 of process-new-edge, elements B'' of $\widehat{\mathcal{O}}(\exists s.B')$ with $B' \in S(B)$ are added to queue(A). We need to show that **INV4** is preserved. Thus, there exists an axiom $\exists s.B' \sqsubseteq B''$ in $\mathcal{O}$. It is readily proved above that s is A_0-reachable. Due to **INV2** and that $B' \in S(B)$, B' is also A_0-reachable. Hence, augmenting the queue with B'' preserves (the special case of) **INV4**.

- In line 5 of process-new-edge, we need to show that u is A_0-reachable for **INV5** to be preserved. There is an axiom $t \circ s \sqsubseteq u$ in $\mathcal{O}$ such that $(A', A) \in R(t)$ and $(A, B) \in R(s)$. By **INV3**, both s and t are A_0-reachable, and so is u as required.

- In line 8 of process-new-edge, preservation of **INV5** can be demonstrated in a parallel way.

Now we show that α is indeed A_0-reachable w.r.t. $\mathcal{O}$, thus in $\mathcal{O}_{A_0}^{\mathsf{reach}}$. We make a case distinction w.r.t. the form of α.

- $X \sqsubseteq Y$ is required when $Y \in \widehat{\mathcal{O}}(X)$ has been enqueued to $\mathsf{queue}(A)$ for some concept name A. This means that X had been added to $S(A)$, implying by **INV2** that X is A_0-reachable by **INV2**. Obviously, α is A_0-reachable by the definition of reachable axioms.

- $X_1 \sqcap \cdots \sqcap X_n \sqsubseteq Y$ is required if X_i has been added to $S(A)$ and $\{X_1, \ldots, X_n\} \subseteq S(A)$ for some concept name A and $1 \leq i \leq n$. By **INV2**, every X_i is A_0-reachable, and thus α is A_0-reachable by definition.

- $X \sqsubseteq \exists r.Y$ (analogous to the first case).

- $\exists r.X \sqsubseteq Y$ is required when $Y \in \widehat{\mathcal{O}}(\exists r.X)$ has been enqueued to $\mathsf{queue}(A)$ for some concept name A (line 11 in goal-directed-process and line 3 in process-new-edge). Since $R(r)$ is not empty, **INV3** implies that r is A_0-reachable. Also, X is A_0-reachable since X occurs in some $S(B)$. By definition, α is A_0-reachable.

- $r \sqsubseteq s$ is required when it participates in the outer **for**-loop in process-new-edge. Since r is A_0-reachable by **INV5**, s is also A_0-reachable.

- $u \circ s \sqsubseteq v$ is required when the conditions in line 4 (resp, line 6) of process-new-edge are satisfied. Obviously, α is A_0-reachable since both u and s are.

❑

We now establish the following result which plays a principal role in modularization-based axiom pinpointing (see Subsection 5.2.4).

Theorem 43 ($\mathcal{O}_A^{\mathsf{reach}}$ is strong subsumption module). *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology and A a concept name. Then $\mathcal{O}_A^{\mathsf{reach}}$ is a strong subsumption module for A in $\mathcal{O}$.*

Proof. The fact that $\mathcal{O}_A^{\mathsf{reach}}$ is a subsumption module is shown in Corollary 41 on page 86. To show that it is strong, assume that $A \sqsubseteq_{\mathcal{O}} B$ holds, but there is a MinA $\mathcal{S}$ for $A \sqsubseteq_{\mathcal{O}} B$ that is not contained in $\mathcal{O}_A^{\mathsf{reach}}$. Thus, there must be an axiom $\alpha \in \mathcal{S} \setminus \mathcal{O}_A^{\mathsf{reach}}$. Define $\mathcal{S}_1 := \mathcal{S} \cap \mathcal{O}_A^{\mathsf{reach}}$. Note that $\mathcal{S}_1$ is a strict subset of $\mathcal{S}$ since $\alpha \notin \mathcal{S}_1$. We claim that $A \sqsubseteq_{\mathcal{S}} B$ implies $A \sqsubseteq_{\mathcal{S}_1} B$ which contradicts the fact that $\mathcal{S} \nsubseteq \mathcal{O}_A^{\mathsf{reach}}$ is a MinA for $A \sqsubseteq_{\mathcal{O}} B$.

We prove the claim by showing the contraposition: assume that $A \not\sqsubseteq_{\mathcal{S}_1} B$, i.e., there is a model $\mathcal{I}_1$ of $\mathcal{S}_1$ such that $A^{\mathcal{I}_1} \nsubseteq B^{\mathcal{I}_1}$. We extend $\mathcal{I}_1$ to $\mathcal{I}$ by setting $y^{\mathcal{I}} := \emptyset$ for all symbols (concept and role names) y that are not A-reachable. It is easy to see that $A^{\mathcal{I}} \nsubseteq B^{\mathcal{I}}$. In fact, we have $A^{\mathcal{I}} = A^{\mathcal{I}_1}$ (since A is A-reachable); and $B^{\mathcal{I}} = B^{\mathcal{I}_1}$ if B is A-reachable, or $B^{\mathcal{I}} = \emptyset$ otherwise. It remains to be shown that $\mathcal{I}$ is indeed a model of $\mathcal{S}$, i.e. satisfies all axioms $\beta = (\beta_L \sqsubseteq \beta_R)$ in $\mathcal{S}$. There are two possibilities:

- $\beta \in \mathcal{S}_1$. Since $\mathcal{S}_1 \subseteq \mathcal{O}_A^{\mathsf{reach}}$, all symbols in $\mathsf{Sig}(\beta)$ are A-reachable. Consequently, $\mathcal{I}_1$ and $\mathcal{I}$ coincide on the names occurring in $\beta_L \sqsubseteq \beta_R$. Since $\mathcal{I}_1$ is a model of $\mathcal{S}_1$, we thus have $(\beta_L)^{\mathcal{I}} = (\beta_L)^{\mathcal{I}_1} \subseteq (\beta_R)^{\mathcal{I}_1} = (\beta_R)^{\mathcal{I}}$.

- $\beta \in \mathcal{S} \backslash \mathcal{S}_1$. Since $\mathcal{S}_1 = \mathcal{S} \cap \mathcal{O}_A^{\mathsf{reach}}$, we have that β is not A-reachable. Thus, β_L contains a name x that is not A-reachable. By the definition of $\mathcal{I}$, $x^{\mathcal{I}} = \emptyset$, implying that $(\beta_L)^{\mathcal{I}} = \emptyset$. Hence, the axiom is trivially satisfied.

So, $\mathcal{I}$ is a model of $\mathcal{S}$ as required. ❏

5.1.3 Related extraction methods

Recently, various techniques for extracting fragments of ontologies have been proposed in the literature. An example is the algorithm proposed in [SR06] which was developed specifically for the Galen medical knowledge base. The algorithm traverses in definitional order and into existential quantifications but does not take into account other dependencies such as role hierarchy and GCIs. This method is definition-closed and as such corresponds to unfolding a concept. If applied to our example ontology $\mathcal{O}_{\mathsf{med}}$ (Figure 2.2 on page 24), the algorithm would extract only α_7 and α_8 as the segmentation output for the signature $\mathbf{S} = \{\mathsf{Pancarditis}\}$, whereas $\mathcal{O}_{\mathsf{Pancarditis}}^{\mathsf{reach}}$ comprises α_7–α_{10} and α_{12}–α_{15}. Obviously, the segmentation output is not a subsumption module for Pancarditis in the sense of Definition 23 on page 31 since it does not preserve, e.g., the subsumption $\mathsf{Pancarditis} \sqsubseteq_{\mathcal{O}_{\mathsf{med}}} \mathsf{HeartDisease}$.

Another example is the Prompt-Factor tool [NM03] which implements an algorithm that, given an ontology $\mathcal{O}$ and a signature $\mathbf{S}$, computes a subset $\mathcal{O}_1 \subseteq \mathcal{O}$ by retrieving to $\mathcal{O}_1$ axioms that contain symbols in $\mathbf{S}$ and extending $\mathbf{S}$ with $\mathsf{Sig}(\mathcal{O}_1)$ until a fixpoint is reached. This is similar to our modules based on the *extended loose reachability*, but it does not distinguish symbols occurring on lhs and rhs of axioms. Hence, it corresponds to the reachability problem in an *undirected* graph. Considering our example ontology, the tool would return the whole ontology as output for $\mathbf{S} = \{\mathsf{Pancarditis}\}$, even though several axioms are irrelevant.

The following proposition formally states the relationships among the results of these two methods and the reachability-based module:

Proposition 44 ($\mathcal{O}_{\mathbf{S}}^{\mathsf{def}} \subseteq \mathcal{O}_{\mathbf{S}}^{\mathsf{reach}} \subseteq \mathcal{O}_{\mathbf{S}}^{\mathsf{loose}} \subseteq \mathcal{O}_{\mathbf{S}}^{\mathsf{prompt}}$)**.** *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology and $\mathbf{S}$ a signature. Then, the following hold:*

1. $\mathcal{O}_{\mathbf{S}}^{\mathsf{def}} \subseteq \mathcal{O}_{\mathbf{S}}^{\mathsf{reach}}$

2. $\mathcal{O}_{\mathbf{S}}^{\mathsf{reach}} \subseteq \mathcal{O}_{\mathbf{S}}^{\mathsf{loose}}$

3. $\mathcal{O}_{\mathbf{S}}^{\mathsf{loose}} \subseteq \mathcal{O}_{\mathbf{S}}^{\mathsf{prompt}}$

where $\mathcal{O}_{\mathbf{S}}^{\mathsf{def}}$ is the result of applying the algorithm from [SR06] to $\mathcal{O}$ and $\mathbf{S}$, $\mathcal{O}_{\mathbf{S}}^{\mathsf{reach}}$ is the reachability-based module for $\mathbf{S}$ in $\mathcal{O}$, $\mathcal{O}_{\mathbf{S}}^{\mathsf{loose}}$ is the module based on the extended loose reachability for $\mathbf{S}$ in $\mathcal{O}$, and $\mathcal{O}_{\mathbf{S}}^{\mathsf{prompt}}$ is the result from the Prompt-Factor [NM03] tool applied to $\mathcal{O}$ and $\mathbf{S}$.

Proof. Point 1 is trivial since $\mathcal{O}_{\mathbf{S}}^{\mathsf{reach}}$ is definition-closed, i.e., contains all concept definitions that are used for unfolding a concept $A \in \mathbf{S}$.

Since reachability in (directed) hypergraph is stronger than that in (directed) graph, Point 2 follows from the fact that if a hyperedge $\langle \{x_1, \ldots, x_n\}, x \rangle$ (in the corresponding hypergraph for $\mathbf{S}$-reachability) is induced by an axiom $\alpha = (\alpha_L \sqsubseteq \alpha_R)$ with $\{x_1, \ldots, x_n\} = \mathsf{Sig}(\alpha_L)$ and $x \in \mathsf{Sig}(\alpha_R)$, then there must also be an edge $\langle x_i, x \rangle$ in $\mathcal{G}'_l(\mathcal{O})$ for every x_i.

For Point 3, $\mathcal{O}_{\mathbf{S}}^{\mathsf{prompt}}$ is based on the extended loose reachability graph, in which edges are undirected (bidirectional). Let $\mathcal{G}'_l(\mathcal{O})$ be the extended loose reachability graph for $\mathcal{O}$, and $\mathcal{G}''_l(\mathcal{O})$ denote the undirected graph obtained from $\mathcal{G}'_l(\mathcal{O})$. Obviously, if there is a path from an $x \in \mathbf{S}$ in the directed graph $\mathcal{G}'_l(\mathcal{O})$, then so is there in the undirected graph $\mathcal{G}''_l(\mathcal{O})$. Hence, the module based on reachability in $\mathcal{G}'_l(\mathcal{O})$ (i.e., $\mathcal{O}_{\mathbf{S}}^{\mathsf{loose}}$) is contained in the one based on reachability in $\mathcal{G}''_l(\mathcal{O})$ (i.e., $\mathcal{O}_{\mathbf{S}}^{\mathsf{prompt}}$). ❑

By definition, a locality-based module (see Definition 22 on page 31 and [CHKS07, CHWK07]) for a given signature is not necessarily unique, but the algorithm given in [CHKS07] extract the *minimal* locality-based module for a given signature. It was shown in [CHKS08] that given an ontology $\mathcal{O}$ and a signature $\mathbf{S}$, there always exists the unique minimal locality-based module $\mathcal{O}_{\mathbf{S}}^{\mathsf{loc}}$. The authors also showed that modules based on syntactic locality are subsumption module in the sense defined in Definition 23 on page 31. In the following, we show that the result from their algorithm coincide with that from Algorithm 6. In other words, the reachability-based module is equivalent to the minimal locality-based module.

Theorem 45 ($\mathcal{O}_{\mathbf{S}}^{\mathsf{reach}}$ is the minimal locality-based module). *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology, $\mathbf{S} \subseteq \mathsf{Sig}(\mathcal{O})$ a signature. Then, $\mathcal{O}_{\mathbf{S}}^{\mathsf{reach}}$ is the minimal locality-based module for $\mathbf{S}$ in $\mathcal{O}$.*

Proof. First, we show that $\mathcal{O}_{\mathbf{S}}^{\mathsf{reach}}$ is a locality-based module. To prove this, it suffices to show that, for each axiom $\alpha = (\alpha_L \sqsubseteq \alpha_R) \in \mathcal{O} \backslash \mathcal{O}_{\mathbf{S}}^{\mathsf{reach}}$, α is syntactically local w.r.t. $\mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_{\mathbf{S}}^{\mathsf{reach}})$. Since $\mathcal{O}_{\mathbf{S}}^{\mathsf{reach}}$ comprises all $\mathbf{S}$-reachable axioms, α is not $\mathbf{S}$-reachable, i.e., there exists an $x \in \mathsf{Sig}(\alpha_L)$ such that x is not $\mathbf{S}$-reachable. By Point 4 of Proposition 38 on page 82, $x \notin \mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_{\mathbf{S}}^{\mathsf{reach}})$. Since x occurs in α_L, by Definition 22 on page 31, α is syntactically local w.r.t. $\mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_{\mathbf{S}}^{\mathsf{reach}})$, as required.

It remains to be shown that $\mathcal{O}_{\mathbf{S}}^{\mathsf{reach}}$ is minimal. Assume to the contrary that a proper subset $\mathcal{O}_{\mathbf{S}}^{\mathsf{reach}} \backslash \{\alpha\}$ is a locality-based module, for some axiom $\alpha = (\alpha_L \sqsubseteq \alpha_R) \in \mathcal{O}_{\mathbf{S}}^{\mathsf{reach}}$. By definition, each axiom $\beta \in \mathcal{O} \backslash (\mathcal{O}_{\mathbf{S}}^{\mathsf{reach}} \backslash \{\alpha\})$ is syntactically local w.r.t. $\mathbf{S}' = \mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_{\mathbf{S}}^{\mathsf{reach}} \backslash \{\alpha\})$. In particular, α is syntactically local w.r.t. $\mathbf{S}'$. Our claim is that α is *not* $\mathbf{S}$-reachable w.r.t. $\mathcal{O}$, contradicting the fact that $\alpha \in \mathcal{O}_{\mathbf{S}}^{\mathsf{reach}}$. $\lightning$

Claim: Let $\mathbf{S}' = \mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_{\mathbf{S}}^{\mathsf{reach}} \backslash \{\alpha\})$ with α syntactically local w.r.t. $\mathbf{S}'$. Then, α is not $\mathbf{S}$-reachable w.r.t. $\mathcal{O}$.

Since $\alpha = (\alpha_L \sqsubseteq \alpha_R)$ is syntactically local w.r.t. $\mathbf{S}'$, there must exist an $x \in \mathsf{Sig}(\alpha_L)$ such that $x \notin \mathbf{S}'$. There are two mutually disjoint possibilities:

- $x \notin \mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_{\mathbf{S}}^{\mathsf{reach}})$. Then, x (thus, α) is not $\mathbf{S}$-reachable by Point 4 of Proposition 38.

- $x \in \mathsf{Sig}(\alpha) \backslash (\mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_{\mathbf{S}}^{\mathsf{reach}} \backslash \{\alpha\}))$. Then, x does not occur in $\mathbf{S}$ nor in any other axioms in $\mathcal{O}_{\mathbf{S}}^{\mathsf{reach}}$ apart from α. It follows that x cannot be $\mathbf{S}$-reachable since it only occurs in α, in which it occurs on the left-hand side. Thus, α is not $\mathbf{S}$-reachable w.r.t. $\mathcal{O}$.

❏

Below we formulate an immediate consequence of Theorem 43 and 45:

Corollary 46. *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology and A a concept name. Then, a locality-based module for $\mathbf{S} = \{A\}$ in $\mathcal{O}$ is a strong subsumption module for A in $\mathcal{O}$.*

Proof. Let $\mathcal{O}' \subseteq \mathcal{O}$ be a locality-based module for $\mathbf{S} = \{A\}$ in $\mathcal{O}$. Theorem 45 implies that $\mathcal{O}_A^{\mathsf{reach}} \subseteq \mathcal{O}'$. By Theorem 43, $\mathcal{O}_A^{\mathsf{reach}}$ contains all the MinAs for $A \sqsubseteq_{\mathcal{O}} B$ for all concept names B in $\mathcal{O}$. Hence, all those MinAs must also be contained $\mathcal{O}'$, implying that $\mathcal{O}'$ is a strong subsumption module. ❏

The property of strong subsumption module also extends to locality-based modules in the expressive DL $\mathcal{SHOIQ}$. As this is out of scope of the present dissertation, we refer the interested reader to [SQJH08].

Another consequence of Theorem 45 is that Algorithm 6 can be used to extract a locality-based module in an $\mathcal{EL}^+$ ontology. The main difference, in contrast to the algorithm used in [CHKS07, CHWK07], is that Algorithm 6 considers only "active" axioms for $\mathsf{Sig}(\alpha_R)$ when a new axiom $\alpha_L \sqsubseteq \alpha_R$ is extracted, as opposed to all the axioms in the ontology. This could help to greatly reduce the search space especially when the number of active axioms is relatively small. Also, testing whether an $\mathcal{EL}^+$ axiom $\alpha = (\alpha_L \sqsubseteq \alpha_R)$ is non-local w.r.t. a signature $\mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_{\mathbf{S}})$ boils down to testing $\mathbf{S}$-reachability of α, which is a simpler operation of testing set inclusion $\mathsf{Sig}(\alpha_L) \subseteq^? \mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_{\mathbf{S}})$. This is due to the fact that any $\mathcal{EL}^+$ concept description and role composition α_L, with an $x \in \mathsf{Sig}(\alpha_L)$ interpreted as the empty set, are themselves interpreted as the empty set. This observation could be used to optimize module extraction of ontologies formulated in expressive Description Logics.

5.2 Axiom Pinpointing

Similar to writing large software, building large-scale ontologies is an error-prone endeavor. Techniques for standard reasoning developed in Chapter 4, such as subsumption, can help alert the developer or user of an ontology to the existence of errors. For example, in the current version of the medical ontology Snomed ct, the concept 'amputation of finger' is classified as a subconcept of 'amputation of hand,' which is clearly unintended [SBSS07, SMH07]. Formally, the following subsumption holds:

$$\mathsf{AmputationOfFinger} \sqsubseteq_{\mathcal{O}^{\textsc{Snomed}}} \mathsf{AmputationOfHand}. \tag{5.1}$$

Finding the axioms that are responsible for this among almost four thousand axioms of Snomed ct is at best not always easy by hand.

To overcome this, the problem of computing all justifications (henceforward, MinAs) has emerged as a key reasoning service for ontology design and maintenance

[SC03, PSK05, MLPB06, BPS07b, BP07, KPHS07, BS08]. In [SC03], Schlobach and Cornet have coined the name “axiom pinpointing” for the task of finding all MinAs, given an entailment. Most of the work on axiom pinpointing in DLs was concerned with rather expressive DLs (see, e.g., [SC03, PSK05, MLPB06]).

As a first step towards providing such support, Schlobach and Cornet [SC03] describe an algorithm for computing all the MinAs for a given entailment that follows from an ontology (see Definition 17 on page 29). A MinA helps the user to comprehend why a certain consequence holds. The underlying DL considered in [SC03] is $\mathcal{ALC}$ with an unfoldable TBox, and the unwanted entailments are unsatisfiability of concepts. The algorithm is an extension of the known tableau-based satisfiability algorithm for $\mathcal{ALC}$ [SSS91, BN07], where labels keep track of which axioms are responsible for an assertion to be generated during the run of the algorithm.

The problem of computing MinAs of a DL ontology was actually considered earlier in the context of extending DLs by default rules. In [BH95], Baader and Hollunder tackled this problem by introducing a labeled extension of the tableau-based consistency algorithm for $\mathcal{ALC}$ ABoxes [Hol90], which is very similar to the one described later in [SC03]. The main difference is that the algorithm described in [BH95] does not directly compute minimal subsets having an entailment, but rather a monotone Boolean formula whose variables correspond to the axioms of the knowledge bases and whose minimal satisfying valuations correspond to the MinAs. Another difference is that in [BH95] the ABox is divided into a static and a refutable part, where the elements of the static part are assumed to be always present, and subsets are built only of the refutable part of the ABox.

The approach by Schlobach and Cornet was extended by Parsia et al. [PSK05] to more expressive DLs, and the one by Baader and Hollunder was extended by Meyer et al. [MLPB06] to the case of $\mathcal{ALC}$ with general TBoxes, which are no longer unfoldable.

While the previous work on pinpointing in DLs considered fairly expressive DLs that contain at least $\mathcal{ALC}$, the results reported in this section are concerned with pinpointing in the lightweight DL $\mathcal{EL}^+$. Also, we focus on subsumption as opposed to unsatisfiability because this is more interesting in sub-$\mathcal{EL}^+$ logics and facilitate discussion of the modularization-based approach.[3]

In Subsection 5.2.1, we investigate the inherently high complexity of axiom pinpointing in spite of tractability of standard reasoning. Subsection 5.2.2 is dedicated to the black-box approach. It shows that one MinA can always be computed in polynomial time, presents an improved black-box algorithm motivated by the binary search, and describes an algorithm based on the hitting-set tree search for computing all MinAs. The glass-box techniques for axiom pinpointing are discussed in Subsection 5.2.3. We describe an extension to the classification algorithm with the axiom labeling techniques and show that it computes all MinAs in exponential time. Additionally, a simplified version of the labeled algorithm is proposed that computes a small (possibly non-minimal) set of axioms. To cope with large-scale ontologies, we develop a novel approach based on the computation of reachability-based modules in Subsection 5.2.4.

[3]Unsatisfiability and subsumption are interreducible (see Section 2.3).

5.2.1 Complexity of axiom pinpointing

In this subsection, we show hardness results regarding the computation of all MinAs. In fact, we only use expressivity of the sublanguage $\mathcal{HL}$ of $\mathcal{EL}$ with the simplest form of terminology, i.e., unfoldable TBoxes.[4] These hardness results are immediately transfered to $\mathcal{EL}^+$ and $\mathcal{EL}^{++}$, as well as other Description Logics.

If we want to compute all MinAs, then in the worst case an exponential runtime cannot be avoided since there may be *exponentially many MinAs* for a given subsumption. The following example shows that this is already the case for acyclic $\mathcal{HL}$ TBoxes.

Example 47 (Exponentially many MinAs). For all $n \geq 1$, define $\mathcal{T}_n$ to be an acyclic $\mathcal{HL}$ TBox consisting of the following (primitive) concept definitions:

$$\begin{array}{lcl} A & \sqsubseteq & P_1 \sqcap Q_1; \\ P_i & \sqsubseteq & P_{i+1} \sqcap Q_{i+1}, \text{ for } 1 \leq i < n; \\ Q_i & \sqsubseteq & P_{i+1} \sqcap Q_{i+1}, \text{ for } 1 \leq i < n; \\ P_n & \sqsubseteq & B; \\ Q_n & \sqsubseteq & B \end{array}$$

The size of $\mathcal{T}_n$ is linear in n, and we have the consequence $A \sqsubseteq_{\mathcal{T}_n} B$. Obviously, there are 2^n MinAs for $A \sqsubseteq_{\mathcal{T}_n} B$ since, for each i such that $1 \leq i \leq n$, it suffices to have either P_i's or Q_i's definition. That is, either $P_i \sqsubseteq P_{i+1} \sqcap Q_{i+1}$ or $Q_i \sqsubseteq P_{i+1} \sqcap Q_{i+1}$ in the case that $i < n$, and either $P_n \sqsubseteq B$ or $Q_n \sqsubseteq B$ otherwise. $\dashv$

Though the standard reasoning of subsumption is polynomial, the combinatorial nature of axiom pinpointing makes it hard to deal with all possible MinAs. In fact, as soon as we want to know more about the properties of the set of *all* MinAs, this cannot be achieved in polynomial time (unless P=NP). For instance, determining whether there exists a MinA of which cardinality is bounded by a given natural number n is NP-hard.

Lemma 48 (Lower bound). *Let $\mathcal{T}$ be an acyclic $\mathcal{HL}$ TBox, A, B concept names occurring in $\mathcal{T}$, and n a natural number. It is NP-hard to decide whether or not there is a MinA for $A \sqsubseteq_{\mathcal{T}} B$ of cardinality $\leq n$.*

Proof. This can be shown by a reduction of the NP-hard *hitting set* problem [GJ79]: Given a collection $S_1, \ldots S_k$ of sets and a natural number n, determining whether or not there is a set H of cardinality $\leq n$ such that $S \cap S_i \neq \emptyset$ for $i = 1, \ldots, k$. Such a set H is called a *hitting set.* Given $S_1 = \{p_{11}, \ldots, p_{1\ell_1}\}, \ldots, S_k = \{p_{k1}, \ldots, p_{k\ell_k}\}$, we use a concept name P_{ij} for every element $p_{ij} \in S_1 \cup \ldots \cup S_k$ as well as the additional concept names $A, B, A_{11}, \ldots, A_{k\ell_k}, Q_1, \ldots, Q_k$. Define the acyclic $\mathcal{HL}$ TBox $\mathcal{T}$ as the smallest set containing:

$$\begin{array}{llcl} (1) & A & \sqsubseteq & \bigsqcap_{1 \leq i \leq k, 1 \leq j \leq \ell_i} A_{ij}; \\ (2) & A_{ij} & \sqsubseteq & P_{ij}, \text{ for } 1 \leq i \leq k, 1 \leq j \leq \ell_i; \\ (3) & P_{ij} & \sqsubseteq & Q_i, \text{ for } 1 \leq i \leq k, 1 \leq j \leq \ell_i; \\ (4) & B & \equiv & Q_1 \sqcap \ldots \sqcap Q_k \end{array}$$

[4]The sublanguage $\mathcal{HL}$ does not allow for roles and existential restrictions. The name is motivated by the fact that GCIs involving $\mathcal{HL}$ concepts are essentially propositional Horn clauses.

Our claim is that $S_1, \ldots, S_k$ has a hitting set of cardinality $\leq n$ if, and only if, there is a MinA for $A \sqsubseteq_{\mathcal{T}} B$ of cardinality $\leq n + k + 2$.

"Only if" Let H be a hitting set of cardinality $\leq n$. Define a subset $\mathcal{S} \subseteq \mathcal{T}$ to be the smallest set such that:

- axioms $(1), (4)$ are in $\mathcal{S}$;
- for each $p_{ij} \in H$, $A_{ij} \sqsubseteq P_{ij} \in \mathcal{S}$;
- for $1 \leq i \leq k$ and $p_{ij} \in H$, $P_{ij} \sqsubseteq Q_i \in \mathcal{S}$ if $P_{ij'} \sqsubseteq Q_i \notin \mathcal{S}$ for all $j' \neq j$.

Intuitively, we take exactly one axiom $P_{ij} \sqsubseteq Q_i$, for each i, where $A_{ij} \sqsubseteq P_{ij} \in \mathcal{S}$. This is possible since H is a hitting set. So, we have $A \sqsubseteq_{\mathcal{S}} B$. Obviously, $\mathcal{S}$ has cardinality $\leq n + k + 2$, thus there is a MinA $\mathcal{S}' \subseteq \mathcal{S}$ for $A \sqsubseteq_{\mathcal{T}} B$ of cardinality $\leq n + k + 2$.

"If" Let $\mathcal{S} \subseteq \mathcal{T}$ be a MinA for $A \sqsubseteq_{\mathcal{T}} B$ of cardinality $\leq n + k + 2$. Since $\mathcal{S}$ is a MinA, it contains (1),(4), and exactly k axioms of type (3), one for each Q_i. The remaining $\leq n$ axioms are of type (2). Define a set $H := \{p_{ij} \mid A_{ij} \sqsubseteq P_{ij} \in \mathcal{S}\}$. By construction, H is a hitting set, and its cardinality is $\leq n$. ❑

The following result is a direct consequence of the fact that subsumption in $\mathcal{EL}^{++}$ is decidable in polynomial time.

Lemma 49 (Upper bound). *Let $\mathcal{O}$ be an $\mathcal{EL}^{++}$ ontology , A, B concept names occurring in $\mathcal{T}$, and n a natural number. The problem of deciding whether or not there is a MinA for $A \sqsubseteq_{\mathcal{T}} B$ of cardinality $\leq n$ is in NP.*

Proof. A subset $\mathcal{S} \subseteq \mathcal{O}$ of cardinality n can be *guessed* in polynomial time and then *verified* whether the subsumption $A \sqsubseteq_{\mathcal{S}} B$ holds. Clearly, such a subset exists if, and only if, there is a MinA of cardinality $\leq n$. ❑

The lower bound for the DL $\mathcal{HL}$, in conjunction with the matching upper bound for the DL $\mathcal{EL}^{++}$, gives us NP-completeness of the decision problem in the logics $\mathcal{HL}$, $\mathcal{EL}$, $\mathcal{EL}^{+}$, and $\mathcal{EL}^{++}$:

Theorem 50. *Let $\mathcal{T}$ be a terminology ranging from acyclic $\mathcal{HL}$ TBox to an $\mathcal{EL}^{++}$ ontology. Also, let A, B be concept names occurring in $\mathcal{T}$, and n a natural number. Then, it is NP-complete to decide whether or not there is a MinA for $A \sqsubseteq_{\mathcal{T}} B$ of cardinality $\leq n$.*

In the context of output enumeration problems, in contrast to decision problems, it is often more sensible to consider a different kind of complexity. The so-called *enumeration complexity* for an enumeration problem has been introduced in [JPY88] which takes into account not only the input (in our context, the ontology) but also the output (the MinAs) of the algorithm. In a nutshell, there are three complexity classes listed as follows, ordered from better behavior to worse:

Polynomial delay algorithms generate all the MinAs, one after the other in some order, in such a way that the delay until the first MinA is output and between any two consecutive MinAs is bounded by a polynomial in the size of the ontology.

Incremental polynomial time algorithms can, given the ontology and a number of (pre-computed) MinAs, generate or determine nonexistence of another MinA in time polynomial in the combined size of the ontology and the given MinAs.

Polynomial total time algorithms generate all the MinAs in time polynomial in the combined size of the ontology and all those MinAs.

It is not hard to visualize the hierarchy of these three complexity classes, i.e., polynomial delay implies incremental polynomial time which in turn implies polynomial total time. Ideally, one would expect a system to be so responsive that it generates output with a polynomial delay. However, this notion is quite strong, and there is no known axiom pinpointing algorithm that runs in polynomial delay. The least that we could hope for is an algorithm that runs in polynomial total time.

It is still open as to which counting complexity class the problem of generating all MinAs belongs. However, if we allow the knowledge base to comprise not only *refutable* axioms but also *static* axioms (i.e., axioms believed to be flawless and ought not be refuted), a hardness result concerning this type of complexity is known. Intuitively, we are only interested in computing minimal sets of the refutable axioms, from which, together with static axioms, the consequence follows. A similar ontological setting, as well as potential application scenarios, is discussed earlier in Section 4.4. Formally, we define the slightly generalized axiom pinpointing problem as follows:

Definition 51 (MinA for $\mathcal{HL}$ duo-TBox). Let $\mathcal{T} = (\mathcal{T}_s \uplus \mathcal{T}_r)$ be an $\mathcal{HL}$ TBox and A, B concept names occurring in it such that $A \sqsubseteq_{\mathcal{T}} B$. Then, a *minimal axiom set (MinA) for* $A \sqsubseteq_{\mathcal{T}} B$ is a subset $\mathcal{S}$ of $\mathcal{T}_r$ such that $A \sqsubseteq_{\mathcal{T}_s \cup \mathcal{S}} B$, but $A \not\sqsubseteq_{\mathcal{T}_s \cup \mathcal{S}'} B$ for all strict subsets $\mathcal{S}' \subset \mathcal{S}$. ◇

Let $\mathcal{T}$, A, B be as defined in the definition above. It has been shown in [BPS07b] that there is *no* (unless P=NP) polynomial total time algorithm that computes all MinAs for $A \sqsubseteq_{\mathcal{T}} B$ with $\mathcal{T} = (\mathcal{T}_s \uplus \mathcal{T}_r)$ an $\mathcal{HL}$ duo-TBox.

All the results that have been presented so far suggest that axiom pinpointing is inherently hard, despite the tractability of the standard decision problems in the corresponding logics. It is readily evident that the source of complexity lies at the combinatorial nature of the set problem. In fact, a single MinA can be computed in polynomial time, provided that the underlying DL of interest is tractable. The next subsection has the details.

5.2.2 Black-box techniques

As mentioned in the previous subsection, a polynomial runtime can be attained if only a single MinA is required for a subsumption relationship. Algorithm 7 describes how this is achieved in an extremely naïve way. It goes through all axioms (in a given fixed order) and throws away those axioms that are not essential to obtain the subsumption relationship at hand. An axiom is not *essential* if the subsumption still holds in its absence. This method of computing MinAs is commonly known in the literature as the *black-box* approach. The term 'black box' refers to an existing DL reasoner used off the

Algorithm 7 Linear extraction of a MinA.

```
Procedure lin-extract-mina($A, B, \mathcal{O}$)
Input: $A, B$: concept names; $\mathcal{O}$: $\mathcal{EL}^+$ ontology
Output: MinA $\mathcal{S}$ for $A \sqsubseteq_{\mathcal{O}} B$
1: $\mathcal{S} := \mathcal{O}$
2: for each axiom $\alpha \in \mathcal{O}$ do
3:    if $A \sqsubseteq_{\mathcal{S} \setminus \{\alpha\}} B$ then
4:       $\mathcal{S} := \mathcal{S} \setminus \{\alpha\}$
5:    end if
6: end for
7: return $\mathcal{S}$
```

shelf without the need to be internally modified. The minimization algorithm exploits a DL reasoner by calling its standard inference services, for instance subsumption queries in this case. Since the algorithm performs n subsumption tests (where n is the number of axioms occurring the ontology) and each subsumption test takes only polynomial time, the overall complexity of this algorithm is polynomial. It is straightforward to see that its output actually is a MinA for $A \sqsubseteq_{\mathcal{O}} B$.

Theorem 52. *Given an $\mathcal{EL}^+$ ontology $\mathcal{O}$ and concept names A, B such that $A \sqsubseteq_{\mathcal{O}} B$, Algorithm 7 terminates in time polynomial in the size of $\mathcal{O}$, and yields a MinA for $A \sqsubseteq_{\mathcal{O}} B$.*

Although it requires polynomial time in theory, computing one MinA using Algorithm 7 does not perform well on an average-size ontology and is highly infeasible on large-scale ontologies like Snomed ct.[5] In fact, it has to make as many calls to the subsumption checking procedure as there are axioms in the ontology, which is almost four hundred thousands in the case of Snomed ct.

Improved pruning algorithms

There is actually a lot of room for improvement following the observation that the input ontology is normally much larger than the output MinA, i.e., most of the axioms are irrelevant to a particular entailment. One way to improve the naïve algorithm is to employ the so-called *sliding window* technique [KPHS07], in which a window of varying size is *sliding* through the axioms and those axioms in the window are pruned from the ontology. The special case where the window is of size one axiom boils down to Algorithm 7. This simple idea effectively reduces the number of subsumption tests because a large number of the axioms can be dispensed with during the minimization. It is not priori clear what size the sliding window should have. In [KPHS07], the authors start with window size n being either one tenth of the number of axioms or ten (which ever is greater) and shrink the window size by some factor when pruning is repeated. Whenever axioms in a block of window are not removed due to the presence

[5]The naïve minimization algorithm did not terminate on Snomed ct in 48 hours and took about 3 minutes to compute a MinA from NotGalen.

Algorithm 8 Logarithmic extraction of a MinA.

```
Procedure log-extract-mina($A, B, \mathcal{O}$)
Input: $A, B$: concept names; $\mathcal{O}$: $\mathcal{EL}^+$ ontology
Output: MinA for $A \sqsubseteq_{\mathcal{O}} B$
1: return log-extract-mina-r($A, B, \emptyset, \mathcal{O}$)

Procedure log-extract-mina-r($A, B, \mathcal{O}_s, \mathcal{O}_r$)
Input: $A, B$: concept names; $\mathcal{O}_s$: support set (of axioms); $\mathcal{O}_r$: $\mathcal{EL}^+$ ontology
Output: minimal subset $\mathcal{S} \subseteq \mathcal{O}_r$ such that $A \sqsubseteq_{\mathcal{O}_s \cup \mathcal{S}} B$
1: if $|\mathcal{O}_r| = 1$ then
2:    return $\mathcal{O}_r$
3: end if
4: $\mathcal{S}_1, \mathcal{S}_2 :=$ halve($\mathcal{O}_r$)
5: if $A \sqsubseteq_{\mathcal{O}_s \cup \mathcal{S}_1} B$ then
6:    return log-extract-mina-r($A, B, \mathcal{O}_s, \mathcal{S}_1$)
7: end if
8: if $A \sqsubseteq_{\mathcal{O}_s \cup \mathcal{S}_2} B$ then
9:    return log-extract-mina-r($A, B, \mathcal{O}_s, \mathcal{S}_2$)
10: end if
11: $\mathcal{S}'_1 :=$ log-extract-mina-r($A, B, \mathcal{O}_s \cup \mathcal{S}_2, \mathcal{S}_1$)
12: $\mathcal{S}'_2 :=$ log-extract-mina-r($A, B, \mathcal{O}_s \cup \mathcal{S}'_1, \mathcal{S}_2$)
13: return $\mathcal{S}'_1 \cup \mathcal{S}'_2$
```

of an essential axiom in it, pruning is repeated with a smaller window to single out this essential axiom. This process continues until the window size is one.

In [BS08], we have proposed to use the binary search technique of "halve and check" to prune axioms. The same idea has been employed in [BM07] to minimize valuation when computing prime implicates. Essentially, instead of going through the axioms one by one and pruning one axiom at a time, the algorithm partitions the ontology into two halves. It then employs a DL reasoner to check whether one of them entails the subsumption. If the answer is 'yes,' it immediately recurses on that half, throwing away half of the axioms in one step. Intuitively, this means that the essential axioms in a MinA are all in one of the partitions. Otherwise, i.e., the answer is 'no,' essential axioms spread over both partitions of the ontology. In this case, the algorithm recurses on each half, while using the other half as the "support set." Algorithm 8 outlines this pruning strategy in more detail, where the function halve partitions $\mathcal{O}$ into $\mathcal{S}_1 \uplus \mathcal{S}_2$ with $||\mathcal{S}_1| - |\mathcal{S}_2|| \leq 1$. Intuitively, the support set $\mathcal{O}_s$ carries axioms presumed to be essential for the subsumption. It is similar to the static component of a duo-TBox (see Definition 51 on page 96), in the sense that the function log-extract-mina-r minimizes $\mathcal{O}_r$ w.r.t. $A \sqsubseteq_{\mathcal{O}_s \cup \mathcal{O}_r} B$. The support set is needed when essential axioms appear in both halves of the ontology. While one half is being minimized, the other half is treated as if it were the static part of a duo-TBox.

It follows from the results in [BM07] that computing a MinA $\mathcal{S}$ for a given subsumption $A \sqsubseteq_{\mathcal{O}} B$ with Algorithm 8 requires $O\left((|\mathcal{S}| - 1) + |\mathcal{S}| log(|\mathcal{O}|/|\mathcal{S}|)\right)$ subsumption tests, which greatly improves on the naïve minimization algorithm. That is, the number of subsumption tests required by the algorithm is logarithmic in the size of the input (i.e., the ontology $\mathcal{O}$) but linear in the size of the output (i.e., the MinA $\mathcal{S}$). As its complexity suggests, the logarithmic minimization algorithm is effective only when MinAs are much smaller than the ontology, or its computational overhead may not pay off. In realistic ontologies, it is fortunately the case that MinAs are relatively very small. For instance, the only MinA for our amputation example in SNOMED CT consists of six axioms (see Figure 6.8 on page 128). Whereas Algorithm 7 requires the constant number of about four hundred thousand subsumption tests, Algorithm 8 needs only about one hundred subsumption tests. Besides, early subsumption tests required by the linear pruning algorithm are w.r.t. larger ontologies than late subsumption tests since we start with $\mathcal{O}$ and iteratively throw away axioms. With this algorithm, it is feasible to extract a small MinA from a large ontology. Subsection 6.2.5 discusses the performance of black-box algorithms for computing one MinA in SNOMED CT.

Computing all MinAs

Given an ontology and an unwanted entailment, a single MinA is not sufficient to remove the entailment from the ontology unless there exists only one MinA for the entailment in question. To check whether a given MinA $\mathcal{S}$ is unique for $A \sqsubseteq_{\mathcal{O}} B$, we simply check if the subsumption still holds in the ontology excluding an axiom α from $\mathcal{S}$, i.e., check if $A \sqsubseteq_{\mathcal{O}\setminus\{\alpha\}} B$ for each $\alpha \in \mathcal{S}$. Obviously, if the test is affirmative for some axiom $\alpha \in \mathcal{S}$, another MinA different from $\mathcal{S}$ must exist and can be obtained by pruning $\mathcal{O}\setminus\{\alpha\}$.

This can be generalized to the following problem: given an ontology $\mathcal{O}$, a subsumption $A \sqsubseteq_{\mathcal{O}} B$ and a collection of MinAs $\mathcal{S}_1, \ldots, \mathcal{S}_k$ for $A \sqsubseteq_{\mathcal{O}} B$, determine whether there exists another MinA $\mathcal{S}$ (different from all $\mathcal{S}_i$) for the subsumption. If any, it is easy to see that such a MinA $\mathcal{S}$ has to do away with an axiom α_i from each MinA $\mathcal{S}_i$, with $1 \leq i \leq k$. Otherwise, $\mathcal{S}$ is a superset of some $\mathcal{S}_i$ and thus not a MinA. Then, we need to search for a reduced ontology $\mathcal{O}\setminus\mathcal{H}$ with $\mathcal{H}$ a minimal set such that $\mathcal{H} \cap \mathcal{S}_i \neq \emptyset$ for all MinAs $\mathcal{S}_i$. If $A \sqsubseteq_{\mathcal{O}\setminus\mathcal{H}} B$, then another MinA $\mathcal{S}$ (different from all $\mathcal{S}_i$) exists and is contained in $\mathcal{O}\setminus\mathcal{H}$. Either Algorithm 7 or 8 can then be used to prune $\mathcal{O}\setminus\mathcal{H}$ and obtain a new MinA from the shrunk ontology.

Observe that searching for a set $\mathcal{H}$ is in fact the problem of computing *minimal hitting sets* (see [GJ79] and Definition 19 on page 30): given a collection $\mathcal{S}_1, \ldots \mathcal{S}_k$ of sets, compute sets $\mathcal{H}$ such that $\mathcal{H} \cap \mathcal{S}_i \neq \emptyset$ for all $i = 1, \ldots, k$, and there is no proper subset $\mathcal{H}' \subset \mathcal{H}$ with this property. The *hitting set tree (HST)* algorithm developed in [Rei87], as well as some optimization techniques introduced thereafter [GSW92], can be utilized in our context of axiom pinpointing.

A hitting set tree (HST) algorithm for axiom pinpointing

As mentioned in Section 2.4, the ontology $\mathcal{O}$ corresponds to the universal set, while the set of all MinAs $\mathcal{S}_1, \ldots, \mathcal{S}_k$ corresponds to the collection of subsets. The difference however is that, in axiom pinpointing, MinAs are not given but rather what need to be computed. These are computed with the help of a black-box sub-procedure (i.e., either Algorithm 7 or 8) on the fly, while the *hitting set tree (HST)* is being expanded. Every non-terminal node in the HST is labeled with a MinA, while every edge is labeled with an axiom $\alpha \in \mathcal{O}$ (more precisely, $\alpha \in \bigcup_{1\leq i\leq k} \mathcal{S}_i$). For each node n in the HST, $H(n)$ denotes the set of edge labels on the path from the root of the HST to the node n. Then, the label for n is a MinA $\mathcal{S}$ such that $\mathcal{S} \cap H(n) = \emptyset$, if such a MinA exists. In such a case (i.e., n is labeled by $\mathcal{S}$), n has a successor n_α for each axiom $\alpha \in \mathcal{S}$, and the edge $n \to n_\alpha$ is labeled with α. If there is no such MinA, n is a terminus and thus marked by $\odot$. In this case, $H(n)$ is a hitting set for the collection of all MinAs.[6]

Algorithm 9 describes in detail the pinpointing algorithm based on HST expansion. In the procedure hst-extract-all-minas, two global variables $\mathbf{C}$ and $\mathbf{H}$ are declared and initialized with $\emptyset$. They are used throughout the HST expansion to store the computed MinAs and hitting sets, respectively. In line 2, a first MinA $\mathcal{S}$ is computed with the help of the black-box sub-procedure log-extract-mina[7] and is taken as the label of the root. Branches from the root are spawned by calling the recursive procedure expand-hst for every axiom $\alpha \in \mathcal{S}$ (line 4–6).

Line 1 to 5 of expand-hst implement two optimizations for the HST algorithms [Rei87, GSW92, KPHS07] that help reduce the size of the HST and minimize calls to the black-box sub-procedure. We discuss here only those optimizations that are applicable to axiom pinpointing:[8]

Early path termination: This optimization is based on the observation that a superset of a hitting set is itself a hitting set. Hence, if the current node n in the HST is such that $H(n) \supseteq H'$ for some previously computed hitting set H', then $H(n)$ is known to be a hitting set as well. As a result, there is no need to call the black-box sub-procedure to check whether there is a MinA to label the node n. In such a case, the node n is immediately marked by $\otimes$, and the path is terminated.

Additionally, the current path to n can be terminated early if it is guaranteed that *all* possible paths starting with $H(n)$ have been considered previously in the HST. This is the case if there exists a prefix-path P of some previously computed hitting set H' such that $P = H(n)$.[9]

MinA reuse: Recall that any non-terminus node n in the HST has to be labeled with a MinA $\mathcal{S}$ such that $\mathcal{S} \cap H(n) = \emptyset$, if such a MinA exists. Without the "MinA reuse" optimization, such a MinA is computed by calling the black-box sub-procedure with a shrunk ontology $\mathcal{O}\backslash H(n)$ as input. The obtained MinA $\mathcal{S}$ satisfies the property stated above since $\mathcal{S} \subseteq \mathcal{O}\backslash H(n)$. To avoid calling to the

[6]Note that we do not need to know all the MinAs in order to compute a hitting set for all of them.

[7]This can be replaced by any black-box algorithm that computes one MinA, e.g., lin-extract-mina.

[8]E.g., an optimization designed to handle $\mathcal{S}_1 \subset \mathcal{S}_2$ in $\mathbf{C}$ cannot be employed here since any two MinAs are incomparable relative to set inclusion.

[9]A hitting set computed by the HST algorithm can be viewed as a *list* where its elements are ordered according to the path from the root to a leaf in the HST.

Algorithm 9 Hitting set tree (HST) pinpointing algorithm.

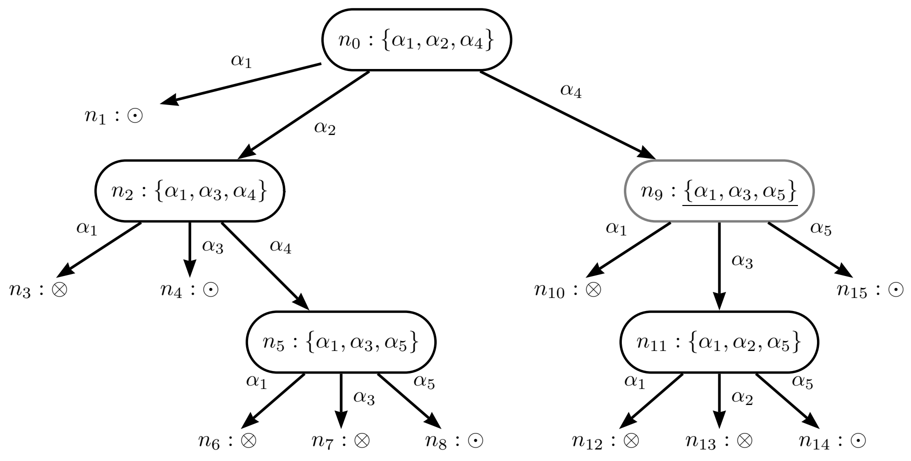
```
Procedure hst-extract-all-minas($A, B, \mathcal{O}$)
Input: $A, B$: concept names; $\mathcal{O}$: $\mathcal{EL}^+$ ontology
Output: collection $\mathbf{C}$ of all MinAs for $A \sqsubseteq_{\mathcal{O}} B$
 1: Global : $\mathbf{C}, \mathbf{H} := \emptyset$
 2: $\mathcal{S}$ := log-extract-mina($A, B, \mathcal{O}$)
 3: $\mathbf{C} := \{\mathcal{S}\}$
 4: for each axiom $\alpha \in \mathcal{S}$ do
 5:    expand-hst($A, B, \mathcal{O} \backslash \{\alpha\}, \{\alpha\}$)
 6: end for
 7: return $\mathbf{C}$

Procedure expand-hst($A, B, \mathcal{O}, H$)
Input: $A, B$: concept names; $\mathcal{O}$: $\mathcal{EL}^+$ ontology; $H$: list of edge labels
Side effects: modifications to $\mathbf{S}$ and $\mathbf{H}$
 1: if there exists some $H' \in \mathbf{H}$ such that $H' \subseteq H$ or
        $H'$ contains a prefix-path $P$ with $P = H$ then
 2:    return                                        (early path termination $\otimes$)
 3: end if
 4: if there exists some $\mathcal{S} \in \mathbf{C}$ such that $H \cap \mathcal{S} = \emptyset$ then
 5:    $\mathcal{S}' := \mathcal{S}$                                        (MinA reuse)
 6: else
 7:    $\mathcal{S}'$ := log-extract-mina($A, B, \mathcal{O}$)
 8: end if
 9: if $\mathcal{S}' \neq \emptyset$ then
10:    $\mathbf{C} := \mathbf{C} \cup \{\mathcal{S}'\}$
11:    for each axiom $\alpha \in \mathcal{S}'$ do
12:       expand-hst($A, B, \mathcal{O} \backslash \{\alpha\}, H \cup \{\alpha\}$)
13:    end for
14: else
15:    $\mathbf{H} := \mathbf{H} \cup \{H\}$                              (normal termination $\odot$)
16: end if
```

sub-procedure, an existing MinA $\mathcal{S}'$ such that $\mathcal{S}' \cap H(n) = \emptyset$ can be reused to label the node n.

If none of the two optimizations applies, the black-box sub-procedure needs to be invoked to compute another MinA $\mathcal{S}'$ (line 7). If such a MinA exists, expand-hst proceeds by spawning new edges, one for each of the axioms in $\mathcal{S}'$ (line 9–13). Otherwise, a hitting set is found, and the path terminates normally (line 15).

Example 53 (HST pinpointing algorithm). Consider the $\mathcal{HL}$ TBox $\mathcal{T}_n$ in Example 47 on page 94. With $n = 2$, $\mathcal{T}_2$ comprises 5 axioms as follows:

$$\begin{array}{llll} \alpha_1 : A \sqsubseteq P_1 \sqcap Q_1, & \alpha_2 : P_1 \sqsubseteq P_2 \sqcap Q_2, & \alpha_4 : P_2 \sqsubseteq B, \\ & \alpha_3 : Q_1 \sqsubseteq P_2 \sqcap Q_2, & \alpha_5 : Q_2 \sqsubseteq B; \end{array}$$



$n_0 : \{\alpha_1, \alpha_2, \alpha_4\}$
α_1
$n_1 : \odot$
α_2
α_4
$n_2 : \{\alpha_1, \alpha_3, \alpha_4\}$
$n_9 : \underline{\{\alpha_1, \alpha_3, \alpha_5\}}$
$n_3 : \otimes$
$n_4 : \odot$
$n_5 : \{\alpha_1, \alpha_3, \alpha_5\}$
$n_{10} : \otimes$
$n_{11} : \{\alpha_1, \alpha_2, \alpha_5\}$
$n_{15} : \odot$
$n_6 : \otimes$
$n_7 : \otimes$
$n_8 : \odot$
$n_{12} : \otimes$
$n_{13} : \otimes$
$n_{14} : \odot$

Figure 5.1: The hitting set tree produced by Algorithm 9 on Example 53.

and entails the subsumption $\sigma = (A \sqsubseteq B)$. As claimed in the referred example, there are $2^2 = 4$ MinAs for σ in $\mathcal{T}_2$. Figure 5.1 demonstrates the process of computing all the MinAs by the HST pinpointing algorithm (i.e., Algorithm 9). To begin with, $\mathcal{T}_2$ is pruned by log-extract-mina to obtain the first MinA $\{\alpha_1, \alpha_2, \alpha_4\}$, which is the label of the root node n_0 of the HST. The first branch terminates immediately since $\mathcal{T}_2 \backslash \{\alpha_1\}$ does not entail σ (marked by $\odot$ in n_1). On the other hand, $\mathcal{T}_2 \backslash \{\alpha_2\} \models \sigma$, and the second MinA $\{\alpha_1, \alpha_3, \alpha_4\}$ can be computed by pruning $\mathcal{T}_2 \backslash \{\alpha_2\}$ which labels n_2. When branching from n_2 with α_1, the optimization “early path termination” is triggered since $H(n_3) \supseteq H(n_1)$, preventing a call to log-extract-mina. In this case, n_3 is immediately labeled with $\otimes$, and the path is terminated early. This process continues to expand HST until it finds all other MinAs for σ. Observe that the underlined label of n_9 in the right-most branch is the result of the “MinA reuse” optimization, where $\{\alpha_1, \alpha_3, \alpha_5\}$ is taken from n_5. The total number of calls to log-extract-mina is the number of distinct nodes (i.e., 4) plus the number of termini marked by $\odot$ (i.e., 5).

$\dashv$

An obvious advantage of the black-box approaches is that the techniques can be used with any existing reasoners that support standard inference services. The only major drawback is its low performance on large-scale ontologies. In Subsection 5.2.4, we address this problem by presenting a highly effective optimization based the computation of reachability-based modules.

5.2.3 Glass-box techniques

An alternative to the approach presented in the previous subsection is known as the *glass-box* approach. Likewise, the term ‘glass box’ refers to a DL reasoner whose internal machinery is visible and of interest. In fact, the glass-box approach takes a decision procedure (e.g., tableau algorithm) that gives affirmative or negative answer to a logical entailment query, and modifies it to a generating procedure that produces as output a MinA (all MinAs) for a logical entailment in question.

The central tenet shared by all the previous work is that it extends a given decision procedure with labels that keep the information about relevant axioms used during the computation. A label may be a monotone Boolean formula or a set of sets of axioms. Most of the previous work [SC03, BH95, MLPB06, KPHS07] focused on tableau algorithms for specific DLs, while [BP07] considers a general tableau algorithm for arbitrary DLs. In this subsection, we employ the glass-box techniques in the $\mathcal{EL}^+$ classification algorithm and discuss related issues.

A labeled classification algorithm

The labeling technique developed for tableau-based algorithms can be applied to the polynomial-time classification algorithm (see Subsection 4.1.3). The approaches presented in [BH95, BPS07b] extend the standard reasoning algorithms by labeling each inferred assertion with a so-called *pinpointing formula*: a monotone Boolean formula from which all the MinAs can be derived. To facilitate the discussion that follows, we follow the approach used in [SC03, PSK05] that labels each assertion with a set of sets of (indices of) axioms. Such a set naturally corresponds to a pinpointing formula in disjunctive normal form.

Inferred assertions the $\mathcal{EL}^+$ classification algorithm computes are of the forms $B \in S(A)$ or $(A, B) \in R(r)$ (see Figure 4.2 on page 52). To facilitate describing the modifications to this algorithm, these two types of assertions are viewed as (A, B) and (A, r, B), respectively. In the *labeled* classification algorithm, each assertion π is labeled with a set ℓ of axioms in $\mathcal{O}$. The purpose of the label ℓ is to keep track of the information about the axioms that have been used (through rule applications) in order to produce π. Formally, we maintain *the set of labeled assertions*:

$$\mathcal{A}_{S,R} \subseteq \Big(\mathsf{CN}^\top(\mathcal{O}) \times (\mathsf{CN}^\top(\mathcal{O}) \cup \{\bot\}) \times 2^{\mathcal{O}}\Big) \cup \Big(\mathsf{CN}^\top(\mathcal{O}) \times \mathsf{RN}(\mathcal{O}) \times \mathsf{CN}^\top(\mathcal{O}) \times 2^{\mathcal{O}}\Big).$$

Observe that, unlike $S(\cdot)$ and $R(\cdot)$, an assertion (A, B) or (A, r, B) may occur in $\mathcal{A}_{S,R}$ several times with different labels $\ell \in 2^{\mathcal{O}}$. The labeled algorithm initializes $\mathcal{A}_{S,R}$ with the smallest set consisting of:

- $(A, A)^\emptyset$ and $(A, \top)^\emptyset$, for all concept names $A \in \mathsf{CN}^\top(\mathcal{O})$;
- $(A, r, A)^\ell$, for r a role name such that $\mathcal{O} \models \epsilon \sqsubseteq r$, where ℓ is a minimal set of axioms containing $\epsilon \sqsubseteq s \in \mathcal{O}$ and a chain of role inclusion axioms $r_i \sqsubseteq r_{i+1} \in \mathcal{O}$ with $0 \leq i < n$, $r_0 = s$ and $r_n = r$.

Note that in the presence of reflexive roles, the labeled algorithm may initialize $\mathcal{A}_{S,R}$ with exponentially many labeled assertions. The following example demonstrates this:

Example 54. Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology containing the following role inclusion axioms:

$$\begin{array}{lll} \epsilon \sqsubseteq u_0, & \epsilon \sqsubseteq v_0; & \\ u_i \sqsubseteq u_{i+1}, & u_i \sqsubseteq v_{i+1}, & \text{for } 0 \leq i < n; \\ v_i \sqsubseteq u_{i+1}, & v_i \sqsubseteq v_{i+1}, & \text{for } 0 \leq i < n; \\ u_n \sqsubseteq r, & v_n \sqsubseteq r & \end{array}$$

CR1 If $\{(X, A_1)^{\ell_1}, \dots, (X, A_n)^{\ell_n}\} \subseteq \mathcal{A}_{S,R}$, $\alpha : A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B \in \mathcal{O}$
and there is *no* $(X, B)^{\ell} \in \mathcal{A}_{S,R}$ such that $\ell \subseteq \ell_1 \cup \cdots \cup \ell_n \cup \{\alpha\}$
then $\mathcal{A}_{S,R} := \mathcal{A}_{S,R} \oplus (X, B)^{\ell_1 \cup \cdots \cup \ell_n \cup \{\alpha\}}$

CR2 If $(X, A)^{\ell_1} \in \mathcal{A}_{S,R}$, $\alpha : A \sqsubseteq \exists r.B \in \mathcal{O}$,
and there is *no* $(X, r, B)^{\ell} \in \mathcal{A}_{S,R}$ such that $\ell \subseteq \ell_1 \cup \{\alpha\}$
then $\mathcal{A}_{S,R} := \mathcal{A}_{S,R} \oplus (X, r, B)^{\ell_1 \cup \{\alpha\}}$

CR3 If $\{(X, r, Y)^{\ell_1}, (Y, A)^{\ell_2}\} \subseteq \mathcal{A}_{S,R}$, $\alpha : \exists r.A \sqsubseteq B \in \mathcal{O}$,
and there is *no* $(X, B)^{\ell} \in \mathcal{A}_{S,R}$ such that $\ell \subseteq \ell_1 \cup \ell_2 \cup \{\alpha\}$
then $\mathcal{A}_{S,R} := \mathcal{A}_{S,R} \oplus (X, B)^{\ell_1 \cup \ell_2 \cup \{\alpha\}}$

CR4 If $\{(X, r, Y)^{\ell_1}, (Y, \bot)^{\ell_2}\} \subseteq \mathcal{A}_{S,R}$,
and there is *no* $(X, \bot)^{\ell}$ such that $\ell \subseteq \ell_1 \cup \ell_2$
then $\mathcal{A}_{S,R} := \mathcal{A}_{S,R} \oplus (X, \bot)^{\ell_1 \cup \ell_2}$

CR5 If $(X, r, Y)^{\ell_1} \in \mathcal{A}_{S,R}$, $\alpha : r \sqsubseteq s \in \mathcal{O}$,
and there is *no* $(X, s, Y)^{\ell}$ such that $\ell \subseteq \ell_1 \cup \{\alpha\}$
then $\mathcal{A}_{S,R} := \mathcal{A}_{S,R} \oplus (X, s, Y)^{\ell_1 \cup \{\alpha\}}$

CR6 If $\{(X, r, Y)^{\ell_1}, (Y, s, Z)^{\ell_2} \subseteq \mathcal{A}_{S,R}$, $\alpha : r \circ s \sqsubseteq t \in \mathcal{O}$,
and there is *no* $(X, t, Z)^{\ell}$ such that $\ell \subseteq \ell_1 \cup \ell_2 \cup \{\alpha\}$
then $\mathcal{A}_{S,R} := \mathcal{A}_{S,R} \oplus (X, t, Z)^{\ell_1 \cup \ell_2 \cup \{\alpha\}}$

Figure 5.2: Labeled completion rules.

Obviously, the number of role inclusion axioms is linear in n, and $\mathcal{O} \models \epsilon \sqsubseteq r$. There are overall 2^{n+1} minimal chains of role inclusion axioms, i.e., $\epsilon \sqsubseteq r_0 \sqsubseteq \cdots \sqsubseteq r_n \sqsubseteq r$ where $r_i \in \{u_i, v_i\}$. $\dashv$

After initialization, the set of assertions $\mathcal{A}_{S,R}$ (equivalent to $S(\cdot)$ and $R(\cdot)$ in Figure 4.2 on page 52) is extended by applying the *labeled* completion rules shown in Figure 5.2 until no more rule applies. The update operator $\oplus$ is defined as follows:

$$\mathcal{A}_{S,R} \oplus \pi^{\ell} \equiv (\mathcal{A}_{S,R} \backslash \{\pi^{\ell'} \mid \ell \subset \ell'\}) \cup \{\pi^{\ell}\}$$

Intuitively, $\oplus$ not only adds a new labeled assertion but also makes sure that only minimal labels are maintained. Observe that a labeled completion rule may be applied several times to assert the same assertion but with distinct labels. We denote by $\mathcal{A}^*_{S,R}$ the completed set of labeled assertions obtained after the labeled classification algorithm terminates and by $lab^*(\pi)$ the set of π's labels in $\mathcal{A}^*_{S,R}$, i.e., $\{\ell \mid \pi^{\ell} \in \mathcal{A}^*_{S,R}\}$. In contrast to the (unlabeled) classification algorithm, the labeled algorithm may not terminate after a polynomial number of rule applications.

Lemma 55 (Termination). *Given an $\mathcal{EL}^+$ ontology $\mathcal{O}$ in normal form. The labeled classification algorithm terminates in time exponential in the size of $\mathcal{O}$.*

Proof. Each rule application updates $\mathcal{A}_{S,R}$ with the operator $\oplus$. An existing assertion $\pi^{\ell'}$ in $\mathcal{A}_{S,R}$ may be removed when a new assertion π^{ℓ} has a proper subset $\ell \subset \ell'$ as its label. Together with the preconditions of the labeled completion rules, it is ensured that each labeled assertion is added to $\mathcal{A}_{S,R}$ at most once. Unlike in the unlabeled classification algorithm, there are exponentially many labeled assertions due to exponentially many labels. Hence, the number of rule applications is exponential in the size of $\mathcal{O}$. Each rule application evidently takes time at most exponential in the size of $\mathcal{O}$. ❑

To show that the labeled classification algorithm is correct, we adopt the ω-*projection* approach used in [BH95]. The idea is to establish the connection between application of the labeled completion rules (see Figure 5.2), starting with an ontology $\mathcal{O}$, on the one hand, and application of the (unlabeled) completion rules (see Figure 4.2 on page 52), starting with a sub-ontology $\mathcal{S} \subseteq \mathcal{O}$, on the other hand.

Definition 56 ($\mathcal{S}$-projection). Let $\mathcal{S} \subseteq \mathcal{O}$ be a sub-ontology, and $\mathcal{A}_{S,R}$ a set of labeled assertions. The $\mathcal{S}$-projection of $\mathcal{A}_{S,R}$ (for short, $\mathcal{S}(\mathcal{A}_{S,R})$) is obtained from $\mathcal{A}_{S,R}$ by removing all assertions whose labels are not subsets of $\mathcal{S}$. (That is, $\mathcal{S}(\mathcal{A}_{S,R}) := \{\pi \mid \pi^{\ell} \in \mathcal{A}_{S,R}$ and $\ell \subseteq \mathcal{S}\}$.) ◇

In what follows, we view $R(\cdot)$ and $S(\cdot)$ in the *unlabeled* classification algorithm as a single set of *unlabeled* assertions in a straightforward way. Also, let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology in normal form (without range restrictions) with which the labeled classification algorithm starts, and $\mathcal{S} \subseteq \mathcal{O}$ a sub-ontology.

In the following, soundness and completeness of the labeled classification algorithm are shown by projecting applications of labeled completion rules to those of unlabeled completion rules, and reusing the known soundness and completeness results of the unlabeled classification algorithm [BBL05].

Lemma 57 (Soundness). *Let $\mathcal{A}_{S,R}$ and $\mathcal{A}'_{S,R}$ be sets of labeled assertions such that $\mathcal{A}'_{S,R}$ is obtained from $\mathcal{A}_{S,R}$ by applying one of the labeled completion rule. Then, one of the following holds: either $\mathcal{S}(\mathcal{A}'_{S,R}) = \mathcal{S}(\mathcal{A}_{S,R})$, or $\mathcal{S}(\mathcal{A}'_{S,R})$ is obtained from $\mathcal{S}(\mathcal{A}_{S,R})$ by applying the corresponding (unlabeled) completion rule.*

Proof. Assume that **CR1** is applied to $\{(X, A_1)^{\ell_1}, \ldots, (X, A_n)^{\ell_n}\} \subseteq \mathcal{A}_{S,R}$ such that $\alpha : A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B \in \mathcal{O}$. Let $\ell' = \ell_1 \cup \cdots \cup \ell_n \cup \{\alpha\}$. There are two possibilities:

- First, $\ell' \not\subseteq \mathcal{S}$. In this case, we show that $\mathcal{S}(\mathcal{A}'_{S,R}) = \mathcal{S}(\mathcal{A}_{S,R})$ by making a case distinction:
  - If no $(X, B)^{\ell}$ is in $\mathcal{A}_{S,R}$ such that $\ell \subseteq \mathcal{S}$, then (X, B) is not in $\mathcal{S}(\mathcal{A}_{S,R})$ by the definition of $\mathcal{S}$-projection (Definition 56). After rule application, we have $(X, B)^{\ell'} \in \mathcal{A}'_{S,R}$. Since $\ell' \not\subseteq \mathcal{S}$, the $\mathcal{S}$-projection definition again implies that (X, B) is not in $\mathcal{S}(\mathcal{A}'_{S,R})$.
  - If there exists an $\ell \subseteq \mathcal{S}$ such that $(X, B)^{\ell}$ is in $\mathcal{A}_{S,R}$, then (X, B) is already in $\mathcal{S}(\mathcal{A}_{S,R})$ by the definition of $\mathcal{S}$-projection. The new assertion $(X, B)^{\ell'}$ cannot replace $(X, B)^{\ell}$ after the rule application since $\ell' \not\subset \ell$. Thus, we have $(X, B)^{\ell} \in \mathcal{A}'_{S,R}$, implying by the $\mathcal{S}$-projection definition that (X, B) remains in $\mathcal{S}(\mathcal{A}'_{S,R})$.

- Second, $\ell' \subseteq \mathcal{S}$. This, together with the definition of $\mathcal{S}$-projection, implies that $\{(X, A_1), \ldots, (X, A_n)\} \subseteq \mathcal{S}(\mathcal{A}_{S,R})$. Since $\mathcal{A}'_{S,R}$ is obtained by updating $\mathcal{A}_{S,R}$ with $(X, B)^{\ell'}$, we know that (X, B) is contained in $\mathcal{S}(\mathcal{A}'_{S,R})$. If this assertion is already present in $\mathcal{S}(\mathcal{A}_{S,R})$, we have $\mathcal{S}(\mathcal{A}_{S,R}) = \mathcal{S}(\mathcal{A}'_{S,R})$. Otherwise, $\mathcal{S}(\mathcal{A}'_{S,R})$ is obtained from $\mathcal{S}(\mathcal{A}_{S,R})$ by applying the unlabeled **CR1**.

All other cases can be demonstrated in an analogous way. ❑

The next lemma states that the projected set of assertions $\mathcal{S}(\mathcal{A}_{S,R})$ is also complete. This means, in particular, that the subsumer sets $S(\cdot)$ are complete.

Lemma 58 (Completeness). *Let $\mathcal{A}^*_{S,R}$ be the complete set of labeled assertions to which none of the labeled completion rules applies. Then, none of the (unlabeled) completion rules applies to $\mathcal{S}(\mathcal{A}^*_{S,R})$.*

Proof. We consider only **CR1**; all other cases can be demonstrated in an analogous way. Assume that the assertions $(X, A_1), \ldots, (X, A_n)$ are present in $\mathcal{S}(\mathcal{A}^*_{S,R})$ and $\alpha : A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B \in \mathcal{O}$. To show: the unlabeled **CR1** cannot be applied to $\mathcal{S}(\mathcal{A}^*_{S,R})$. There are two possibilities for $\mathcal{S}$:

- $\alpha \notin \mathcal{S}$. In this case, the preconditions of **CR1** are not fulfilled, and thus the rule is *not* applicable.

- $\alpha \in \mathcal{S}$. Since $(X, A_1), \ldots, (X, A_n)$ are present in $\mathcal{S}(\mathcal{A}^*_{S,R})$, there must be labeled assertions $(X, A_i)^{\ell_i} \in \mathcal{A}^*_{S,R}$, for $i \leq n$, such that $\ell_i \subseteq \mathcal{S}$. Completeness of $\mathcal{A}^*_{S,R}$ implies that the labeled **CR1** is not applicable. For this reason, it must be the case that $\mathcal{A}^*_{S,R}$ contains $(X, B)^{\ell}$ with $\ell \subseteq \ell_1 \cup \cdots \cup \ell_n \cup \{\alpha\}$. Since $\ell_i \subseteq \mathcal{S}$ (for all i) and $\alpha \in \mathcal{S}$, we have $\ell \subseteq \mathcal{S}$. By the definition of $\mathcal{S}$-projection (Definition 56), (X, B) is contained in $\mathcal{S}(\mathcal{A}^*_{S,R})$. This concludes that the unlabeled **CR1** is *not* applicable to $\mathcal{S}(\mathcal{A}^*_{S,R})$.

❑

Putting the soundness and completeness together, the following three properties can be established.

Theorem 59. *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology in normal form (without range restrictions), and $\mathcal{A}^*_{S,R}$ the completed set of labeled assertions obtained after termination of the labeled classification algorithm on $\mathcal{O}$. Then, the following hold for all concept names A, B occurring in $\mathcal{O}$:*

1. *if $\{(A, B)^{\ell}, (A, \bot)^{\ell}\} \cap \mathcal{A}^*_{S,R} \neq \emptyset$, then $A \sqsubseteq_{\ell} B$,*

2. *if $A \sqsubseteq_{\mathcal{S}} B$ for a sub-ontology $\mathcal{S} \subseteq \mathcal{O}$, then there exists an $\ell \subseteq \mathcal{S}$ such that $\{(A, B)^{\ell}, (A, \bot)^{\ell}\} \cap \mathcal{A}^*_{S,R} \neq \emptyset$; and*

3. *if $\{\pi^{\ell}, \pi^{\ell'}\} \subseteq \mathcal{A}^*_{S,R}$, then ℓ and ℓ' are "$\subseteq$"-incomparable.*

Proof. Point 1 follows from Lemma 57 and the soundness of the classification algorithm (the "only if" direction of Theorem 27 on page 52). In fact, the ℓ-projection of $\mathcal{A}^*_{S,R}$ contains either (A, B) or $(A, \bot)$. Due to soundness of the unlabeled algorithm, we know that $A \sqsubseteq_\ell B$.

For Point 2, we show the contraposition. Assume that there is *no* such $\ell \subseteq \mathcal{S}$ with $\{(A, B)^\ell, (A, \bot)^\ell\} \cap \mathcal{A}^*_{S,R} \neq \emptyset$. By Definition 56 on page 105, we have that $\{(A, B), (A, \bot)\} \cap \mathcal{S}(\mathcal{A}^*_{S,R}) = \emptyset$. By Lemma 58, $\mathcal{S}(\mathcal{A}^*_{S,R})$ is the complete result of running the unlabeled classification algorithm on $\mathcal{S}$. The completeness of the unlabeled algorithm (the "if" direction of Theorem 27 on page 52) thus implies that $A \not\sqsubseteq_{\mathcal{S}} B$, as required.

For Point 3, it suffices to show that, after initialization and after each application of a labeled completion rule, no two labels ℓ and ℓ' of the same assertion π are "$\subseteq$"-comparable. Recall that $\mathcal{A}^*_{S,R}$ is initialized with the labeled assertions $(A, A)^\emptyset$, $(A, \top)^\emptyset$ and $(A, r, A)^\ell$ if $\mathcal{O} \models \epsilon \sqsubseteq r$. Since $\emptyset$ is the only label for $\pi \in \{(A, A), (A, \top)\}$ and different labels ℓ for $\pi = (A, r, A)$ must be minimal sets, Point 3 is satisfied after initialization. After a rule application, a new label ℓ cannot be a superset (an equivalent) of an existing label ℓ' since the last precondition of each labeled completion rule has to be satisfied. If $\ell \subset \ell'$, the update operator $\oplus$ makes sure that ℓ' is then removed, thus satisfying Point 3. ❑

Intuitively, the three points of Theorem 59 together imply that, after termination of the labeled classification algorithm, $lab^*(A, B)$ is precisely the set of all MinAs for $A \sqsubseteq_{\mathcal{O}} B$. Viewing (the index of) each axiom as a propositional variable, $lab^*(A, B)$ is a pinpointing formula for $A \sqsubseteq_{\mathcal{O}} B$. Pinpointing formulas constructed this way are in disjunctive normal form and are minimal, in the sense that $\ell_1 \not\subseteq \ell_2$ for all disjuncts ℓ_1, ℓ_2 in $\phi(\pi)$. In other words, every prime implicant of $lab^*(A, B)$ is precisely one of its disjuncts.[10]

De-normalization

Though producing minimal axiom sets, the labeled classification algorithm works on an ontology in normal form. These sets cannot immediately be given to the ontology developer as explanations to the entailment in question since they consist of normalized axioms that differ from original ones and may use new concept/role names unknown to the developer.

In order to obtain MinAs in original axioms, we maintain a possibly many-to-many mapping from normalized axioms to their original source axioms. Formally, define a function $\mathsf{denorm} : \widetilde{\mathcal{O}} \rightarrow 2^{\mathcal{O}}$, with $\mathcal{O}$ an $\mathcal{EL}^+$ ontology and $\widetilde{\mathcal{O}}$ its normal form. For brevity, we write $\mathsf{denorm}(\ell)$, with ℓ a set of normalized axioms, to denote $\bigcup_{\alpha \in \ell} \mathsf{denorm}(\alpha)$. Such a function can be constructed on the fly during normalization (see Figure 4.1 on page 48). Given a labeled assertion $(A, B)^\ell \in \mathcal{A}^*_{S,R}$ with $A, B \in \mathsf{CN}^\top(\mathcal{O})$, it is not hard to show that $A \sqsubseteq_{\mathcal{O}'} B$ with $\mathcal{O}' = \mathsf{denorm}(\ell)$. The resulting subset $\mathcal{O}'$ of the original ontology $\mathcal{O}$, however, need *not* be minimal. Non-minimality

[10] A prime implicant of a (monotone) propositional formula ϕ is the smallest conjunction of (positive) literals implying ϕ [Qui52].

is caused by the fact that a source axiom ax may be broken down to multiple axioms $\alpha_1, \ldots, \alpha_n$ in normal form.

Example 60. Let $\mathcal{O} = \{ax_1 : A \sqsubseteq B \sqcap C, ax_2 : C \sqsubseteq B\}$ be an $\mathcal{EL}^+$ ontology, and $\widetilde{\mathcal{O}} = \{\alpha_1 \mapsto \{ax_1\} : A \sqsubseteq B, \alpha_2 \mapsto \{ax_1\} : A \sqsubseteq C, \alpha_3 \mapsto \{ax_2\} : C \sqsubseteq B\}$ its normal form with denorm mapping information. Then, the labeled classification algorithm produces the completed set of assertions $\mathcal{A}^*_{S,R}$ containing $(A, B)^{\{\alpha_1\}}$ and $(A, B)^{\{\alpha_2,\alpha_3\}}$. Though both $\{\alpha_1\}$ and $\{\alpha_2, \alpha_3\}$ are MinAs w.r.t. $\widetilde{\mathcal{O}}$, only the former is mapped to a MinA w.r.t. $\mathcal{O}$, i.e., $\{ax_1\}$. The latter is mapped to a non-minimal subset $\{ax_1, ax_2\}$, thus not a MinA w.r.t. $\mathcal{O}$. ⊣

This does not cause a problem in principle since all the MinAs in normal form are readily available. In fact, we can simply check whether a de-normalized one is minimal without requiring additional DL reasoning. More precisely, given a label $\ell \in lab^*(A, B)$, a subset $\mathcal{S} = \mathsf{denorm}(\ell)$ of the original ontology is minimal if there is no $\ell' \neq \ell$ in $lab^*(A, B)$ such that $\mathsf{denorm}(\ell') \subset \mathcal{S}$.

Theorem 61 (Labeled algorithm computes all MinAs). *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology, $\widetilde{\mathcal{O}}$ its normal form, and* denorm *as defined above. Also, let $\mathcal{A}^*_{S,R}$ be the complete set of labeled assertions obtained after the labeled classification algorithm terminates on $\widetilde{\mathcal{O}}$. Then, the following are equivalent:*

1. *$\mathcal{S}$ is a MinA for $A \sqsubseteq_{\mathcal{O}} B$, and*
2. *there exists an $(A, B)^{\ell} \in \mathcal{A}^*_{S,R}$ such that $\mathcal{S} = \mathsf{denorm}(\ell)$ and there is no $(A, B)^{\ell'} \in \mathcal{A}^*_{S,R}$ with $\mathsf{denorm}(\ell) \subset \mathcal{S}$.*

Keeping a single label

Though termination of the labeled classification algorithm is attained, we helplessly lose tractability of the originally polytime algorithm. This is a direct consequence of our necessary modification to the preconditions of the labeled completion rules. Readily having an assertion in $\mathcal{A}_{S,R}$, for instance $(A, B)^{\ell}$, does not guarantee inapplicability of the rule that involves adding $(A, B)^{\ell'}$. In other words, not only assertions proper but also their labels can determine applicability of the labeled completion rules. Obviously, one cannot expect a polynomial-time algorithm for computing *all* MinAs, given the complexity results in Subsection 5.2.1. However, provision of one MinA in practically acceptable runtime is already useful in many ontology applications.

Therefore, we propose a small modification to the labeled classification algorithm such that it labels each assertion π with precisely one label (i.e., a conjunction of propositional variables). This is done by strengthening the preconditions of completion rule applicability. Precisely, we modify the last precondition clause of every completion rule to "there is *no* π^{ℓ} for some ℓ." Moreover, the update operation $\oplus$ boils down to addition of the new labeled assertion to $\mathcal{A}_{S,R}$. In what follows, we refer to this variant the *single-labeled* classification algorithm.

Observe that termination in polynomial time is regained. In fact, each rule application corresponds to that of the abstract classification algorithm (see Subsection 4.1.3).

The only difference is that the single-labeled algorithm aggregates tracing information of axioms used to obtain an assertion during rule applications. Unfortunately, the set of axioms (in normal form) corresponding to a computed label is not minimal in general. An example demonstrating this is given below:

Example 62. Let the ontology $\mathcal{O}$ consist of the following GCIs:

$$\begin{array}{lll} \alpha_1 : A \sqsubseteq X, & \alpha_2 : X \sqsubseteq \exists r.Y, & \alpha_3 : \exists r.Z \sqsubseteq B \\ \alpha_4 : Y \sqsubseteq X, & \alpha_5 : X \sqsubseteq B, & \alpha_6 : B \sqsubseteq Z \end{array}$$

After an application of $\mathsf{CR1}$ w.r.t. α_1 and $\mathsf{CR2}$ w.r.t. α_2, the assertion $(A, r, Y)^{\{\alpha_1,\alpha_2\}}$ is added to $\mathcal{A}_{S,R}$. Similarly, we obtain $(Y, Z)^{\{\alpha_4,\alpha_5,\alpha_6\}} \in \mathcal{A}_{S,R}$ after three applications of $\mathsf{CR1}$ w.r.t. α_4, α_5 and α_6, consecutively. Both inferred assertions, together with α_3, enable applicability of $\mathsf{CR3}$, adding the assertion $(A, B)^\ell$ to $\mathcal{A}_{S,R}$ with $\ell = \{\alpha_1, \dots, \alpha_6\}$. However, ℓ is not minimal since the subset $\{\alpha_1, \alpha_5\} \subseteq \ell$ also entails $A \sqsubseteq B$. ⊣

In Example 62, the computation would not have terminated, had we employed the full labeled classification algorithm. Instead, the labeled assertion $(A, B)^{\{\alpha_1,\alpha_5\}}$ would eventually have emerged and replaced $(A, B)^\ell$ in $\mathcal{A}_{S,R}$ via the update operator $\oplus$.

Even if the single-labeled classification algorithm yielded a MinA in normal form, we would still have to deal with another source of non-minimality as discussed above. With a single MinA in normal form, it is impossible to check whether or not the resulting set of original axioms is minimal (unless additional DL reasoning is employed). Given an assertion $(A, B)^\ell$ in the completed set of assertions $\mathcal{A}^*_{S,R}$ after termination of the single-labeled classification algorithm, for $A, B \in \mathsf{CN}^\top(\mathcal{O})$, we call the set $\mathsf{denorm}(\ell)$ a *non-minimal axiom set (nMinA)* for $A \sqsubseteq_\mathcal{O} B$. Note that an nMinA satisfies the first but not necessarily the second property for being a MinA (c.f. Definition 17 on page 29).

The black-box approach (either Algorithm 7 on page 97 or Algorithm 8 on page 98) can then be utilized to minimize an nMinA to obtain a MinA. Experiments described in [BPS07b] have shown that this approach works well on medium-size ontologies.[11]

A labeled subsumption algorithm

Both labeled and single-labeled classification algorithms presented above compute all MinAs (an nMinA) for *every* subsumption relationships in the ontology. This is a legacy of the abstract classification algorithm. In the ontology debugging scenario, it is usually the case that the ontology developer focuses on a single entailment (subsumption in this context). The labeled and single-labeled classification algorithms are thus not economical in this sense. Additionally, in order to carry out a fair empirical comparison between the *glass-box optimized* and *modularization optimized* (details in the subsequent subsection) black-box approaches to axiom pinpointing, we need to modify our glass-box algorithm such that it focuses on the subsumption in question.

[11]It failed on Snomed ct with memory exhaustion since each of all the assertions (more than 5 million) is equipped with a label. This requires considerably more space.

Algorithm 10 Glass-box optimized black-box (GOB) pinpointing algorithm.

```
Procedure gob-alg($A, B, \mathcal{O}$)
Input: $A, B$: concept names; $\mathcal{O}$: $\mathcal{EL}^+$ ontology
Output: $\mathcal{S}$: MinA for $A \sqsubseteq_{\mathcal{O}} B$
1: $\mathcal{S}$ := lab-subsumes?($A, B, \mathcal{O}$)
2: return lin-extract-mina($A, B, \mathcal{S}$)
```

To this end, we propose a *goal-directed, single-labeled* subsumption algorithm that is obtained by employing the single-labeling technique of the previous algorithm in the goal-directed subsumption algorithm (see Subsection 4.2). Directed by the goal subsumption, the algorithm activates only relevant concepts (in the sense of Theorem 42 on page 86) and terminates as soon as the subsumption is proved to hold. Soundness of the goal-directed subsumption algorithm (see Point 1 of Theorem 32 on page 65) and Lemma 57 imply the following corollary:

Corollary 63. *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology in normal form, $A, B \in \mathsf{CN}(\mathcal{O})$, and $\mathcal{A}^*_{S,R}$ the set of labeled assertions obtained after termination of the single-labeled subsumption algorithm on $\mathcal{O}$. Then, if $(A, B)^\ell \in \mathcal{A}^*_{S,R}$, then $A \sqsubseteq_\ell B$.*

Like the single-labeled classification algorithm, however, the assertion's label in $\mathcal{A}^*_{S,R}$ does not necessarily represent a minimal set. But, this is not the only source of non-minimality. Example 60 on page 108 demonstrates that de-normalization potentially yields a non-minimal set of original axioms $\mathsf{denorm}(\ell)$ even though the set of normalized axioms ℓ is minimal. Let lab-subsumes? be the single-labeled variant of Algorithm 2 on page 64 that takes as input an ontology $\mathcal{O}$ and subsumption $A \sqsubseteq B$, and returns an nMinA for $A \sqsubseteq_{\mathcal{O}} B$ if the subsumption holds or the empty set otherwise. Algorithm 10 outlines our *Glass-box Optimized Black-box (GOB)* algorithm.[12]

5.2.4 Modularization-based approach

In the GOB approach presented above, the labeled subsumption algorithm (which computes an nMinA) can be seen as an optimization to the blackbox algorithm. We employ the same idea in our novel approach based on the computation of reachability-based modules (see Subsection 5.1.1). By Corollary 41 on page 86, we know that $A \sqsubseteq_{\mathcal{O}} B$ if, and only if, $A \sqsubseteq_{\mathcal{O}^{\mathsf{reach}}_A} B$. It implies that $\mathcal{O}^{\mathsf{reach}}_A$ contains at least a MinA for $A \sqsubseteq_{\mathcal{O}} B$, and thus it is sufficient to consider only axioms in the module when extracting a MinA. Algorithm 11 outlines our Modularization Optimized Black-box (MOB) algorithm mob-alg. Since reachability-based modules are usually quite small (see [Sun08, CHKS07] and experimental results described in Subsection 6.2.4) but not so small as nMinAs yielded by the labeled subsumption algorithm, it is not a priori clear whether using the more complicated logarithmic minimization really pays off. Refer to [BS08] and Subsection 6.2.5 for some evaluation results and discussion.

[12]The linear and logarithmic minimization algorithms are interchangeable in the GOB approach. Due to usually very small size of nMinAs, however, the former has proved more efficient and thus used in our evaluations.

Algorithm 11 Modularization optimized black-box (MOB) pinpointing algorithm.

```
Procedure mob-alg($A, B, \mathcal{O}$)
Input: $A, B$: concept names; $\mathcal{O}$: $\mathcal{EL}^+$ ontology
Output: MinA for $A \sqsubseteq_{\mathcal{O}} B$
1: $\mathcal{O}_A^{\mathsf{reach}}$ := extract-module($\mathcal{O}, \{A\}$)
2: return log-extract-mina($A, B, \mathcal{O}_A^{\mathsf{reach}}$)

Procedure mob-hst-alg($A, B, \mathcal{O}$)
Input: $A, B$: concept names; $\mathcal{O}$: $\mathcal{EL}^+$ ontology
Output: collection of all MinAs for $A \sqsubseteq_{\mathcal{O}} B$
1: $\mathcal{O}_A^{\mathsf{reach}}$ := extract-module($\mathcal{O}, \{A\}$)
2: return hst-extract-all-minas($A, B, \mathcal{O}_A^{\mathsf{reach}}$)
```

Though lin-extract-mina performs slightly better in most cases w.r.t. Snomed, log-extract-mina outperforms it by several orders of magnitude w.r.t. FullGalen. For the sake of robustness, we adopt the logarithmic algorithm as the default minimization method in our MOB approach.

In [BS08], we proposed an efficient method for checking uniqueness of the first MinA $\mathcal{S}_1$ by using reachability-based modules. For this method to be complete, we need a strong subsumption module in the sense of Definition 23 on page 31. Fortunately, the reachability-based module $\mathcal{O}_A^{\mathsf{reach}}$ is a strong subsumption module (see Theorem 43 on page 89). To this end, it suffices to test whether or not $A \sqsubseteq_{\mathcal{O}_A^{\mathsf{reach}} \setminus \{\alpha\}} B$ for each axiom $\alpha \in \mathcal{S}_1$. Note that the same idea does not apply to the GOB approach since an nMinA is not a strong subsumption module.

Naturally, we exploit the strongness property of $\mathcal{O}_A^{\mathsf{reach}}$ to drastically reduce the search space the hitting set tree (HST) algorithm (see Algorithm 9 on page 101) and the pruning algorithm need to traverse when computing all MinAs for $A \sqsubseteq_{\mathcal{O}} B$. This full pinpointing algorithm is outlined in mob-hst-alg of Algorithm 11.

To demonstrate the practicability of axiom pinpointing on large-scale ontologies like Snomed ct, the three combined methods (i.e., gob-alg in Algorithm 10, and mob-alg and mob-hst-alg in Algorithm 11) have been implemented in the CEL system. Moreover, the goal-directed subsumption algorithm (Algorithm 2 on page 64) implemented also in CEL is used as the black-box subsumption reasoner while pruning axioms (either by Algorithm 7 on page 97 or Algorithm 8 on page 98). Various experimental results of these three methods on Snomed ct are discussed in Subsection 6.2.5.

Chapter 6

Empirical Evaluation

This chapter is dedicated to evaluation of the various techniques (both for standard and supplemental reasoning) developed in the present dissertation. The algorithms described in the previous two chapters have been implemented in the CEL system. The system description, as well as the reference manual and a few examples, can be found in [Sun05a]. In the first section of this chapter, we introduce several realistic biomedical ontologies that are used to benchmark CEL and other state-of-the-art DL reasoners. Experiments and their empirical results are presented and discussed in Section 6.2.

6.1 Ontology Test Suite

Surprisingly many realistic ontologies are formulated using the expressivity of $\mathcal{EL}$ and extensions thereof—most of which are from the life science domain. To evaluate our reasoning algorithms and demonstrate their scalability, we have selected a number of large-scale biomedical ontologies to be used in our experiments. In the following, we introduce each of these ontologies by providing a short description and Description Logic properties, such as the number of axioms.

The Systematized Nomenclature of Medicine, Clinical Terms (SNOMED) is a comprehensive medical and clinical ontology. In Chapter 3, a case study of $\mathcal{EL}^+$ provides an insight into this ontology. It is an unfoldable $\mathcal{ELH}$ TBox (see Definition 3 on page 20) augmented with two complex role inclusion axioms.[1]. As of release *January/2005*, SNOMED consists of 340 972 primitive and 38 719 full concept definitions, 11 role hierarchy axioms, and a role inclusion axiom; and refers to 379 691 concept and 62 role names.

The Thesaurus of the US National Cancer Institute (NCI) is a large and carefully designed ontology that has become a reference terminology covering areas of basic and clinical science. The knowledge represented in NCI includes the domains of diseases, drugs, anatomy, genes, gene products, techniques and biological processes, all with a cancer-centric focus in its content. It was originally

[1]To represent the right identity rule in the presence of roleGroup (see Section 3.1 for the details)

designed to support coding activities across the National Cancer Institute and to facilitate translational research in cancer.

In our experiments, we considered the OWL version[2] that is formulated precisely as an unfoldable $\mathcal{EL}$ TBox augmented with domain and range restrictions. It primitively defines 27 652 concept names and refers to 70 role names, each of which is constrained by a pair of domain and range restrictions.

The Galen Medical Knowledge Base (Galen) has been developed within an EU project that sought to produce a reference ontology in a specialized DL (called GRAIL) for use in developing and managing other terminologies and indexing knowledge required for decision support, user interfaces and other knowledge management tasks [Rec07].

The full version of this ontology contains 23 136 concept and 950 role names.[3] It is precisely based on $\mathcal{SHIF}$ dispensed with negation, disjunction and value restriction. The DL $\mathcal{EL}^+$ however can express most of its axioms, namely 95.75%, and we obtained this fragment (henceforth, $\mathcal{O}^{\text{FullGalen}}$) for experimental purposes by dropping role inverse and functionality axioms. The resulting ontology can still be considered realistic and identical to the original one apart from a small number of missing subsumption relationships involving the removed role axioms.

Since the full version is both large and complex to be handled by DL reasoners, a simplified version of Galen has often been considered as a decent benchmark ontology for testing DL reasoners. This version[4] has originally been produced by Horrocks to evaluate FaCT and KRIS in his PhD thesis [Hor97]. It consists of 2 748 concept and 413 role names. Again, we considered its $\mathcal{EL}^+$ fragment, denoted by $\mathcal{O}^{\text{NotGalen}}$, by dropping role inverse and functionality axioms.

The Gene Ontology (Go) project is a collaborative effort to address the need for consistent descriptions of gene products in different databases. It has developed and is maintaining three controlled vocabularies (i.e., ontologies) that describe gene products in terms of their associated *biological processes*, *cellular components* and *molecular functions* in a species-independent manner.

The ontology, denoted by $\mathcal{O}^{\text{Go}}$, is formulated as an unfoldable $\mathcal{EL}$ TBox with a single transitive role part-of. The release of $\mathcal{O}^{\text{Go}}$ used in our experiments consists of 20 465 concept names and the same number of primitive concept definitions, one for each concept name. The facts that $\mathcal{O}^{\text{Go}}$ is a primitive TBox and that it contains no cyclic dependencies make it relatively easy to classify by most DL reasoners.

The Foundational Model of Anatomy (Fma) is an evolving ontology concerned with the formal representation of human anatomy. Its ontological framework can

[2]Downloadable at `http://www.mindswap.iorg/2003/CancerOntology`
[3]Downloadable at `http://www.co-ode.org/galen`
[4]Downloadable at `http://www.cs.man.ac.uk/~horrocks/OWL/Ontologies/galen.owl`

Ontologies	♯Concepts	♯Roles	♯Axioms						Cycles
	$\vert\mathsf{CN}(\mathcal{O})\vert$	$\vert\mathsf{RN}(\mathcal{O})\vert$	PCDef	CDef	GCI	RH	RI	DR	
$\mathcal{O}^{\text{Go}}$	20 465	1	19 465	0	0	0	1	0	no
$\mathcal{O}^{\text{Nci}}$	27 652	70	27 635	0	0	0	0	140	no
$\mathcal{O}^{\text{Fma}}$	75 139	2	75 139	0	0	0	2	0	yes
$\mathcal{O}^{\text{NotGalen}}$	2 748	413	2 030	695	408	416	26	0	yes
$\mathcal{O}^{\text{FullGalen}}$	23 136	950	13 149	9 968	1 951	958	58	0	yes
$\mathcal{O}^{\text{Snomed}}$	379 691	62	340 972	38 719	0	11	2	0	no

Table 6.1: The test suite of realistic biomedical ontologies.

be applied and extended to other species.[5] Fma has four interrelated components: (i) the *anatomy taxonomy* classifies anatomical entities according to their generalization relationships; (ii) the *anatomical structural abstraction* specifies the part–whole and spatial relationships among entities in the anatomy taxonomy; (iii) the *anatomical transformation abstraction* specifies the morphological transformation of anatomical entities; and (iv) the *metaknowledge* ontology specifies the principles and rules that govern the way concepts and roles in the other three components are represented.

The ontology, denoted by $\mathcal{O}^{\text{Fma}}$, is indeed a large $\mathcal{EL}$ TBox extended with two transitivity axioms, one for part-of and the other for has-part. The number of concept names is 75 139. Concepts in $\mathcal{O}^{\text{Fma}}$ are only primitively defined like most anatomical entities in any medical ontologies, yet connectivity of concepts and cyclic dependencies make this ontology rather complex.

Table 6.1 summarizes the size and other pertinent characteristics of all the test-suite ontologies. Numbers of axioms are broken down into the following kinds: primitive concept definitions (PCDef), full concept definitions (CDef), general concept inclusions (GCI), role hierarchy axioms (RH), role inclusions (RI) and domain and range restrictions (DR). Note that $\mathcal{O}^{\text{Nci}}$ constrains each role name with a pair of domain and range restrictions, and that domain restrictions could be regarded as GCIs in $\mathcal{EL}^+$. For $\mathcal{O}^{\text{NotGalen}}$ and $\mathcal{O}^{\text{FullGalen}}$, cyclic dependencies already exist when taking into account only concept definitions. The role inclusions in all ontologies except $\mathcal{O}^{\text{Snomed}}$ are indeed transitivity axioms. The two role inclusions in $\mathcal{O}^{\text{Snomed}}$ represent the right-identity rule in the presence of roleGroup (see Subsection 3.1 for the details).

It is worthwhile to mention that several other biomedical ontologies are—at the time of writing—readily available in OWL[6] thanks to a systematic mapping methodology and tool provided in [GHH$^+$07]. In fact, most life science ontologies obtained from the *Open Biomedical Ontologies (OBO)*[7] repository turn out to be expressible in $\mathcal{EL}^+$. These were not included in our experiments mostly because they are superseded by aforementioned ontologies both in terms of scale and of expressivity.

[5] Available at `http://sig.biostr.washington.edu/projects/fm/AboutFM.html`
[6] Downloadable at `http://owl.cs.manchester.ac.uk/repository/`
[7] Available at `http://www.bioontology.org/repositories.html`

6.2 Testing Methodology and Empirical Results

The majority of the reasoning techniques presented from Chapter 4 to 5 have been implemented in the CEL reasoner.[8] The current version of CEL is written in Common Lisp and compiled and built using Allegro Common Lisp.

Like most evaluation methods for DL and other reasoning systems, all the experiments described in this section use 'CPU time' as the main performance indicator. Memory consumption is also discussed whenever appropriate. In order to confine the execution environment and hence to induce sensible comparison, the experiments were performed on the same Linux testing server sitting in a temperature-controlled room. The server was equipped with a couple of 2.19GHz AMD Opteron processors and 2 GB of physical memory.

In the following, we describe our testing methodology and the empirical results of each of the following reasoning algorithms implemented in the CEL reasoner:

- classification,
- incremental classification,
- subsumption query answering,
- modularization, and
- axiom pinpointing.

6.2.1 Classification

In order to evaluate the performance of CEL, the realistic ontologies described in the previous section were preprocessed, classified and 'taxonomized.' By preprocessing, CEL loads the ontology from file into the system, then normalizes and prepares it for further operations (see Subsection 4.1.1 and 4.1.2). Classification is done by an invocation to the 'refined classification algorithm' presented in Subsection 4.1.4 which yields completed sets of subsumers. This information suffices to correctly answer subsumption between two concept names but, unfortunately, not to output the concept hierarchy or concept DAG. Algorithm 3 on page 70 is used to compute the concept DAG from the completed set of subsumers.

These three steps were performed on all of the six benchmark ontologies. Computation time results are shown in the middle section of Table 6.2, whereas the upper part present the percentage of positive subsumption relationships. Observe that there are fewer than 1% of all pairs of concept names in all ontologies, for which the subsumption holds. Evidently, most of the time was spent on classification, while the computation time for the concept DAG was relatively minuscule given that the ontology had been classified. In fact, it took less time to compute the DAG than to preprocess the ontology. The first ($\mathsf{CEL}^\dagger$) and second ($\mathsf{CEL}^\ddagger$) rows of the lower part of Table 6.2 list the overall computation time, respectively, where the optimization

[8]The name stands for "Polynomial-time **C**lassifier for the Description Logic $\mathcal{EL}^+$," and the reasoner is available at `http://lat.inf.tu-dresden.de/systems/cel/`.

Ontologies	$\mathcal{O}^{\text{Go}}$	$\mathcal{O}^{\text{Nci}}$	$\mathcal{O}^{\text{Fma}}$	$\mathcal{O}^{\text{NotGalen}}$	$\mathcal{O}^{\text{FullGalen}}$	$\mathcal{O}^{\text{Snomed}}$
Positive subs. (%)	0.0482	0.0441	0.0184	0.6021	0.1648	0.0074
Preprocessing	0.46	1.09	5.39	0.28	2.54	60.56
Classification	0.95	2.28	3 920.23	2.51	197.99	1 188.85
Computing DAG	0.26	0.38	1.81	0.04	0.71	8.85
CEL†	1.67	3.75	3 927.43	2.83	201.24	1 258.26
CEL‡	0.98	3.75	9.04	2.83	201.24	1 258.26
FaCT^{++}	20.12	1.72	*time-out*	3.28	*mem-out*	605.57
HermiT	16.75	34.92	123.32	12.35	*mem-out*	*mem-out*
KAON2	*mem-out*	*mem-out*	*time-out*	*mem-out*	*mem-out*	*time-out*
Pellet	52.58	36.11	7 753.57	31.56	*mem-out*	*mem-out*
RacerPro	17.11	13.36	629.72	17.06	*time-out*	1 155.43

Table 6.2: Computation times (second).

"disabling $R(\cdot)$ for primitive TBoxes" (see Subsection 4.1.5) was disabled and enabled. Observe that this optimization is effective on $\mathcal{O}^{\text{Go}}$ and $\mathcal{O}^{\text{Fma}}$ which comprises only primitive concept definitions. The standard version of CEL, with the optimization disabled, took 1.67 and 3 927 seconds to process $\mathcal{O}^{\text{Go}}$ and $\mathcal{O}^{\text{Fma}}$, respectively, while it merely took 0.98 and 9.04 seconds, respectively, when the optimization was enabled. The presence of domain restrictions, full concept definitions and GCIs prevent the other ontologies to benefit from this optimization, hence equal computation time by both versions.

The empirical results have confirmed the algorithmic conjecture that the classification algorithm performs well on loosely connected ontologies like most in our test suite. For tightly connected ontologies, the algorithm has to construct a dense graph, possibly with large node labels ($S(\cdot)$) or a large number of edges ($R(\cdot)$) or both. The larger this graph structure becomes, the more the classification algorithm deteriorates. A notable example is the classification (CEL†) of $\mathcal{O}^{\text{Fma}}$ which took more than twice the classification time of $\mathcal{O}^{\text{Snomed}}$ but generated less than one tenth of the subsumption relationships in $\mathcal{O}^{\text{Snomed}}$. A very large portion of time was spent on populating $R(\cdot)$, i.e., generating edges in the completion graph. The contrast in execution time in the case of $\mathcal{O}^{\text{Fma}}$ suggests that the way $R(\cdot)$ is stored and maintained could be improved.

Since classification is one of the most classical inference services, it is supported by all modern DL systems. For this reason, classification time is often used as a performance indicator for DL systems. A number of state-of-the-art DL reasoners—i.e., FaCT^{++}[9] [TH06], HermiT[10] [MSH07], KAON2[11] [Mot06], Pellet[12] [SPC$^+$07] and RacerPro[13] [HM01b]—were considered for performance comparison. These DL reasoners vary in the sense that they implement different reasoning calculi and are written

[9]Version 1.2.0 available at `http://code.google.com/p/factplusplus/`

[10]Latest version available at `http://web.comlab.ox.ac.uk/people/Boris.Motik/HermiT/`

[11]Latest version available at `http://kaon2.semanticweb.org/`

[12]Version 1.5.2 available at `http://pellet.owldl.com/`

[13]Version 1.9.2 available at `http://www.racer-systems.com/`

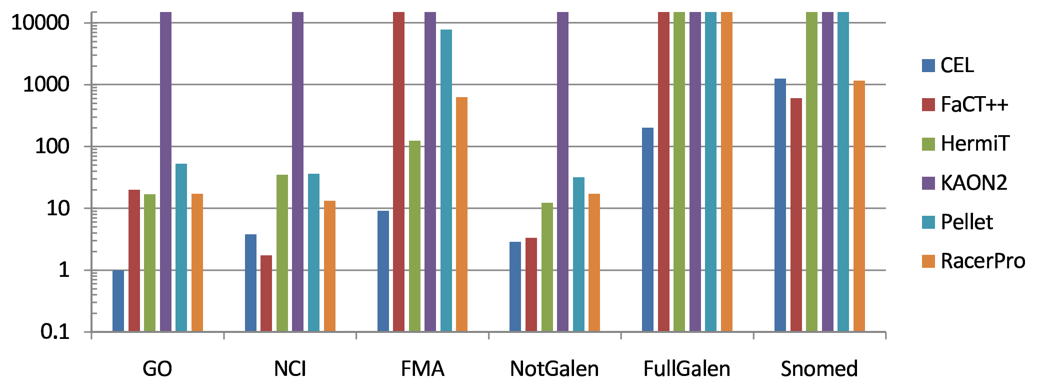


Figure 6.1: Performance comparison through classification time (second).

in different languages. For HermiT, KAON2 and Pellet, Sun's Java Runtime Environment (JRE) version 1.6.0 was used with alloted 1.5GB heap space. Some reasoners are not equipped with a profiling facility to internally measure CPU time. To achieve comparable measurement, an external timing utility was used with all the classifying systems.[14]

All ontologies in the test suite as described in the previous section were used as benchmarks for comparing the performance of the DL reasoners. Since KAON2's parser, and hence HermiT's parser, does not support (an extension of) the KRSS syntax [PSS93], ontologies in the OWL format were used in their experiments. In the case of $\mathcal{O}^{\textsc{Snomed}}$, the two complex role inclusions were only passed to CEL and FaCT^{++} but not to the other reasoners, as the latter do not support such axioms. Additionally, we needed to rename all the roles, because SNOMED uses the same codes for both roles and 'attributive' concepts but KAON2 and HermiT do not support such name punning. It has to be noted however that such renaming could by no means affect the meaning nor the classification results of the ontology. The rest of the table shows the (two-run average) time taken by the respective reasoners to classify the biomedical ontologies, where *mem-out* means that the reasoner failed due to memory exhaustion, and *time-out* means that the reasoner did not terminate within the allocated time of 24 hours. Figure 6.1 depicts a comparison chart of reasoners' performance based on their classification time, where both *mem-out* and *time-out* are displayed as full vertical bars.

It can be seen from the chart and the table that CEL is the only DL reasoner that can classify all six biomedical ontologies in the test suite and outperforms HermiT, KAON2 and Pellet in all cases. Compared with the other reasoners, CEL is faster than RacerPro w.r.t. all ontologies but $\mathcal{O}^{\textsc{Snomed}}$, and faster than FaCT^{++} w.r.t. all ontologies but $\mathcal{O}^{\textsc{Nci}}$ and $\mathcal{O}^{\textsc{Snomed}}$. It should be noted that, when it first came into existence in 2005 [BLS05], CEL was the only academic DL system that was capable of classifying

[14]The shell time command: `$> time --verbose -- ReasonerCommand [ReasonerOptions...]`

entire Snomed ct. This has subsequently sparked interest in the DL community to research on optimization techniques specific to the biomedical ontologies (in particular, to Snomed ct), and later enabled tableau-based reasoners like FaCT^{++} and RacerPro to take advantage of simple structures of ontologies of this kind. Some of the most effective optimizations employed in these systems are described in [HT05, HMW08]. For instance, the so-called 'completely defined' (CD) optimization [HT05] helps to avoid the expensive top-search and bottom-search operations (see Subsection 4.3.1) for those concept names A with a primitive definition like:

$$A \quad \sqsubseteq \quad B_1 \sqcap \cdots \sqcap B_n$$

with B_i concept names. This optimization is highly effective on $\mathcal{O}^{\text{Snomed}}$ since it contains many such definitions.[15] When a large number of GCIs are present as in the case of $\mathcal{O}^{\text{FullGalen}}$, however, these reasoners fail due to either memory exhaustion or time out. Interestingly, CEL is the only reasoner that can classify $\mathcal{O}^{\text{FullGalen}}$.

HermiT and Pellet can classify the first four ontologies but fail on the last two, both due to a memory problem. The HermiT reasoner, which implements the much less non-deterministic hypertableau calculus [MSH07], shows a relatively good performance. In fact, it noticeably outperforms Pellet in all cases and is even faster than FaCT^{++} and RacerPro on some ontologies. KAON2 cannot classify any ontologies of this scale, but it is fair to remark that this DL system has been designed to deal with and optimized for conjunctive queries w.r.t. a large number of individuals.[16]

In what follows, the testing methodology and empirical results for incremental classification, subsumption query answering, modularization and axiom pinpointing are described. In these experiments, only the CEL system was considered.

6.2.2 Incremental classification

To simulate usage scenarios of incremental classification, each ontology $\mathcal{O}$ in the test suite was partitioned into a permanent ontology—representing the well-established ontology that has previously been classified—and a set of temporary concept axioms—simulating newly authored axioms the modeler wants to add to the ontology. To this end, we have carried out 10 repetitions of the following operations for each tested ontology $\mathcal{O}$ and each number $n = 2, 4, 8, 16, 32, 64$: *(i)* partitioned $\mathcal{O}$ into $\mathcal{O}_p$ and $\mathcal{O}_t$ such that the latter consisted of n randomly selected concept axioms from $\mathcal{O}$; *(ii)* classified $\mathcal{O}_p$ normally (Algorithm 1 on page 55); and finally, *(iii)* incrementally classified $\mathcal{O}_t$ against $\mathcal{O}_p$ (Algorithm 4 on page 73). The time required to compute steps (ii) and (iii) was measured. The 20% trimmed average[17] classification and incremental classification times of the 10 repetitions are shown in Table 6.3, and the percentage of time relative to the full classification time of the whole ontology $\mathcal{O}$ is visualized

[15]A highly-optimized implementation of the $\mathcal{EL}^+$ classification algorithm exists and is claimed to classify $\mathcal{O}^{\text{Snomed}}$ within two minutes [Law08].

[16]See [BHJV08] for an evaluation of this reasoning on KAON2 among other reasoners.

[17]In order to exclude rare extreme values, two time values (i.e., the lowest and highest) were discarded, and the average was based on the eight median values.

♯Temp. axioms ($\vert\mathcal{O}_t\vert$)	$\mathcal{O}^{\text{Go}}$		$\mathcal{O}^{\text{Nci}}$		$\mathcal{O}^{\text{Fma}}$	
	C. time	IC. time	C. time	IC. time	C. time	IC. time
2	0.83	0.14	2.24	0.88	3913.33	5.06
4	0.83	0.14	2.27	0.89	3921.03	6.72
8	0.82	0.14	2.27	0.90	3920.37	26.61
16	0.82	0.15	2.27	0.91	3924.56	61.17
32	0.82	0.15	2.27	0.93	3918.16	155.51
64	0.82	0.16	2.27	1.00	3895.23	358.16
♯Temp. axioms ($\vert\mathcal{O}_t\vert$)	$\mathcal{O}^{\text{NotGalen}}$		$\mathcal{O}^{\text{FullGalen}}$		$\mathcal{O}^{\text{Snomed}}$	
	C. time	IC. time	C. time	IC. time	C. time	IC. time
2	2.28	0.60	195.05	18.48	1 182.07	21.58
4	2.29	0.63	195.08	18.73	1 184.22	21.58
8	2.26	0.66	194.85	19.25	1 184.81	21.75
16	2.22	0.73	194.82	19.96	1 184.19	22.16
32	2.18	1.01	194.77	21.33	1 183.58	22.95
64	2.04	1.32	192.06	26.24	1 183.60	22.93

Table 6.3: Incremental classification time (second).

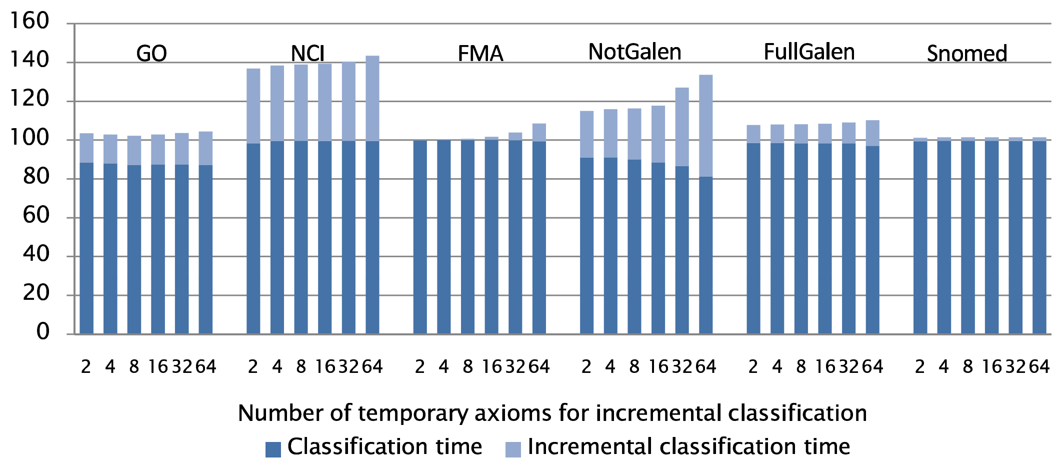


Figure 6.2: Relative (incremental) classification time w.r.t. full classification time (percentage).

in Figure 6.2. For each bar on the chart, the dark blue slice represents the *relative* classification time of $\mathcal{O}_p$, while the light blue slice represents the *relative* incremental classification time of $\mathcal{O}_t$ against $\mathcal{O}_p$. Hence, the entire bar depicts the overall computation time.

Classification of $\mathcal{O}_p$ took less time than the entire ontology $\mathcal{O}$ (except for little noise in the case of $\mathcal{O}^{\text{Fma}}$), since the former is a subset of the latter. The time required to incrementally classify $\mathcal{O}_t$ varied, depending on the ontology and according to the number of new axioms. As a rule, the larger $\mathcal{O}_t$ is, the more time it took to incrementally classify it and also the more time it took overall to classify $\mathcal{O}_p$ and incrementally classify $\mathcal{O}_t$. The overall computation time (i.e., the height of each bar) was less than 150% for all ontologies and all numbers n. In the case of $\mathcal{O}^{\textsc{Go}}$, $\mathcal{O}^{\textsc{Fma}}$, $\mathcal{O}^{\textsc{FullGalen}}$ and $\mathcal{O}^{\textsc{Snomed}}$, at most only 10% additional time was needed in order to incrementally classify up to 64 additional axioms. The proportion of incremental classification time is larger for $\mathcal{O}^{\textsc{NotGalen}}$ than other ontologies since it is relatively much smaller. In fact, 64 axioms already constitute more than 2% of the entire ontology. Though the size of $\mathcal{O}^{\textsc{Nci}}$ is in the same range as that of $\mathcal{O}^{\textsc{Go}}$ and $\mathcal{O}^{\textsc{FullGalen}}$, its relative incremental classification time was much greater (already at $n = 2$). This phenomenon can probably be explicated by the fact that, on average, a concept definition in $\mathcal{O}^{\textsc{Nci}}$ contains many existential restrictions. Moreover, our range elimination technique (see Subsection 4.1.2) replaces each and every GCI of the form $A \sqsubseteq \exists r.B$ by precisely three GCIs in the case of $\mathcal{O}^{\textsc{Nci}}$. As a result, a small number of original axioms in $\mathcal{O}^{\textsc{Nci}}$ could generate many normalized axioms to be processed in the incremental classification procedure.

At any rate, the duo-ontology classification algorithm (Algorithm 4 on page 73) noticeably improved on standard classification from scratch, provided that the ontology is modified only by adding axioms. As discussed in Section 4.4, limited retraction of axioms, namely *all* those axioms in $\mathcal{O}_t$, is possible in this setting by dumping the additional subsumption information obtained during the incremental classification. To retract only part of the axioms from $\mathcal{O}_t$, say $\mathcal{O}'_t \subseteq \mathcal{O}_t$, one could first retract all axioms in $\mathcal{O}_t$ and then incrementally classify $\mathcal{O}_t \backslash \mathcal{O}'_t$ against $\mathcal{O}_p$. This should still be more efficient than full classification from scratch since the dumping time of the additional subsumptions is negligible and incremental classification of $\mathcal{O}_t \backslash \mathcal{O}'_t$ against $\mathcal{O}_p$ takes much less time.

The other meaningful experiment on the incremental classification algorithm is to simulate the evolution of the large medical ontology of Snomed ct. As mentioned in Chapter 3, the present version of the ontology is the result of gradual development and expansion, as well as of merging Snomed rt with Clinical Terms version 3. To simulate this evolution process, we first took a subset of $\mathcal{O}^{\textsc{Snomed}}$ with about two hundred thousand axioms and then classified it as usual. After the initial classification was finished, we repeatedly supplied 100 additional axioms from the rest of $\mathcal{O}^{\textsc{Snomed}}$ and incrementally classified them against the previously classified axioms. This process was carried out until no more axioms were left to be incrementally classified, i.e., the entire ontology has eventually been considered.

The time required in each step of incremental classification is plotted in Figure 6.3. Each of the very first steps took as small a classification time as 6 seconds or less, and it slowly increased over the course of $\mathcal{O}^{\textsc{Snomed}}$'s expansion. The median of incremental classification time over ten seconds was only 11.46 second, the amount of time that should arguably be tolerable to be adopted in realistic development environment for Snomed ct.[18]

[18]There were two extreme time values of 58 and 211 seconds which were likely the results of rare interference from garbage collection.

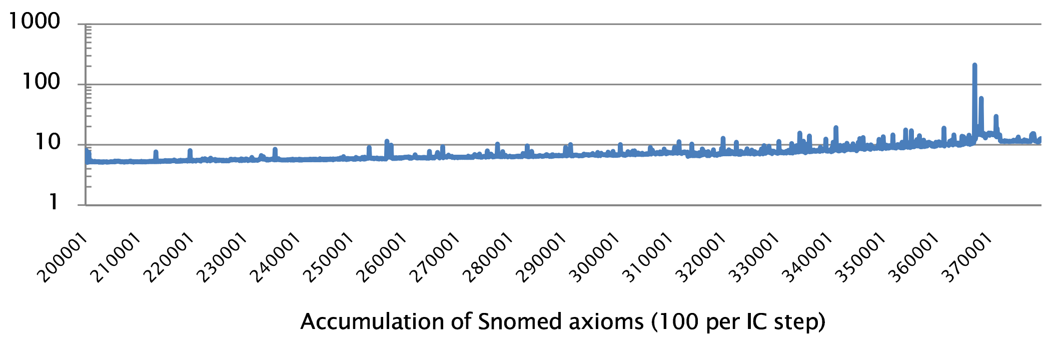


Figure 6.3: Incremental classification time (second) for $\mathcal{O}^{\text{Snomed}}$'s evolution.

Ontology	$\mathcal{O}^{\text{Go}}$	$\mathcal{O}^{\text{Nci}}$	$\mathcal{O}^{\text{Fma}}$	$\mathcal{O}^{\text{NotGalen}}$	$\mathcal{O}^{\text{FullGalen}}$	$\mathcal{O}^{\text{Snomed}}$
p_subs($\mathcal{O}$)	0.016/10	0.062/10	0.44/10	0.29/10	75.43/7 900	0.41/20
n_subs($\mathcal{O}$)	0.048/50	0.166/10	0.48/10	0.97/20	3 477.35/8 050	1.01/1 040

Table 6.4: Average/maximum subsumption testing time (*milli*second).

6.2.3 Subsumption query answering

This subsection describes the experiments and their results of subsumption testing (Algorithm 2 on page 64) in $\mathcal{EL}^+$ w.r.t. the tested biomedical ontologies. To evaluate the goal-directed subsumption algorithm, we have sampled[19] two sets of subsumptions as follows: *(i)* randomly select 1 000 concept names from $\mathsf{CN}(\mathcal{O})$; *(ii)* for each A from step (i), sample 5 distinct *positive* subsumptions $\mathcal{O} \models A \sqsubseteq B$ with $B \notin \{A, \top\}$; *(iii)* for each A from step (i), sample 5 distinct *negative* subsumptions $\mathcal{O} \not\models A \sqsubseteq B$ for some B. We denote by p_subs($\mathcal{O}$) and n_subs($\mathcal{O}$) the sets of sampled subsumptions obtained by steps (ii) and (iii) from the ontology $\mathcal{O}$, respectively.

The goal-directed subsumption algorithm (Algorithm 2 on page 64) without any heuristics and caching has been run for each subsumption in p_subs($\mathcal{O}$) and n_subs($\mathcal{O}$). The average/maximum CPU time for each subsumption test in each ontology is shown in Table 6.4. Observe that, on average, querying single subsumptions using Algorithm 2 took tiny fractions of a second in most cases except for the negative subsumptions in $\mathcal{O}^{\text{FullGalen}}$. The hardest case for the ontology took just above eight seconds in order to decide non-subsumption. These experiments empirically confirmed the conjecture concerning the algorithm's behavior that it should run faster on positive subsumption since it immediately terminates once the goal subsumption is identified but has to continue to finish processing all the queue entries otherwise. Except for $\mathcal{O}^{\text{Fma}}$, the time differences between querying positive subsumptions, i.e., p_subs($\mathcal{O}$), and negative ones, i.e., n_subs($\mathcal{O}$), are more than double.

[19]Since there are about *144 billion pairs* of concept names in the case of $\mathcal{O}^{\text{Snomed}}$ and some subsumption queries against $\mathcal{O}^{\text{FullGalen}}$ took several seconds, performing subsumption queries between *all* pairs would not be feasible; hence, the need for sampling.

Ontologies	Extraction time				Recursive batch
	median	average	maximum	total	extraction time
$\mathcal{O}^{\text{Go}}$	$\sim$0.00	0.0001	0.01	1.41	0.52
$\mathcal{O}^{\text{Nci}}$	$\sim$0.00	0.0001	0.19	2.19	0.94
$\mathcal{O}^{\text{Fma}}$	0.10	0.0688	1.17	5 171	*mem-out*
$\mathcal{O}^{\text{NotGalen}}$	$\sim$0.00	0.0005	0.03	1.42	0.48
$\mathcal{O}^{\text{FullGalen}}$	0.01	0.0317	0.92	734	*mem-out*
$\mathcal{O}^{\text{Snomed}}$	$\sim$0.00	0.0082	5.46	3 110	*mem-out*

Table 6.5: Time to extract the reachability-based modules (second).

It is interesting to give a remark on the relationship between the subsumption querying time and the size of the reachability-based module as suggested by Theorem 42 on page 86. The theorem states that the goal-directed subsumption algorithm running on $A \sqsubseteq B$ w.r.t. $\mathcal{O}$ only requires axioms from the module $\mathcal{O}_A^{\mathsf{reach}}$. In the next subsection, experimental results on the reachability-based modularization are discussed. In particular, it will be seen that subsumption querying time is roughly proportional to module sizes (see Table 6.4 above and Table 6.6 on page 124). The only exception is $\mathcal{O}^{\text{Fma}}$, of which subsumption querying time could drastically be minimized by the optimization "disabling $R(\cdot)$ for primitive TBoxes" in spite of its large modules.

6.2.4 Modularization

Two sets of experiments were carried out to evaluate the modularization based on reachability. The first set followed the experiments described in [CHKS07], where a module for each concept name in each ontology was extracted (henceforth, referred to as *c-module* for brevity). The reasons were that modules for single concepts form a good indicator of the typical size of the modules compared to the whole ontology. Moreover, modules for single concepts are especially interesting for optimization both in standard reasoning of classification [CHKS07, CHWK07] and in axiom pinpointing [BS08, SQJH08]. Section 6.2.5 reports on the practical effectiveness of using reachability-based modules in optimizing axiom pinpointing. The second set of experiments concerned non-atomic signatures of varying sizes.

For each ontology $\mathcal{O}$ in the test suite and each concept name A occurring in $\mathcal{O}$, we have extracted the reachability-based module $\mathcal{O}_A^{\mathsf{reach}}$ by using Algorithm 6 on page 84. The time required to extract each c-module and its size were measured and are summarized in Table 6.5 and 6.6, respectively. Observe that it took only a tiny amount of time to extract a c-module based on reachability, where more than two third of all the extractions required less than 10 milliseconds (shown as ~ 0.00 in the table). However, extracting a large number of c-modules (i.e., as many as the number of concept names) required considerably more time and even longer than classification in some cases. This was nevertheless the result of multiple individual extractions that are independent of each other and as such did not exploit the recursive nature of reachability. In fact, Point 3 of Proposition 38 could be used to recursively extract c-modules according to the reverse reachability order. Precisely, if A is B-

Ontologies	Module size (%)		
	median	average	maximum
$\mathcal{O}^{\textsc{Go}}$	19 (0.0928)	28.42 (0.1389)	190 (0.9284)
$\mathcal{O}^{\textsc{Nci}}$	12 (0.0434)	28.97 (0.1048)	436 (1.577)
$\mathcal{O}^{\textsc{Fma}}$	22 234 (29.59)	14 881.13 (19.80)	22 276 (29.65)
$\mathcal{O}^{\textsc{NotGalen}}$	33 (1.201)	61.82 (2.250)	435 (15.83)
$\mathcal{O}^{\textsc{FullGalen}}$	167 (0.7218)	3 795.13 (16.40)	8 553 (36.97)
$\mathcal{O}^{\textsc{Snomed}}$	19 (0.0050)	30.99 (0.0082)	262 (0.0690)

Table 6.6: Size of the reachability-based modules (♯axioms and percentage).

reachable, then $\mathcal{O}^{\mathsf{reach}}_A \subseteq \mathcal{O}^{\mathsf{reach}}_B$. Therefore, by extracting the smaller module $\mathcal{O}^{\mathsf{reach}}_A$ prior to the larger one $\mathcal{O}^{\mathsf{reach}}_B$, the extraction of the latter can exploit the information made available from previous extractions. This idea was implemented and evaluated against the test ontologies. Unlike Algorithm 6, it is not possible to measure individual extraction time since the recursive algorithm extracts all c-modules at once.

The batch extraction time results are shown in the right column of Table 6.5, where *mem-out* means that the algorithm failed due to memory exhaustion. As expected, the recursion optimization helped to speed up extraction of c-modules, provided that no memory issue was incurred. Considering $\mathcal{O}^{\textsc{Go}}$, $\mathcal{O}^{\textsc{Nci}}$ and $\mathcal{O}^{\textsc{NotGalen}}$, the recursive algorithm required less than half of the CPU time required by Algorithm 6. In the other three cases, however, recursive batch extraction was unattainable because a huge amount of memory was required in order to store the extracted c-modules and their signatures. Though there are less c-modules in $\mathcal{O}^{\textsc{FullGalen}}$ than in $\mathcal{O}^{\textsc{Nci}}$, many c-modules in the former are of size at least an order of magnitude larger than those in the latter. In contrast, $\mathcal{O}^{\textsc{Snomed}}$ has only small c-modules, but its sheer number of concept names (thus, c-modules) makes it also space-demanding for the recursive algorithm. The same thing can be said for $\mathcal{O}^{\textsc{Fma}}$, as it combines both the issue of a large number of c-modules and that of large c-modules.

Statistical data concerning the size of c-modules are summarized in Table 6.6, where values in parentheses represent the percentage of the size of a c-module relative to the size of the whole ontology. Except for $\mathcal{O}^{\textsc{Fma}}$ and $\mathcal{O}^{\textsc{FullGalen}}$, all ontologies have relatively very small c-modules, i.e., in the range below 450 axioms. The exceptional ontologies have idiosyncratic structures, namely two distinct groups of c-modules, that were revealed by reachability-based modularization. In the case of $\mathcal{O}^{\textsc{FullGalen}}$, just above half of all c-modules (i.e., 12 119) are of size less than or equal to 459 axioms, while the rest (i.e., 11 017) are of size between 7 875 and 8 553 axioms. Similarly, 24 867 c-modules in $\mathcal{O}^{\textsc{Fma}}$ are of size less than or equal to 32 axioms, and the rest of size between 22 235 and 22 276 axioms. Surprisingly, there is no c-module of size between those of these two groups.

The distribution of sizes of *small* c-modules in all ontologies is depicted in Figure 6.4. For readability reasons, frequency bars for c-module size larger than 200 are trimmed off the chart. This does not affect the reading of the chart since more than 95% of the small c-modules are included and the ignored size values evenly disperse the trimmed area of the chart. The depicted distribution is natural in the sense that

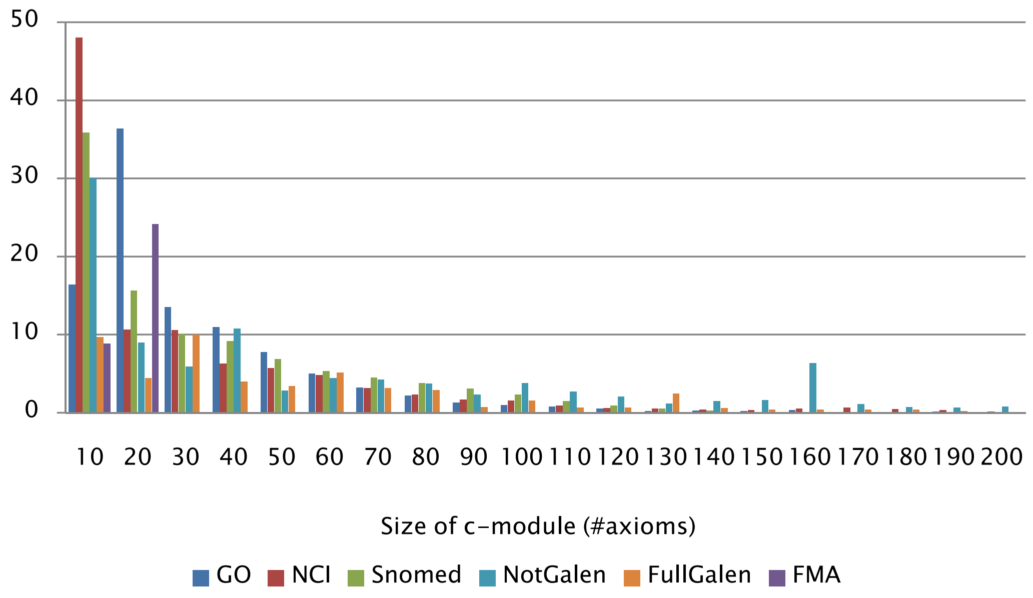


Figure 6.4: Relative frequency of *small* c-modules.

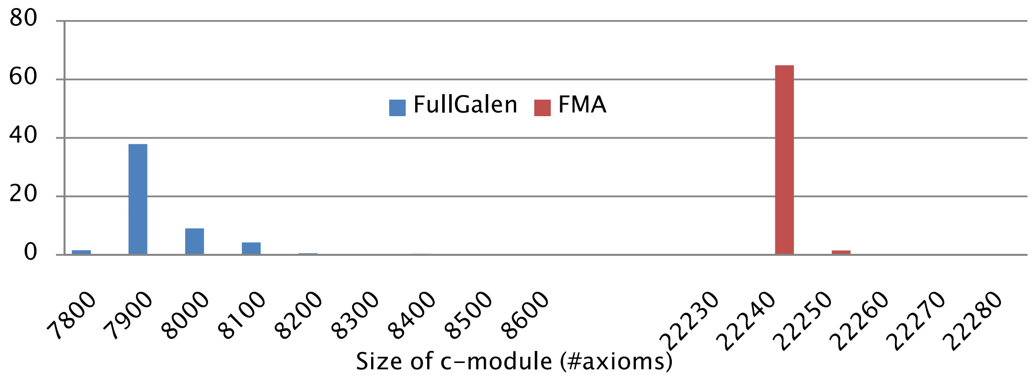


Figure 6.5: Relative frequency of *large* c-modules.

there are a large number of smaller c-modules and a small number of larger ones. This pattern is most vividly visible in the case of $\mathcal{O}^{\text{Go}}$, $\mathcal{O}^{\text{Nci}}$ and $\mathcal{O}^{\text{Snomed}}$, where about 50% of c-modules are of size 20 or less and about 95% of c-modules are of size 100 or less. A similar pattern can also be seen—to a lesser degree—in the case of the two Galen ontologies.

As pointed out earlier, there are two clusters of c-modules in $\mathcal{O}^{\text{FullGalen}}$ and $\mathcal{O}^{\text{Fma}}$ that are determined by their size. This disrupt distribution (see Figure 6.5) can be seen as an indicator of the presence of big cyclic dependencies in the ontologies. From

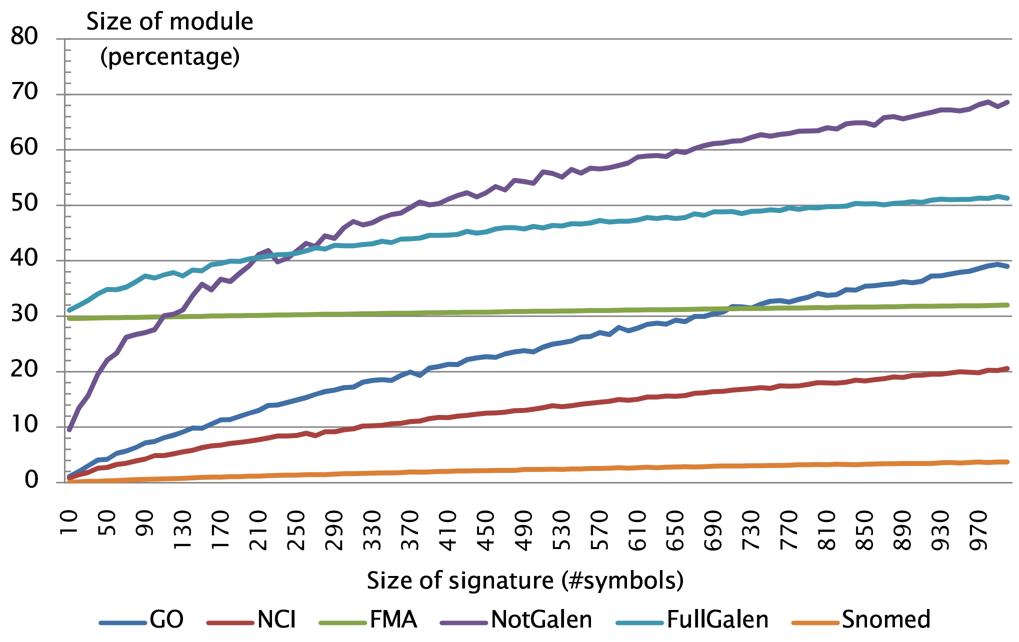


Figure 6.6: Size of the reachability-based modules against size of the signature.

the chart, there are almost 40% of c-modules in $\mathcal{O}^{\text{FullGalen}}$ of size between 7 800 and 7 900 axioms and more than 60% of c-modules in $\mathcal{O}^{\text{Fma}}$ of size between 22 230 and 22 240 axioms.[20] Though also containing cyclic dependencies, sizes of c-modules in $\mathcal{O}^{\text{NotGalen}}$ do not show such a distinctly disrupt distribution, but one could observe a local peak in the range between 150 and 160 (see Figure 6.4).

To simulate ontology reuse scenario, where a part of a well-established ontology relevant to the signature of interest is imported, we have designed and performed another set of experiments. In these experiments, signatures of varying sizes from 10 to 1 000 (at 10-symbol intervals) were randomly generated from the signature of each test ontology. For each ontology $\mathcal{O}$ and each generated signature $\mathbf{S} \subseteq \mathsf{Sig}(\mathcal{O})$, the reachability-based module $\mathcal{O}^{\mathsf{reach}}_{\mathbf{S}}$ for $\mathbf{S}$ in $\mathcal{O}$ was extracted. The size of the reachability-based module is plotted against the size of the signature in Figure 6.6. Observe that the growth trends of $\mathcal{O}^{\text{Snomed}}$, $\mathcal{O}^{\text{Nci}}$, $\mathcal{O}^{\text{Go}}$ and $\mathcal{O}^{\text{NotGalen}}$ appear proportional to the average size of c-module in the respective ontology (see Table 6.6 on page 124). The modules in $\mathcal{O}^{\text{FullGalen}}$ and $\mathcal{O}^{\text{Fma}}$ started at a relatively large size (i.e., about 30%) because there was a good chance that one of the concept names in the signature was involved in the larger cluster of c-modules. Since the reachability-based modules in $\mathcal{O}^{\text{Snomed}}$ were particularly very small, we have performed 'stress test' on it by allowing the signature to grow up to 60 000 symbols. Figure 6.7 depicts the results. Surprisingly, we needed almost 15% symbols from $\mathsf{Sig}(\mathcal{O}^{\text{Snomed}})$ to obtain as large a module

[20] A brief inspection of $\mathcal{O}^{\text{Fma}}$ has revealed that this large cycle is a result of the cyclic definitions of anatomical concepts, which are linked to each other by the roles part-of and has-part.

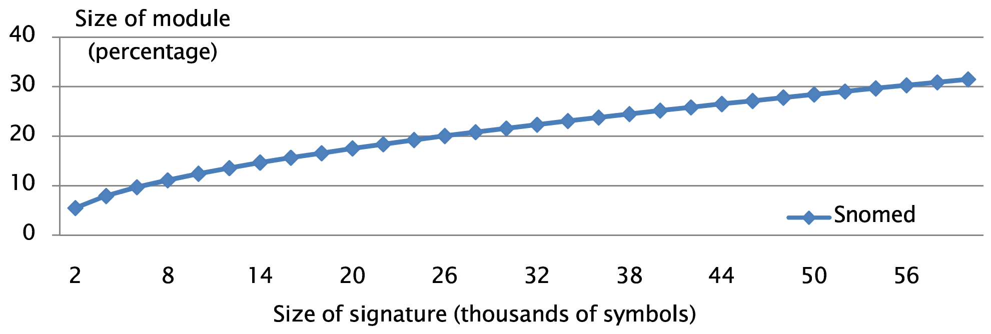


Figure 6.7: Size of the reachability-based modules in $\mathcal{O}^{\textsc{Snomed}}$.

as 30% of its axioms. This result suggests that module extraction based on reachability is quite robust, both in terms of extraction time and module size, for several of our test ontologies especially $\mathcal{O}^{\textsc{Snomed}}$.

6.2.5 Axiom pinpointing

This sections describes the methodology of testing our axiom pinpointing algorithms presented in Section 5.2 and their empirical results. Precisely, the following three algorithms:

- the Glass-box Optimized Black-box (GOB) single pinpointing algorithm (gob-alg in Algorithm 10 on page 110)
- the Modularization Optimized Black-box (MOB) single pinpointing algorithm (mob-alg in Algorithm 11 on page 111)
- the Modularization Optimized Black-box (MOB) full pinpointing algorithm (mob-hst-alg in Algorithm 11 on page 111)

have been implemented and evaluated on $\mathcal{O}^{\textsc{Snomed}}$.

Since $\mathcal{O}^{\textsc{Snomed}}$ (as well as the other ontologies in our test suite) does not contain any unsatisfiable concepts, the type of consequence of interest for axiom pinpointing in this context is subsumption. As used as our motivation earlier, the subsumption relationship (Subsumption 5.1)

$$\sigma : \mathsf{AmputationOfFinger} \quad \sqsubseteq_{\mathcal{O}^{\textsc{Snomed}}} \quad \mathsf{AmputationOfHand}$$

holds in the considered version of $\mathcal{O}^{\textsc{Snomed}}$. An attempt to run the naïve pruning algorithm (Algorithm 7 on page 97) failed; it did not terminate on this input after 24 hours. The logarithmic extraction (Algorithm 8 on page 98) was successful on σ and required 1 565 seconds to compute a MinA of cardinality 6. By using (a simplified version of) mob-hst-alg, it can be verified that this is indeed the only MinA for this subsumption. The MinA comprising 6 axioms is shown in Figure 6.8. These axioms suggest that

$$
\begin{array}{rcl}
\mathsf{direct\text{-}procedure\text{-}site} & \sqsubseteq & \mathsf{procedure\text{-}site} \\
\mathsf{AmputationOfFinger} & \sqsubseteq & \mathsf{AmputationOfFingerWithoutThumb} \\
\mathsf{AmputationOfFingerWithoutThumb} & \equiv & \mathsf{HandExcision} \sqcap \\
 & & \exists \mathsf{roleGroup}.(\exists \mathsf{direct\text{-}procedure\text{-}site}.\mathsf{Finger}_S \sqcap \\
 & & \qquad \exists \mathsf{method}.\mathsf{Amputation}\ ) \\
\mathsf{AmputationOfHand} & \equiv & \mathsf{HandExcision} \sqcap \\
 & & \exists \mathsf{roleGroup}.(\exists \mathsf{procedure\text{-}site}.\mathsf{Hand}_S \sqcap \\
 & & \qquad \exists \mathsf{method}.\mathsf{Amputation}\ ) \\
\mathsf{Finger}_S & \sqsubseteq & \mathsf{DigitOfHand}_S \sqcap \mathsf{Hand}_P \\
\mathsf{Hand}_P & \sqsubseteq & \mathsf{Hand}_S \sqcap \mathsf{UpperExtremity}_P
\end{array}
$$

Figure 6.8: The only MinA for $\mathsf{AmputationOfFinger} \sqsubseteq_{\mathcal{O}^{\textsc{Snomed}}} \mathsf{AmputationOfHand}$.

the culprit for this unintended subsumption is the incorrect use of the SEP-triplet encoding (see Subsection 3.2.1 for more details). Though Algorithm 8 improved greatly on Algorithm 7, it was still far from being satisfactory for use in realistic debugging scenarios. We could do much better by using either $\mathsf{gob\text{-}alg}$ or $\mathsf{mob\text{-}alg}$ algorithm, which employs a different technique to reduce the search space before pruning the axioms. The nMinA for σ (as produced by the labeled subsumption algorithm) contains 7 axioms, whereas the reachability-based module for $\mathsf{AmputationOfFinger}$ in $\mathcal{O}^{\textsc{Snomed}}$ contains 57 axioms. Since the nMinA and module each guarantees to cover all axioms in a MinA, a pruning technique can be used to extract a MinA from them. Due to the usually small size of the nMinAs and the reachability-based modules in $\mathcal{O}^{\textsc{Snomed}}$, the linear pruning technique proved more efficient on this specific ontology.[21] For the amputation example, both $\mathsf{gob\text{-}alg}$ and $\mathsf{mob\text{-}alg}$ algorithms took less than a second to compute exactly the same MinA.

This experiment was generalized to consider other subsumptions. However, testing the algorithms on all (positive) subsumptions was not feasible, since there are more than five million subsumption relationships that follow from $\mathcal{O}^{\textsc{Snomed}}$: assuming an average extraction time of half a second, this would have required a month. For this reason, we generated 5 (possibly overlapping) sets of 1 000 sampled concept names from $\mathsf{CN}(\mathcal{O}^{\textsc{Snomed}})$, denoted by $\mathsf{c\text{-}samples}(n)$ with $n = \{1, 2, 3, 4, 5\}$. For each n, we ran $\mathsf{gob\text{-}alg}$ (respectively, $\mathsf{mob\text{-}alg}$) on all the subsumption relationships $A \sqsubseteq_{\mathcal{O}^{\textsc{Snomed}}} B$ such that $A \in \mathsf{c\text{-}samples}(n)$, $B \notin \{A, \top\}$, and $\mathcal{O}^{\textsc{Snomed}} \models A \sqsubseteq B$. Intuitively, we considered all positive subsumption relationships between concept names that are not tautologies and whose left-hand side component belongs to $\mathsf{c\text{-}samples}(n)$. This sampling methodology is better than randomly sampling two concept names A, B such that the subsumption between them holds, since the latter may potentially avoid hard cases, e.g., a sample of told subsumptions. Our sampled subsumptions cover told as well as complex subsumptions which well represent both easy and hard cases

[21] Despite what being said, the logarithmic pruning technique has been implemented as the default subprocedure for axiom pinpointing in CEL since it is much more robust on arbitrary ontologies and not much less efficient in the case of $\mathcal{O}^{\textsc{Snomed}}$.

in $\mathcal{O}^{\text{Snomed}}$. Given $|\mathsf{c\text{-}samples}(n)| = 1\,000$ for all n, the number of subsumptions to be considered varied depending on the sampled concepts (see the second column of Table 6.7). For each subsumption considered, the time to compute an nMinA, its size, the time to prune its axioms to obtain a MinA and the MinA size were measured. The same was carried out for the mob-alg where, instead of nMinAs, reachability-based modules were considered. The average/maximum of these experimental results are listed in Table 6.7, segregated by $\mathsf{c\text{-}samples}(n)$.

Observe that the average time to compute a single MinA (i.e., the sum of the time results in columns 3 and 5) was less than a second in all samples, regardless of the algorithm. Though the extracted modules were already quite small (i.e., comprising 52 axioms on average and 165 axioms at most), the computed nMinAs were much smaller (i.e., 8 axioms on average and 57 at most). This again empirically supports Theorem 42 on page 86 since the labeled subsumption algorithm essentially behaves the same as the unlabeled one but only keeps tract of axioms it requires. The price for this smaller size however was that, in all cases, it took on average about three times longer to extract an nMinA, i.e., the labeled subsumption algorithm plus a small overhead for de-normalization mapping, than it did to extract the corresponding module. Additionally, the extraction of an nMinA has to be done every time a MinA for $A \sqsubseteq_{\mathcal{O}^{\text{Snomed}}} B$ is to be computed, unlike the module $\mathcal{O}^{\text{Snomed}}_A$ which can be reused for any subsumptions $A \sqsubseteq_{\mathcal{O}^{\text{Snomed}}} B'$. Given the fact that nMinAs are smaller than modules, it should in principle take less time to prune the former than the latter to obtain a MinA. This nevertheless was not the case in our experiments. Our conjecture is that garbage collection[22] had to be carried out more often in gob-alg than in mob-alg since it required more memory to run the labeled subsumption algorithm than to run the module extraction algorithm. Moreover, there were some overheads in communication between the two Lisp packages.[23]

Since the result from the pruning algorithm depends on the order of axioms in the input, it should not be surprising that the two approaches could yield different MinAs in the case that more than one exist. However, the average and maximum sizes of the computed MinAs by the two approaches deviate only a little as observable in the last but one column of the table. On average, a MinA in $\mathcal{O}^{\text{Snomed}}$ comprises about 6 to 7 axioms like the one for the example subsumption σ which is small enough to be inspected by the ontology developer by hand. Finally, the last column shows the ratio of the size of MinA to that of nMinA (module, respectively). It demonstrates that 84.87% of axioms in the nMinAs (13.41% in the modules) are relevant for the subsumption in question w.r.t. $\mathcal{O}^{\text{Snomed}}$. Figure 6.9 depicts the size distribution of the modules, the nMinAs and the MinAs. As easily visible from the chart, the modules are quite small, but the nMinAs and yet MinAs are even smaller. In fact, the majority

[22]Garbage collection greatly influenced time results in axiom pinpointing experiments due to the repeated use of the core reasoner as a black box which allocated and dumped the memory throughout the computation.

[23]Unlike mob-alg, the gob-alg approach requires two core reasoners: the labeled subsumption algorithm, which computes an nMinA; and the (unlabeled) subsumption algorithm, which is used as a black box to prune the nMinA. To avoid namespace problems, these have been realized by using two distinct packages.

Concept samples A from $\mathsf{CN}(\mathcal{O}^{\textsc{Snomed}})$	♯Subs. samples	Time to ext. nMinA for $A \sqsubseteq_{\mathcal{O}_A^{\textsc{Snomed}}} B$ (avg/max)	nMinA size (avg/max)	Pruning time for $A \sqsubseteq_{\text{nMinA}} B$ (avg/max)	MinA size (avg/max)	MinA/nMinA ratio (%)
c-samples(1)	14 279	0.0420 / 0.99	7.92 / 50	0.5732 / 5.26	6.68 / 37	84.27
c-samples(2)	14 209	0.0465 / 1.45	7.68 / 47	0.5701 / 5.77	6.53 / 35	85.01
c-samples(3)	14 840	0.0394 / 1.15	7.94 / 57	0.5907 / 5.60	6.76 / 38	85.17
c-samples(4)	14 617	0.0392 / 2.17	7.58 / 54	0.5717 / 6.56	6.42 / 36	84.68
c-samples(5)	14 377	0.0397 / 1.87	7.47 / 46	0.5676 / 5.57	6.37 / 33	85.20
Overall	72 322	0.0413 / 2.17	7.72 / 57	0.5748 / 6.56	6.55 / 38	84.87

Concept samples A from $\mathsf{CN}(\mathcal{O}^{\textsc{Snomed}})$	♯Subs. samples	Time to extract module $\mathcal{O}_A^{\textsc{Snomed}}$ (avg/max)	Module size (avg/max)	Pruning time for $A \sqsubseteq_{\mathcal{O}_A^{\textsc{Snomed}}} B$ (avg/max)	MinA size (avg/max)	MinA/$\mathcal{O}_A^{\textsc{Snomed}}$ ratio (%)
c-samples(1)	14 279	0.0142 / 0.05	52.69 / 161	0.3122 / 7.89	7.08 / 37	13.44
c-samples(2)	14 209	0.0135 / 0.06	52.80 / 165	0.2226 / 8.15	6.91 / 35	13.09
c-samples(3)	14 840	0.0139 / 0.05	52.08 / 145	0.2201 / 8.05	7.13 / 39	13.70
c-samples(4)	14 617	0.0139 / 0.06	51.17 / 163	0.1790 / 3.61	6.87 / 35	13.43
c-samples(5)	14 377	0.0133 / 0.06	51.23 / 158	0.1828 / 3.50	6.86 / 35	13.39
Overall	72 322	0.0138 / 0.06	51.99 / 165	0.2231 / 8.15	6.97 / 39	13.41

Table 6.7: Empirical results of the GOB and MOB approaches to axiom pinpointing on five sets of sampled subsumptions in Snomed ct (time in seconds; size in number of axioms).

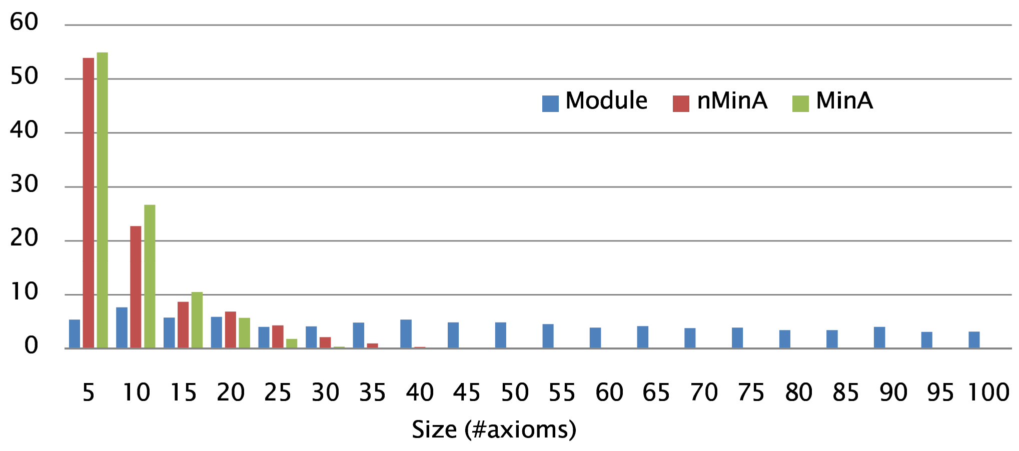


Figure 6.9: Relative frequency of the sizes of modules, nMinAs and MinAs for the sampled subsumptions in $\mathcal{O}^{\textsc{Snomed}}$.

of all subsumptions (80%) have a MinA of size ten axioms or less, and more than half (55%) have a MinA of size five or less. Though the reachability-based module for $\mathbf{S} = \{A\}$ in $\mathcal{O}$ is larger than its counterpart nMinA for $A \sqsubseteq_{\mathcal{O}} B$, they cover all axioms in all MinAs (and thus, diagnoses) for all subsumptions $A \sqsubseteq B'$ with B' any concept name in $\mathsf{CN}(\mathcal{O})$ (see Theorem 43 on page 89 and Definition 23 on page 31).

A MinA is useful in explaining logical consequences (for instance, the 'amputation' example discussed above) but not enough to suppress them. In general, a subsumption may potentially have exponentially many MinAs (see Example 47 on page 94). Removing all the axioms in *a* MinA does not guarantee that the consequence will be suppressed because axioms in *another* MinA may still remain to support it. In order to get rid of an unwanted consequence, a *MaNA* (see Definition 18 on page 29) or a *diagnosis* (see Definition 20 on page 30) needs to be computed. Since MaNAs are usually large and can easily be computed from diagnoses, we are often more interested in the computation of the latter. Precisely, given an ontology $\mathcal{O}$ and a diagnosis $\mathcal{D}$ for σ w.r.t. $\mathcal{O}$, the set of axioms $\mathcal{O} \backslash \mathcal{D}$ is a MaNA for σ w.r.t. $\mathcal{O}$. The modularization-based HST pinpointing algorithm mob-hst-alg (refer to Algorithm 11 on page 111 and Algorithm 9 on page 101) can be used to compute *all* MinAs as well as *all* diagnoses. The last experiment carried out in this thesis concerns this algorithm on $\mathcal{O}^{\textsc{Snomed}}$.

Sampled subsumptions used in the previous experiments *with more than one MinA* were considered in this experiment. There were hard cases in our sample where more than 100 MinAs existed and the constructed hitting set tree was very large in spite of the optimizations. In those hard cases, it took more than 24 hours and up to 72 hours to compute all MinAs for a subsumption. For this reason, the number of computed MinAs was limited in our experiment to 10. Therefore, the statistics shown in the following will be divided into two groups:

- easy-samples consists of sampled subsumptions that have at least two but less

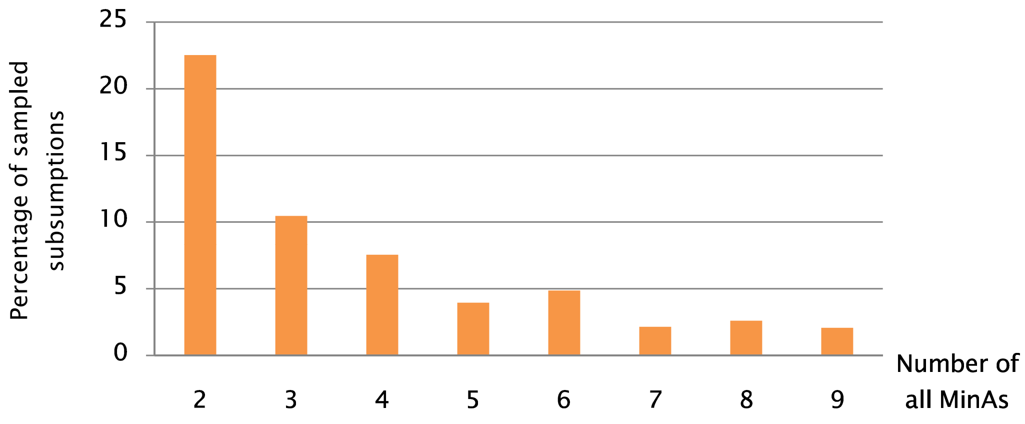


Figure 6.10: Relative frequency of the numbers of *all* MinAs for easy-samples in $\mathcal{O}^{\text{Snomed}}$.

Samples	♯MinAs (avg/max)	MinA size (avg/max)	♯Common ax. (μ)	♯All ax. (ν)	μ/ν
easy-samples	3.74 / 9	8.02 / 26	4.77 / 22	12.30 / 39	0.39
hard-samples	10 / 10	16.39 / 45	7.01 / 30	32.53 / 63	0.22

Table 6.8: Statistical results on the computed MinAs in $\mathcal{O}^{\text{Snomed}}$.

than ten MinAs; and,

- hard-samples consists of sampled subsumptions that have at least ten MinAs.

Based on all the subsumptions considered, 10 492 (56.19%) subsumptions belong to easy-samples, and 8 181 (43.81%) subsumptions to hard-samples. Table 6.8 shows the average/maximum numbers of MinAs (♯MinAs) and their size. It also presents the average/maximum numbers of common axioms in all MinAs, i.e., $\mu = |\bigcap_{\text{MinAs } \mathcal{S} \text{ for } \sigma} \mathcal{S}|$, and those of all axioms in all MinAs, i.e., $\nu = |\bigcup_{\text{MinAs } \mathcal{S} \text{ for } \sigma} \mathcal{S}|$.[24] The average ratio μ/ν, which indicates the degree of commonality of the computed MinAs, is shown in the last column of the table. The statistical results for easy-samples are complete w.r.t. all the MinAs, whereas those for hard-samples are partial. The relative distribution of ♯MinAs below ten is shown in Figure 6.10. More than half of all the considered subsumptions (51.51%) have 7 MinAs or less, i.e., the median of ♯MinAs for easy-samples and hard-samples collectively is 7. Though nothing can be said about the distribution of ♯MinAs larger than 9, it is known from the test results that about 43% have ten or more MinAs and that the largest known ♯MinAs is 158. It can be observed from the table that the MinA size is larger when there are more MinAs, i.e., a MinA

[24]Note that $\bigcup_{B \in \mathsf{CN}(\mathcal{O})} \bigcup_{\text{MinAs } \mathcal{S} \text{ for } A \sqsubseteq B} \mathcal{S}$ is the smallest strong subsumption module for A in $\mathcal{O}$ according to Definition 23 on page 31.

Samples	♯Diags (avg/max)	Diag size (avg/max)	♯Common ax. (μ)	♯All ax. (ν)	μ/ν
easy-samples	49.53 / 4 539	2.23 / 9	0 / 0	12.30 / 30	0.00
hard-samples	75.72 / 3 984	4.74 / 9	0.32 / 8	26.23 / 51	0.01

Table 6.9: Statistical results on the computed diagnoses in $\mathcal{O}^{\text{Snomed}}$.

for easy-samples is of size 8 axioms on average, whereas a MinA for hard-samples is of size 16. Due to larger sizes of MinAs and a larger number of them, ν (i.e., the number of all axioms) tends to be larger. It is not clear how μ (i.e., the number of common axioms) changes, as larger MinAs increase the chance of common axioms but a large number of MinAs decreases it. Interestingly, the degree of commonality of the axioms in all MinAs is quite high, i.e., μ/ν is 0.39 and 0.22 for the easy and hard cases, respectively. This means that about one third of axioms are shared among all the MinAs. It is sensible to compare the size of the reachability-based module (30.99 axioms; see Table 6.6 on page 124) and ν (12.30 axioms; easy-samples in Table 6.8) since the former is a strong subsumption module and ν is the number of axioms in all MinAs for a particular subsumption $A \sqsubseteq B$. Interestingly, as many as 40% of the axioms in the module participate in a MinA for a particular subsumption. It has to be noted that the module also covers all MinAs for *all* other subsumptions $A \sqsubseteq B'$ with B' a concept name in $\mathsf{CN}(\mathcal{O})$.

The same statistics were collected for diagnoses and are presented in Table 6.9. Compared to MinAs, there are an abundant number of diagnoses of which size is at most 9 axioms. Given this two properties, it is not surprising to see that diagnoses have so few axioms in common. In the case of easy-samples, no single axiom is shared among diagnoses (i.e., $\mu = 0$). Practical implication from these empirical results is twofold: *(i)* to suppress an unwanted subsumption in Snomed, it suffices to remove only a few axioms (i.e., 2.23 for easy-samples and 4.74 for hard-samples), but *(ii)* there are a large number of distinct possibilities in doing so (i.e., at most 4 539 diagnoses with no axiom in common).

Recall that the modularization-based HST pinpointing algorithm consists of three computing components: module extraction, hitting set tree search and the pruning component with a subsumption testing subprocedure. Like the single pinpointing algorithms, the linear pruning strategy was employed in these experiments on $\mathcal{O}^{\text{Snomed}}$. For each sampled subsumption, we have computed all (subject to the limit 10) MinAs and measured the time required to finish each of the computing components. Additionally, the number of subsumption calls was counted. These time results are summarized as average and maximum in Table 6.10, again divided by easy and hard sampled subsumptions. Relevant remarks on the time results are discussed in order:

- The time to extract the reachability-based module is negligible compared to the HST search time and total subsumption time. Comparing between the two samples, it took less time to extract the module in the case of easy-samples than hard-samples since the easy cases also yield smaller modules and smaller reachability-based modules are easier to be extracted. On average, the module

Samples	Time to extract module $\mathcal{O}_A^{\text{Snomed}}$ (avg/max)	HST search time excl. subs. calls (avg/max)	♯Subs. calls (avg/max)	Total subs. testing time (avg/max)
easy-samples	0.01 / 2.06	0.07 / 44.08	177.60 / 4 732	8.80 / 131.97
hard-samples	0.02 / 3.96	0.09 / 39.90	769.98 / 4 308	37.77 / 375.68

Table 6.10: Time results (second) of the modularization-based HST pinpointing algorithm on $\mathcal{O}^{\text{Snomed}}$.

size is 53 axioms for easy-samples whereas 84 for hard-samples.

- The HST search time represents the overheads required by Algorithm 9 on page 101 in order to construct the search tree. In most cases, less time was spent on this HST overheads than on actual subsumption testing. However, in extreme cases like the one that produced 158 MinAs, HST became very large with sparse MinAs found in the search tree. Since the algorithm had to recursively call itself and expand the tree, but seldom invoked the subsumption procedure to produce a next MinA, HST search time turned out to be much larger than subsumption testing time.

- Pinpointing for hard-samples required more subsumption calls for two observable reasons: first, hard subsumptions had more MinAs to be computed by pruning axioms; and second, larger modules had to be pruned in the case of hard subsumptions. Obviously, they required more invocations to the pruning procedure and each invocation required more subsumption calls. As a result, the number of overall subsumption calls averaged 178 and 770 in the case of easy-samples and hard-samples, respectively. These subsumption tests took 37.77 seconds in the case of hard-samples and only 8.80 seconds in easy-samples.

- Finally, the average time to compute all MinAs (the first 10 MinAs, respectively) using the modularization-based HST pinpointing algorithm was 8.88 seconds (37.88 seconds, respectively). Unlike the *full* glass-box approach[25], this algorithm computes the first MinA in less than half a second (see the time results of the MOB approach in Table 6.7 on page 130) and generates the next MinAs—as well as diagnoses—one after the other. This means that, while the computation is being carried out, partial outputs, i.e., some MinAs and diagnoses, are readily available for inspection by the ontology developer. An excessively long computation of all MinAs in certain hard cases can be interrupted when the developer finds the culprit(s) for the error at hand.

It is easy to see that the effectiveness of the MOB approach heavily relies on the c-module's size and the subsumption testing time. Since c-modules in $\mathcal{O}^{\text{Go}}$, $\mathcal{O}^{\text{Nci}}$ and $\mathcal{O}^{\text{NotGalen}}$ are as small as those in $\mathcal{O}^{\text{Snomed}}$ (see Table 6.6 on page 124) and it takes less time to test subsumption w.r.t. these ontologies (see Table 6.4 on page 122), similar empirical results can be expected. For $\mathcal{O}^{\text{Fma}}$ and $\mathcal{O}^{\text{FullGalen}}$, where about half

[25]This has not been considered for implementation in the scope of the present thesis.

of the c-modules are large (see Figure 6.4 on page 125), the MOB approach may not be as effective as in $\mathcal{O}^{\textsc{Snomed}}$. Nevertheless, with the largest c-modules of size about one third of the entire ontology, the MOB approach should still improve on the pure HST pinpointing algorithm (Algorithm 9 on page 101). At any rate, the logarithmic pruning strategy should be adopted whenever the c-module being considered falls into the larger clusters of $\mathcal{O}^{\textsc{Fma}}$ and $\mathcal{O}^{\textsc{FullGalen}}$.

Chapter 7

Conclusions

This thesis investigated and proposed several reasoning techniques—both standard and supplemental in the classical DL reasoning sense—that hopefully help alleviate the hassles of design and maintenance of large-scale ontologies, especially those from the life science domain. The central objective of this work was to investigate the practicability of using DL-based reasoning support in realistic biomedical ontologies in terms of reasonable response time and scalability.

To achieve this aim, we focused attention on the lightweight DL $\mathcal{EL}^+$ that, on the one hand, possesses tractable reasoning problems in contrast to other families of DLs like OWL, and on the other hand, is sufficiently expressive to formulate several biomedical ontologies. Existing and novel reasoning techniques specific to this DL have been investigated and implemented in the CEL system. In the following section, major technical and empirical results of this thesis are discussed.

7.1 Discussion of Achieved Results

The major results achieved in the context of this thesis can be categorized into three groups:

- the development of a modeling paradigm for Snomed ct;
- the design and implementation of techniques for standard reasoning—i.e., classification, incremental classification, subsumption and instance checking;
- the design and implementation of techniques for supplemental reasoning—i.e., modularization and axiom pinpointing;
- the empirical evaluation of the implemented reasoning techniques through extensive and systematic experiments on large-scale biomedical ontologies, including an overview comparison with the leading state-of-the-art DL reasoners w.r.t. ontology classification.

7.1.1 A modeling paradigm for Snomed ct

The renown medical ontology Snomed ct has been investigated thoroughly. In Chapter 3, we have identified a number of ontological and logical issues within this ontology, with emphasis on the faulty implementation of the so-called SEP encoding technique. We argued that this encoding technique encourages modeling errors (as in the 'amputation' example) and unnecessarily triples the number of concepts in the anatomy portion of the ontology. To solve the identified (logical) problems, a new modeling paradigm for Snomed ct using the DL $\mathcal{EL}^+$ has been proposed that shall do away with the hassles of modeling with SEP while still being able to rule out unintended models.

7.1.2 Techniques for standard reasoning

In Section 4.1, a refined version of the known polynomial-time classification algorithm for (a sufficiently expressive fragment of) the DL $\mathcal{EL}^{++}$ has been developed which avoids the potentially costly operation of searching for the next Completion Rule to apply. We have provided proofs of soundness and completeness of the refined classification algorithm based on the existing proofs of the abstract algorithm. Apart from the queue technique used in the refined version, we have developed a number of optimization techniques, many of which are specific to the $\mathcal{EL}$-based algorithm. To yield the concept hierarchy, the complete subsumption information in the form of subsumer sets is exploited in the simplified enhance traversal method that renders the expensive top-search phase as cheap as a simple marking algorithm and totally avoids the bottom-search phase.

A few variants of the refined classification algorithm were also proposed in the rest of Chapter 4 in order to deal with subsumption, incremental classification, instance checking and realization.

7.1.3 Techniques for supplemental reasoning

In Section 5.1, a logic-specific definition of a module based on the notion of reachability in graphs was introduced, and an algorithm for extracting it was developed. Interesting properties of reachability-based modules have been shown. For instance, the reachability-based module is equivalent to the minimal syntactic locality-based module modulo the DL $\mathcal{EL}^+$. The virtues of the module of this kind are twofold: it is inexpensive to extract since reachability is based only on the syntax of the ontology, and it is relatively small as confirmed by the empirical evidence (summarized in Subsection 7.1.4). It has also been shown that axioms required by the goal-directed subsumption algorithms always belong to the reachability-based module which suggests, to the contrary of many's intuition, that testing subsumption using the goal-directed algorithm against the module is not faster than doing so against the whole ontology.

Based on the tractable DL, the jump in complexity of axiom pinpointing has been identified, i.e., deciding a given property w.r.t. all the MinAs for a subsumption is an NP-complete problem despite tractability of the underlying logic. This jump in complexity is unobservable in expressive DLs such as OWL DL, where subsumption already requires exponential run time. It has been shown however that the computation of an arbitrary MinA can still be achieved in polynomial time, given the tractable DL. To compute a MinA (all MinAs), several techniques from the black-box and glass-box approaches have been investigated, including the binary search technique, the hitting-set tree search technique and the axiom labeling technique. Three combined algorithms were proposed and implemented, two of which compute a single MinA and the third one computes all MinAs. Promising empirical results on our approaches to axiom pinpointing are summarized in Subsection 7.1.4.

7.1.4 Empirical evaluation

Most algorithms and techniques presented in this thesis have been implemented in the CEL system which has been used to perform extensive empirical experiments, ranging from the standard reasoning of classification to the supplemental reasoning of axiom pinpointing. Results of the empirical evaluation are summarized as follows:

1. Classification is one of the most classical reasoning services and often used to benchmark DL systems. Six reasoners were benchmarked against six ontologies. CEL performs much better than most reasoners except FaCT^{++} and RacerPro. Compared with these two reasoners, CEL is faster than RacerPro w.r.t. all ontologies but $\mathcal{O}^{\textsc{Snomed}}$, and faster than FaCT^{++} w.r.t. all ontologies but $\mathcal{O}^{\textsc{Nci}}$ and $\mathcal{O}^{\textsc{Snomed}}$.

   It should be noted that, when it first came into existence in 2005, CEL was the only academic DL system that was capable of classifying entire Snomed ct. This has subsequently sparked interest in the DL community to research on optimization techniques specific to the biomedical ontologies, and later enabled tableau-based reasoners like FaCT^{++} and RacerPro to take advantage of simple structures of ontologies of this kind. When a large number of GCIs are present as in the case of $\mathcal{O}^{\textsc{FullGalen}}$, however, they failed due to memory exhaustion. Interestingly, CEL is the only reasoner among the six benchmarked DL systems that was able to classify $\mathcal{O}^{\textsc{FullGalen}}$.

2. The empirical results on incremental classification have confirmed that a large portion of time can be spared if previously inferred subsumption information is reused when small changes are applied to the ontology. Moreover, the simulation of $\mathcal{O}^{\textsc{Snomed}}$'s evolution suggests the practicality of our incremental classification algorithm in development of this ontology.

3. The goal-directed subsumption algorithm has been evaluated on 5 000 positive subsumptions and 5 000 negative ones. In general, positive subsumptions are easier to decide since the algorithm terminates immediately as soon as the subsumption is known to hold. At any rate, the average subsumption querying time is in the sub-second range in most cases (and even sub-millisecond in several cases), except the negative subsumptions in $\mathcal{O}^{\textsc{FullGalen}}$ which took about 3.5 seconds on average.

4. The reachability-based modules for single concept names are cheap to compute (i.e., in the sub-second range) and are very small in most cases. $\mathcal{O}^{\textsc{Snomed}}$ has the smallest modules relative to the ontology size (with the largest having only 0.069% of all axioms), while each of $\mathcal{O}^{\textsc{Fma}}$ and $\mathcal{O}^{\textsc{FullGalen}}$ contains a cluster of larger modules (with the largest reachability-based modules having about one third of their axioms).

   Considering varying sizes of signatures, the growth rate of the reachability-based modules is fairly slow. This suggests its practicality in the ontology re-use scenario.

5. The three pinpointing algorithms have been evaluated on reasonable samples of subsumptions in $\mathcal{O}^{\textsc{Snomed}}$. For single pinpointing algorithms, it took less than a second by both MOB and GOB approaches to compute a MinA of size about 6–7 axioms on average. For the GOB approach, the nMinAs computed by the labeled subsumption algorithm are almost always minimal, in which about one axiom is redundant.

   Finding all MinAs is much harder but still practical with our modularization-based HST pinpointing algorithm that exploits modules to drastically reduce the search space while pruning axioms. The experiments show that there are usually a small number of MinAs (i.e., about 52% of multi-MinA subsumptions have 7 MinAs or fewer) but rare cases with more than 100 MinAs do exist. The HST algorithm simultaneously computes diagnoses (hence, MaNAs) for the subsumption. In $\mathcal{O}^{\textsc{Snomed}}$, there are typically an abundant number of fairly small diagnoses. It follows that, to suppress the subsumption in question, it suffices to remove a few axioms in a diagnosis, but there are many possibilities in doing so.

7.2 Directions for Future Work

Several directions for further research on reasoning techniques based on $\mathcal{EL}^+$ and future development of the CEL system are in order:

- Investigating new optimization techniques and improving the existing ones. Classification on $\mathcal{O}^{\textsc{Fma}}$ has pointed out the problem with a tightly connected ontology that overloads the expected population of $R(\cdot)$, many elements of which are transitive roles made explicit in the completion graph. It is thus worthwhile to investigate whether an alternative treatment of transitive roles will improve the performance.

- Extending the classification algorithm and the queue techniques for rule application to *intractable* extensions of $\mathcal{EL}^+$, providing the currently missing concept and role constructs that are required by some biomedical ontologies, notably, the Galen medical knowledge base. Work in this direction has already been studied by extending the abstract classification algorithm for $\mathcal{EL}^+$ to the one for $\mathcal{ELHI}f_{R^+}$ [Vu08]. Due to its deterministic nature, as opposed to tableau-based algorithms, the extended algorithm is expected to perform relatively well. It however remains to be empirically verified.

- Using module extraction in *full-fledged* incremental classification has been first proposed in [CHWK07]. The idea in a nutshell is to reuse certain previous subsumption information in case it is ensured with the help of modules that the information remains the same, given changes in the ontology. Also, in case that previous subsumption information cannot be reused, modules can still be exploited when new subsumption testing needs to be carried out. This could be used to enhance our (semi-)incremental classification algorithm such that the module-based technique is invoked only when an arbitrary axiom not in $\mathcal{O}_t$ is to be retracted. Addition and retraction of axioms in $\mathcal{O}_t$ are taken care of directly by the (semi-)incremental classification algorithm.

- The modularization-based approach to axiom pinpointing has proved promising both in the tractable DL $\mathcal{EL}^+$ (see [BS08] and Subsection 6.2.5) and in OWL DL [SQJH08]. The effectiveness of this approach directly depends on the size of the module and the time required to compute it. In the previous work, a trade-off between these two qualities was met by adopting the reachability-based and minimal syntactic locality-based modules, respectively. As future work, one could investigate the effectiveness of employing different kinds of (strong subsumption) modules, for instance, semantic locality-based modules.

- Apart from its native Lisp-based console, CEL supports the DIG interface which has XML and HTTP as its underlying technologies. The issue of efficiency due to communication overheads has hindered the use of the reasoner in user-friendly ontology editors like Protégé. The latest version of Protégé (version 4.0) supports a more seamless integration of the editor and a DL reasoner via the in-memory OWL API. Implementation of an OWL API wrapper for the CEL system would undoubtedly extend its usability to a wider group of users.

The outstanding empirical results unequivocally suggest that a careful (though not extremely optimized) implementation of a polynomial-time algorithm can provide practically acceptable response time and scalability that are key requirements in design and development of large-scale ontologies. Nevertheless, it has to be mentioned that tractability also comes with limitation in terms of expressivity. As such, the CEL system is never meant to be a replacement of an OWL reasoner, but rather an alternative reasoner to ontology developers, whose ontological requirements can be met by the expressivity of the DL $\mathcal{EL}^+$. Finally, it is hoped that the development of the CEL system and its empirical evaluation will shed light on the usability and usefulness of tractable Description Logics in the $\mathcal{EL}$ family, and ignite interest in adopting DL-based knowledge representation systems in industrial-scale applications.

Bibliography

[ABB$^+$00] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: Tool for the unification of biology. The gene ontology consortium. *Nature Genetics*, 25(1):25–29, May 2000. ↩ p. 3, 7

[AFF01] G. Ausiello, P. G. Franciosa, and D. Frigioni. Directed hypergraphs: Problems, algorithmic results, and a novel decremental approach. In *Proceedings of the Seventh Italian Conference on Theoretical Computer Science (ICTCS)*, volume 2202 of *Lecture Notes in Computer Science*, pages 312–327. Springer-Verlag, 2001. ↩ p. 82

[Baa96] F. Baader. Using automata theory for characterizing the semantics of terminological cycles. *Ann. of Mathematics and Artificial Intelligence*, 18:175–219, 1996. ↩ p. 4

[Baa03] F. Baader. Terminological cycles in a Description Logic with existential restrictions. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 325–330. Morgan Kaufmann, 2003. ↩ p. 3, 4, 34

[BBL05] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proceedings of the 19th International Conference on Artificial Intelligence (IJCAI-05)*, Edinburgh, UK, 2005. Morgan-Kaufmann Publishers. ↩ p. 3, 4, 5, 38, 39, 45, 51, 52, 58, 74, 105

[BBL08] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope further. In Kendall Clark and Peter F. Patel-Schneider, editors, *In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*, 2008. ↩ p. 22, 45, 49, 51

[BCM$^+$07] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press, second edition, 2007. ↩ p. 145, 146, 151

[BH95] F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. *Journal of Automated Reasoning*, 14:149–180, 1995. ↩ p. 93, 103, 105

[BHJV08] J. Bock, P. Haase, Q. Ji, and R. Volz. Benchmarking OWL reasoners. In *Proceedings of the Workshop on Advancing Reasoning on the Web: Scalability and Commonsense (ARea08)*, CEUR-WS, 2008. ↩ p. 119

[BHN$^+$92] F. Baader, B. Hollunder, B. Nebel, H.-J. Profitlich, and E. Franconi. An empirical analysis of optimization techniques for terminological representation systems, or making KRIS get a move on. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 270–281. Morgan Kaufmann, 1992. ↩ p. 46, 67, 68

[BHN$^+$94] F. Baader, B. Hollunder, B. Nebel, H.-J. Profitlich, and E. Franconi. An emperical analysis of optimization techniques for terminological representation systems or: "making KRIS get a move on". *Applied Intelligence*, 4(2):109–132, 1994. ↩ p. 4, 46, 67, 68, 69, 70

[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://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html. ↩ p. 4

[BL84] R.J. Brachman and H.J. Levesque. The tractability of subsumption in frame-based description languages. In *Proceedings of the 4th Nat. Conf. on Artificial Intelligence (AAAI'84)*, pages 34–37, 1984. ↩ p. 3, 4

[BL85] R.J. Brachman and H.J. Levesque, editors. *Readings in Knowledge Representation.* Morgan-Kaufmann Publishers, Los Altos (CA), USA, 1985. ↩ p. 2

[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 2005 International Workshop on Methods for Modalities (M4M-05)*, 2005. ↩ p. 5, 13, 118

[BLS06] F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In U. Furbach and N. Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 287–291. Springer-Verlag, 2006. ↩ p. 4, 13, 28, 39, 60

[BLS07] F. Baader, C. Lutz, and B. Suntisrivaraporn. Is tractable reasoning in extensions of the Description Logic $\mathcal{EL}$ useful in practice? *Journal of*

Logic, Language and Information, Special Issue on Method for Modality (M4M), 2007. ↩ p. 5, 13, 39

[BM07] A.R. Bradley and Z. Manna. Checking safety by inductive generalization of counterexamples to induction. In *Proceedings of Formal Methods in Computer Aided Design (FMCAD)*, 2007. ↩ p. 98

[BN07] F. Baader and W. Nutt. Chapter 2: Basic Description Logics. In *The Description Logic Handbook [BCM$^+$07]*, pages 47–104. Cambridge University Press, 2007. ↩ p. 26, 63, 93

[BNS08] F. Baader, N. Novakovic, and B. Suntisrivaraporn. A proof-theoretic subsumption reasoner for hybrid $\mathcal{EL}$-TBoxes. In *Proceedings of the 2008 International Workshop on Description Logics (DL2008)*, CEUR-WS, 2008. ↩ p. 15

[BP07] F. Baader and R. Penaloza. Axiom pinpointing in general tableaux. In *Proceedings of the 16th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods TABLEAUX'07*, LNAI, Aix-en-Provence, France, 2007. Springer. ↩ p. 93, 103

[BPS07a] F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the Description Logic $\mathcal{EL}$. In *Proceedings of the 2007 International Workshop on Description Logics (DL2007)*, CEUR-WS, 2007. ↩ p. 14

[BPS07b] F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the Description Logic $\mathcal{EL}^+$. In *Proceedings of the 30th German Conference on Artificial Intelligence (KI'07)*, LNAI, Osnabrück, Germany, 2007. Springer. ↩ p. 14, 93, 96, 103, 109

[Bra79] R.J. Brachman. On the epistemological status of semantic networks. *Associative Networks*, pages 2–50, 1979. ↩ p. 2

[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. ↩ p. 45, 51, 75

[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. ↩ p. 3, 4, 75

[Bra06] S. Brandt. *Standard and Non-standard Reasoning in Description Logics*. PhD thesis, TU Dresden, Institute for Theoretical Computer Science, Germany, 2006. ↩ p. 11, 28, 79

[BS01] F. Baader and U. Sattler. An overview of tableau algorithms for Description Logics. *Studia Logica*, 69:5–40, 2001. ↩ p. 63

[BS08] F. Baader and B. Suntisrivaraporn. Debugging SNOMED CT using axiom pinpointing in the Description Logic $\mathcal{EL}^+$. In *Proceedings of the 3rd Knowledge Representation in Medicine Conference (KR-MED'08): Representing and Sharing Knowledge Using* SNOMED, Phoenix AZ, USA, 2008. ↩ p. 14, 42, 93, 98, 110, 111, 123, 141

[CGG$^+$07] D. Calvanese, G. De Giacomo, B. C. Grau, A. Kaplunova, D. Lembo, M. Lenzerini, R. Möller, R. Rosati, U. Sattler, B. Sertkaya, B. Suntisrivaraporn, S. Tessaris, A.-Y. Turhan, and S. Wandelt. D14: Ontology-based services: Usage scenarios and test ontologies. Project deliverable, TONES, 2007. http://www.tonesproject.org. ↩ p. 43

[CHKS07] B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the right amount: Extracting modules from ontologies. In *Proceedings of WWW*, pages 717–726, Banff, Canada, 2007. ACM. ↩ p. 28, 30, 31, 80, 91, 92, 110, 123

[CHKS08] B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research (JAIR)*, 31:273–318, 2008. ↩ p. 31, 91

[CHM$^+$08] B. Cuenca Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. OWL 2: The next step for OWL. *Journal of Web Semantics*, 2008. ↩ p. 4

[CHWK07] B. Cuenca Grau, C. Halaschek-Wiener, and Y. Kazakov. History matters: Incremental ontology reasoning using modules. In *Proceedings of ISWC*, Busan, South Korea, 2007. Springer. ↩ p. 91, 92, 123, 141

[CMW$^+$08] B. Cuenca Grau, B. Motik, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 web ontology language: Profiles. World Wide Web Consortium (W3C) Working Draft, 2008. ↩ p. 4

[DG84] W.F. Dowling and J. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 1(3):267–284, 1984. ↩ p. 4, 46, 53, 85

[Don07] F.M. Donini. Chapter 3: Complexity of reasoning. In *The Description Logic Handbook [BCM$^+$07]*, pages 105–148. Cambridge University Press, 2007. ↩ p. 38

[GHH$^+$07] C. Golbreic, M. Horridge, I. Horrocks, B. Motik, and R. Shearer. OBO and OWL: Leveraging semantic web technologies for the life sciences. In *Proceedings of the 6th International Semantic Web Conference (ISWC'07)*, volume 4825 of *Lecture Notes in Computer Science*, pages 169–182. Springer, 2007. ↩ p. 8, 115

[GJ79] Michael R. Garey and David S. Johnson, editors. *Computers and Intractability — A Guide to NP-Completeness.* W. H. Freeman and Company, San Francisco CA, USA, 1979. ↩ p. 94, 99

[GLWZ06] S. Ghilardi, C. Lutz, F. Wolter, and M. Zakharyaschev. Conservative extensions in modal logics. In Guido Governatori, Ian Hodkinson, and Yde Venema, editors, *Advances in Modal Logics Volume 6*, pages 187–207. College Publications, 2006. ↩ p. 80

[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. 9

[Gru93b] T.R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993. ↩ p. 5

[GSW92] R. Greiner, B.A. Smith, and R.W. Wilkerson. A correction to the algorithm in reiter's theory of diagnosis. *Readings in model-based diagnosis*, pages 49–53, 1992. ↩ p. 99, 100

[Gua98] N. Guarino. Formal ontology in information systems. In N. Guarino, editor, *Proceedings of FOIS-98*, Trento, Italy, 1998. IOS Press, Amsterdam. ↩ p. 5

[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. 2

[HHK95] M.R. Henzinger, T.A. Henzinger, and P.W. Kopke. Computing simulations on finite and infinite graphs. In *36th Annual Symposium on Foundations of Computer Science*, pages 453–462, Milwaukee, Wisconsin, 1995. IEEE Computer Society Press. ↩ p. 5

[HKS06] I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, pages 57–67. AAAI Press, 2006. ↩ p. 4, 22

[HM01a] V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In B. Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 161–166, 2001. ↩ p. 4

[HM01b] V. Haarslev and R. Möller. RACER system description. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR'01)*, volume 2081 of *Lecture Notes in Computer Science*, pages 701–705. Springer-Verlag, 2001. ↩ p. 3, 4, 28, 117

[HMT01] V. Haarslev, R. Möller, and A.-Y. Turhan. Exploiting pseudo models for TBox and ABox reasoning in expressive Description Logics. In *Proceedings of the First International Joint Conference on Automated Reasoning (IJCAR'01)*, pages 61–75. Springer-Verlag, 2001. $\hookleftarrow$ p. 4

[HMW08] V. Haarslev, R. Möller, and S. Wandelt. The revival of structural subsumption in tableau-based Description Logic reasoners. In *Proceedings of the 2008 International Workshop on Description Logics (DL2008)*, CEUR-WS, 2008. $\hookleftarrow$ p. 5, 119

[Hol90] B. Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. In *Proceedings of the 14th German Workshop on Artificial Intelligence (GWAI'90)*, pages 38–47, London, UK, 1990. Springer-Verlag. $\hookleftarrow$ p. 93

[Hor97] I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics.* PhD thesis, University of Manchester, 1997. $\hookleftarrow$ p. 4, 7, 46, 68, 114

[Hor98] I. Horrocks. Using an expressive Description Logic: FaCT or fiction? In *KR-98*, pages 636–647, 1998. $\hookleftarrow$ p. 3, 4, 28

[HPSv03] 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. $\hookleftarrow$ p. 3

[HRG96] I. Horrocks, A.L. Rector, and C.A. Goble. A Description Logic based schema for the classification of medical data. In Franz Baader, Martin Buchheit, Manfred A. Jeusfeld, and Werner Nutt, editors, *Knowledge Representation Meets Databases, Proceedings of the 3rd Workshop KRDB'96, Budapest, Hungary, August 13, 1996*, volume 4 of *CEUR Workshop Proceedings*, 1996. $\hookleftarrow$ p. 7

[HS04] I. Horrocks and U. Sattler. Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. *Artificial Intelligence*, 160(1):79–104, 2004. $\hookleftarrow$ p. 3, 24

[HST00] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive Description Logics. *Journal of the Interest Group in Pure and Applied Logic*, 8(3):239–264, 2000. $\hookleftarrow$ p. 3, 4

[HT05] I. Horrocks and D. Tsarkov. Optimised classification for taxonomic knowledge bases. In *Proceedings of the 2005 International Workshop on Description Logics (DL'05)*, pages 184–191, 2005. $\hookleftarrow$ p. 4, 5, 70, 119

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Massachusetts, 1979. $\hookleftarrow$ p. 51

[IHT07] IHTSDO. The systematized nomenclature of medicine, clinical terms (SNOMED CT). The International Health Terminology Standards Development Organisation, 2007. `http://www.ihtsdo.org/our-standards/`. $\hookleftarrow$ p. 6, 34

[JPY88] D.S. Johnson, C.H. Papadimitriou, and M. Yannakakis. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988. $\hookleftarrow$ p. 95

[Kd03] 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. $\hookleftarrow$ p. 4

[KLWW08] Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter. CEX and MEX: Logical diff and semantic module extraction in a fragment of OWL. In Kendall Clark and Peter F. Patel-Schneider, editors, *In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*, 2008. $\hookleftarrow$ p. 80

[KPHS07] A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of OWL DL entailments. In *Proceedings of the 6th International Semantic Web Conference (ISWC'07)*, pages 267–280, 2007. $\hookleftarrow$ p. 29, 93, 97, 100, 103

[Küs00] R. Küsters. *Non-Standard Inferences in Description Logics.* PhD thesis, RWHT Aachen, Germany, 2000. $\hookleftarrow$ p. 11, 17, 28, 63, 79

[Law08] M. Lawley. Exploiting fast classification of SNOMED CT for query and integration of health data. In *Proceedings of the 3rd Knowledge Representation in Medicine Conference (KR-MED'08): Representing and Sharing Knowledge Using* SNOMED, Phoenix AZ, USA, 2008. $\hookleftarrow$ p. 75, 119

[LBF$^+$06] C. Lutz, F. Baader, E. Franconi, D. Lembo, R. Möller, R. Rosati, U. Sattler, B. Suntisrivaraporn, and S. Tessaris. Reasoning support for ontology design. In Bernardo Cuenca Grau, Pascal Hitzler, Connor Shankey, and Evan Wallace, editors, *Proceedings of the second international workshop OWL: Experiences and Directions*, 2006. $\hookleftarrow$ p. 9, 12, 43

[LLS06] D. Lembo, C. Lutz, and B. Suntisrivaraporn. D5: Tasks for ontology design and maintenance. Project deliverable, TONES, 2006. http://www.tonesproject.org. $\hookleftarrow$ p. 9, 43

[LW07] C. Lutz and F. Wolter. Conservative extensions in the lightweight Description Logic $\mathcal{EL}$. In Frank Pfenning, editor, *Proceedings of the 21th Conference on Automated Deduction (CADE-21)*, volume 4603 of *Lecture Notes in Artificial Intelligence*, pages 84–99. Springer-Verlag, 2007. $\hookleftarrow$ p. 28, 80

[MB87] R. MacGregor and R. Bates. The LOOM knowledge representation language. Technical Report ISI-RS-87-188, USC Information Sciences Institute, Marina del Rey, CA, 1987. Reprinted from Proceedings of the Knowledge-Based Systems Workshop, April 21-23, 1987. ↩ p. 36

[Min81] M. Minsky. A framework for representing knowledge. In J. Haugeland, editor, *Mind Design: Philosophy, Psychology, Artificial Intelligence*, pages 95–128. MIT Press, Cambridge, MA, 1981. ↩ p. 2

[MLPB06] T. Meyer, K. Lee, J. Pan, and R. Booth. Finding maximally satisfiable terminologies for the Description Logic $\mathcal{ALC}$. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06)*, pages 269–274, 2006. ↩ p. 93, 103

[Mot06] B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases.* PhD thesis, Univesität Karlsruhe (TH), Karlsruhe, Germany, January 2006. ↩ p. 4, 28, 117

[MSH07] B. Motik, R. Shearer, and I. Horrocks. Optimized reasoning in Description Logics using hypertableaux. In Frank Pfenning, editor, *Proc. of the 21st Conference on Automated Deduction (CADE-21)*, volume 4603 of *LNAI*, pages 67–83, Bremen, Germany, July 17–20 2007. Springer. ↩ p. 4, 28, 117, 119

[Neb88] B. Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence*, 34(3):371–383, 1988. ↩ p. 3

[Neb90] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990. ↩ p. 3, 4

[NM03] N. Noy and M. Musen. The PROMPT suite: Interactive tools for ontology mapping and merging. *International Journal of Human-Computer Studies*, 2003. ↩ p. 80, 90

[OPR95] M. O'Neil, C. Payne, and J. Read. Read Codes Version 3: A user led terminology. *Methods of Information in Medicine*, 34:187-921, 1995. ↩ p. 6, 33

[Pat06] J. Patrick. Aggregation and generalisation in SNOMED CT. In *Proceedings of the First Semantic Mining Conference on* SNOMED CT, pages 11–16, 2006. ↩ p. 36, 37

[PSK05] B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL ontologies. In *Proceedings of the 14th international conference on World Wide Web (WWW'05)*, pages 633–640. ACM, 2005. ↩ p. 29, 93, 103

[PSS93] P. Patel-Schneider and B. Swartout. Description-Logic knowledge representation system specification from the KRSS group of the ARPA knowledge sharing effort. Technical report, DARPA Knowledge Representation

System Specification (KRSS) Group of the Knowledge Sharing Initiative, 1993. ↩ p. 6, 33, 118

[Qui52] W.V. Quine. The problem of simplifying truth functions. *Journal of American Math Monthly*, 59:521 – 531, 1952. ↩ p. 107

[Qui53] W.O. Quine. On what there is. *Review of Metaphysics*, 2:21–38, 1948/1953. ↩ p. 5

[Qui67] M.R. Quillian. Word concepts: A theory and simulation of some basic capabilities. *Behavioral Science*, 12:410–430, 1967. ↩ p. 2

[Rec07] A. Rector. Chapter 13: Medical informatics. In *The Description Logic Handbook [BCM^{+}07]*, pages 436–457. Cambridge University Press, 2007. ↩ p. 6, 33, 114

[Rei87] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987. ↩ p. 30, 99, 100

[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)*, Stanford, CA, 1997. AAAI Press. ↩ p. 4, 7

[SBSS07] B. Suntisrivaraporn, F. Baader, S. Schulz, and K. Spackman. Replacing SEP-triplets in SNOMED CT using tractable Description Logic operators. In Jim Hunter Riccardo Bellazzi, Ameen Abu-Hanna, editor, *Proceedings of the 11th Conference on Artificial Intelligence in Medicine (AIME'07)*, volume 4594 of *Lecture Notes in Computer Science*, pages 287–291. Springer-Verlag, 2007. ↩ p. 11, 12, 29, 36, 39, 40, 92

[SC03] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of Description Logic terminologies. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the 17th International Conference on Artificial Intelligence (IJCAI-03)*, pages 355–362, Acapulco, Mexico, 2003. Morgan-Kaufmann Publishers. ↩ p. 28, 29, 93, 103

[SCC97] K.A. Spackman, K.E. Campbell, and R.A. Cote. SNOMED RT: A reference terminology for health care. In *Proceedings of the 1997 AMIA Annual Fall Symposium*, pages 640–644. Hanley&Belfus, 1997. ↩ p. 6, 33

[Sch91] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI'91)*, pages 466–471, 1991. ↩ p. 38

[SDMW02] K.A. Spackman, R. Dionne, E. Mays, and J. Weis. Role grouping as an extension to the Description Logic of Ontylog, motivated by concept modeling in SNOMED. In *Proceedings of the 2002 AMIA Annual Symposium*, pages 712–716. Hanley&Belfus, 2002. ↩ p. 34

[SH01] S. Schulz and U. Hahn. Medical knowledge reengineering: Converting major portions of the UMLS into a terminological knowledge base. *International Journal of Medical Informatics*, 64(2/3):207–221, 2001. ↩ p. 36

[SMH07] S. Schulz, K. Mark, and U. Hahn. Spatial location and its relevance for terminological inferences in bio-ontologies. *BMC Bioinformatics*, 2007. ↩ p. 11, 29, 92

[SMS06] S. Schulz, K. Markó, and B. Suntisrivaraporn. Complex occurrents in clinical terminologies and their representation in a formal language. In *Proceedings of the First European Conference on* SNOMED CT *(SMCS'06)*, Copenhagen, Denmark, 2006. ↩ p. 14

[Spa00] K.A. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED RT. *JAMIA*, 2000. Fall Symposium Special Issue. ↩ p. 6, 24, 33, 36, 40

[Spa01] K.A. Spackman. Normal forms for Description Logic expressions of clinical concepts in SNOMED RT. In *Proceedings of the 2001 AMIA Annual Symposium*, pages 627–631. Hanley&Belfus, 2001. ↩ p. 6, 43

[Spa05] K.A. Spackman. Rates of change in a large clinical terminology: Three years experience with SNOMED clinical terms. In *Proceedings of the 2005 AMIA Annual Symposium*, pages 714–718. Hanley&Belfus, 2005. ↩ p. 3, 6, 33

[SPC$^+$07] E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Web Semantics*, 5(2):51–53, 2007. ↩ p. 4, 28, 117

[SPSW01] M.Q. Stearns, C. Price, K.A. Spackman, and A.Y. Wang. SNOMED clinical terms: Overview of the development process and project status. In *Proceedings of the 2001 AMIA Annual Symposium*, pages 662–666. Hanley&Belfus, 2001. ↩ p. 3, 6, 33

[SQJH08] B. Suntisrivaraporn, G. Qi, Q. Ji, and P. Haase. A modularization-based approach to finding all justifications for OWL DL entailments. In John Domingue and Chutiporn Anutariya, editors, *Proceedings of the 3th Asian Semantic Web Conference (ASWC'08)*, Lecture Notes in Computer Science. Springer-Verlag, 2008. ↩ p. 15, 92, 123, 141

[SR06] J. Seidenberg and A. Rector. Web ontology segmentation: Analysis, classification and use. In *Proceedings of WWW*. ACM, 2006. ↩ p. 80, 90

[SRH98] S. Schulz, M. Romacker, and U. Hahn. Part-whole reasoning in medical ontologies revisited: Introducing SEP triplets into classification-based

Description Logics. *Journal of the American Medical Informatics Association (JAMIA)*, pages 830–834, 1998. Section VIII Standards and Policies - Issues in Knowledge Representation. $\hookleftarrow$ p. 36, 37, 40

[SSB07] S. Schulz, B. Suntisrivaraporn, and F. Baader. SNOMED CT's problem list: Ontologists' and logicians' therapy suggestions. In *Proceedings of The Medinfo 2007 Congress*, Studies in Health Technology and Informatics (SHTI-series), pages 802–806. IOS Press, 2007. $\hookleftarrow$ p. 12, 35

[SSS91] M. Schmidt-Schaubß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991. $\hookleftarrow$ p. 93

[Sun05a] B. Suntisrivaraporn. CEL—the reference manual. Available at http://lat.inf.tu-dresden.de/systems/cel, 2005. $\hookleftarrow$ p. 46, 60, 66, 71, 78, 113

[Sun05b] B. Suntisrivaraporn. Optimization and implementation of subsumption algorithms for the Description Logic $\mathcal{EL}$ with cyclic TBoxes and general concept inclusion axioms. Master thesis, TU Dresden, Germany, 2005. $\hookleftarrow$ p. 5, 7, 34, 47

[Sun08] B. Suntisrivaraporn. Module extraction and incremental classification: A pragmatic approach for $\mathcal{EL}^+$ ontologies. In Sean Bechhofer, Manfred Hauswirth, Joerg Hoffmann, and Manolis Koubarakis, editors, *Proceedings of the 5th European Semantic Web Conference (ESWC'08)*, volume 5021 of *Lecture Notes in Computer Science*, pages 230–244. Springer-Verlag, 2008. $\hookleftarrow$ p. 13, 14, 28, 30, 80, 110

[TBK$^+$06] A.-Y. Turhan, S. Bechhofer, A. Kaplunova, T. Liebig, M. Luther, R. Möller, O. Noppens, P. Patel-Schneider, B. Suntisrivaraporn, and T. Weithöner. DIG 2.0: Towards a flexible interface for Description Logic reasoners. In Bernardo Cuenca Grau, Pascal Hitzler, Connor Shankey, and Evan Wallace, editors, *Proceedings of the Second International Workshop OWL: Experiences and Directions*, November 2006. $\hookleftarrow$ p. 15

[TH06] D. Tsarkov and I. Horrocks. FaCT++ Description Logic reasoner: System description. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer, 2006. $\hookleftarrow$ p. 4, 28, 117

[Tur07] A.-Y. Turhan. *On the Computation of Common Subsumers in Description Logics*. PhD thesis, TU Dresden, Institute for Theoretical Computer Science, Germany, 2007. $\hookleftarrow$ p. 11, 17, 28, 79

[Vu08] Q.H. Vu. Subsumption in the Description Logic in $\mathcal{ELHI}f_{R^+}$ w.r.t. general TBoxes. Master thesis, TU Dresden, Germany, 2008. $\hookleftarrow$ p. 7, 140

[Zol07] E. Zolin. Description Logic complexity navigator. 2007. `http://www.cs.man.ac.uk/ ezolin/dl/`. $\hookleftarrow$ p. 38

[ZPTI06] M. Zhang, J. Patrick, D. Truran, and K. Innes. Deriving a Snomed ct data model. In *Proceedings of the First Semantic Mining Conference on* Snomed ct, pages 59–63, 2006. ↩ p. 41