Structured Objects in OWL: Representation and Reasoning

Boris Motik[†] Bernardo Cuenca Grau[†]
Ulrike Sattler[‡]

[†]Computing Laboratory,
University of Oxford, UK

[‡]Department of Computer Science,
University of Manchester, UK

February 11, 2008

Abstract

Applications of semantic technologies often require the representation of and reasoning with *structured objects*—that is, objects composed of parts connected in complex ways. Although OWL is a general and powerful language, its class descriptions and axioms cannot be used to describe arbitrarily connected structures. An OWL representation of structured objects can thus be underconstrained, which reduces the inferences that can be drawn and causes performance problems in reasoning. To address these problems, we extend OWL with *description graphs*, which allow for the description of structured objects in a simple and precise way. To represent conditional aspects of the domain, we also allow for SWRL-like rules over description graphs. Based on an observation about the nature of structured objects, we ensure decidability of our formalism. We also present a hypertableau-based decision procedure, which we implemented in the HermiT reasoner. To evaluate its performance, we have extracted description graphs from the GALEN and FMA ontologies, classified them successfully, and even detected a modeling error in GALEN.

1 Introduction

Ontologies are nowadays used in many disciplines, such as biology [27], medicine [9], geography [10], astronomy [6], and agriculture [28]. A de facto standard for ontology modeling is the Web Ontology Language (OWL),[1]

[1]In this paper, we focus on OWL DL—the most expressive of the decidable languages of the OWL family.

so most ontologies in these domains were either developed from the start using OWL or translated into OWL from other formalisms. OWL is an expressive language capable of supporting diverse applications. Its logical underpinning is given by description logics (DLs), which provide OWL with a clean model-theoretic semantics, well-understood reasoning problems, and powerful reasoners.

Structured objects—that is, objects composed of other, possibly interrelated objects—pose some well-known problems to OWL and DLs [4, 1, 26]. Such objects abound, for example, in molecular biology and the clinical sciences. Clinical ontologies such as GALEN [29], the Foundation Model of Anatomy (FMA) [24], the National Cancer Institute (NCI) Thesaurus [11], and SNOMED CT [30] are currently being used in large-scale applications. For example, SNOMED CT is being used to annotate patients' medical records in the National Programme for Information Technology (NPfIT) by the UK's National Health Service. All of these ontologies describe structured objects. For example, GALEN models the heart as consisting of the left and the right ventricles, the two atria, and the valves, all of which participate in complex relationships, such as "the two ventricles of a heart are separated by the intraventricular septum."

OWL can be used to describe domains consisting of an arbitrary or even infinite number of objects, but it only allows for axioms that can connect these objects in a certain tree-like manner. In other words, OWL enjoys (a variant of) the *tree model property* [32]: if an OWL ontology has a model, then it has a model with a tree-like (or forest-like) relational structure as well. This property is responsible for the decidability of OWL reasoning [32]; however, it prevents sufficiently accurate description of complex structured objects. This is because schema-level axioms in OWL cannot describe arbitrary relational structures. Consider the previously mentioned diamond-shaped structure involving a heart, its right and left ventricles, and a septum. In addition to a model that corresponds to the structure in which the objects are connected as expected, each schema-level description of the heart in OWL will also have a model where one heart has two septa, each as a part of the left and the right ventricle, respectively. Thus, certain consequences of the diamond-shaped structure cannot be drawn from its formulation in OWL. For example, we cannot conclude that, if the right ventricle has a perforated septum, the left ventricle also has a perforated septum.

To address this lack of expressive power, in Section 4 we propose an extension of OWL for modeling structured objects using *description graphs*. Such graphs consist of vertices labeled with atomic concepts and edges labeled with atomic roles. According to our proposed model-theoretic semantics, these graphs are class-level statements that specify general patterns of connections between objects. In addition, we allow for SWRL-like rules [12] to enable the description of conditional statements about graphs.

Extending DLs with axioms that can enforce arbitrary structures easily leads to undecidability [16]. Our formalism, however, is decidable because it can represent only structured objects whose number of parts is bounded. In practice, structured objects are usually modeled up to a certain level of granularity, which naturally determines this bound. In Section 5, we present a reasoning algorithm for the case where the OWL part is expressed in $\mathcal{SHIQ}$ [14]; it should, however, be possible to extended the algorithm to $\mathcal{SHOIQ}$ [13] and hence cover OWL DL. We thus obtain a powerful, decidable, and practicable language that combines two complementary formalisms: unbounded but tree-like structures can be described using standard OWL axioms, and the naturally bounded structured parts can be described using arbitrarily connected description graphs and rules.

We have implemented our algorithm in the DL reasoner HermiT [21].[2] The validation of our approach is currently difficult due to the lack of test data. Thus, we have devised an algorithm that extracts description graphs from OWL ontologies, and have applied it to GALEN and FMA. The resulting ontologies should be treated with caution; however, domain experts have confirmed that substantial parts of the ontologies reflect the actual human anatomy. Our transformation can thus be a starting point for a more comprehensive remodeling using description graphs. Finally, the ontologies are sufficiently complex to allow us to estimate the practicability of reasoning. We present the transformation algorithm in Section 6.

In Section 7, we discuss the results obtained by classifying the transformed ontologies. Our transformation allowed us to discover a modeling error in GALEN, which we take as indication that our formalism can indeed be useful in practice. Furthermore, classification times for the transformed ontologies are of similar orders of magnitude as for the original ontologies, even though our formalism adds considerable expressive power to OWL.

2 Preliminaries

In this section we recapitulate some well-known definitions of description logics and rules.

2.1 Description Logics

In order not to overload the technical presentation with details, in this paper we present the reasoning algorithms for the DL $\mathcal{SHIQ}$ [14]; we believe, however, that it is straightforward to extend our approach to the more expressive DL $\mathcal{SHOIQ}$ [13] that provides the logical underpinning of OWL DL. Furthermore, with minor modifications our approach can be used with DLs that provide for n-ary relations such as $\mathcal{DLR}$ [5].

[2] `http://web.comlab.ox.ac.uk/oucl/work/boris.motik/HermiT/`

A $\mathcal{SHIQ}$ signature is a triple $\Sigma = (N_C, N_R, N_I)$ consisting of mutually disjoint sets of *atomic concepts* N_C, *atomic roles* N_R, and *individuals* N_I. The set of *roles* is $N_R \cup \{R^- \mid R \in N_R\}$. For $R \in N_R$, let $\mathsf{Inv}(R) = R^-$ and $\mathsf{Inv}(R^-) = R$. The set of *concepts* is the smallest set containing $\top$ (the *top concept*), $\bot$ (the *bottom concept*), A (*atomic concept*), $\neg C$ (*negation*), $C \sqcap D$ (*conjunction*), $C \sqcup D$ (*disjunction*), $\exists R.C$ (*existential restriction*), $\forall R.C$ (*universal restriction*), $\geq n\, R.C$ (*at-least restriction*), and $\leq n\, R.C$ (*at-most restriction*), for $A \in N_C$, C and D concepts, R a role, and n a nonnegative integer.

A *TBox* $\mathcal{T}$ is a finite set of *role inclusions* $R \sqsubseteq S$ for R and S roles, *transitivity axioms* $\mathsf{Trans}(R)$ for R a role, and *general concept inclusions* (GCIs) $C \sqsubseteq D$ for C and D concepts. Let $\sqsubseteq^*_{\mathcal{T}}$ be the reflexive transitive closure of $\{R \sqsubseteq S, \mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \mid R \sqsubseteq S \in \mathcal{T}\}$. A role R is *transitive* in $\mathcal{T}$ if a role S exists such that $S \sqsubseteq^*_{\mathcal{T}} R$, $R \sqsubseteq^*_{\mathcal{T}} S$, and either $\mathsf{Trans}(S) \in \mathcal{T}$ or $\mathsf{Trans}(\mathsf{Inv}(S)) \in \mathcal{T}$; R is *simple* if no transitive role S exists with $S \sqsubseteq^*_{\mathcal{T}} R$. The following property must be satisfied for each TBox $\mathcal{T}$: in each concept $\geq n\, R.C$ and $\leq n\, R.C$ occurring in $\mathcal{T}$, the role R must be simple.[3]

An *ABox* $\mathcal{A}$ is a finite set of assertions for the form $C(a)$ (*concept assertion*), $R(a, b)$ (*role assertion*), $a \approx b$ (*equality assertion*), and $a \not\approx b$ (*inequality assertion*), where C is a concept, R is a role, and a and b are individuals. A $\mathcal{SHIQ}$ knowledge base $\mathcal{K}$ is a pair $(\mathcal{T}, \mathcal{A})$.

An *interpretation* for $\mathcal{K}$ is a pair $I = (\triangle^I, \cdot^I)$, where $\triangle^I$ is a nonempty set and $\cdot^I$ assigns an element $a^I \in \triangle^I$ to each individual a, a set $A^I \subseteq \triangle^I$ to each atomic concept A, and a relation $R^I \subseteq \triangle^I \times \triangle^I$ to each atomic role R. The function $\cdot^I$ is extended to concepts and roles as shown in the upper part of Table 1. I is a *model* of $\mathcal{K}$, written $I \models \mathcal{K}$, if it satisfies all axioms of $\mathcal{K}$ as shown in the bottom part of Table 1. The basic inference problem for $\mathcal{SHIQ}$ is checking satisfiability of $\mathcal{K}$—that is, checking whether a model of $\mathcal{K}$ exists. Other interesting inference problems, such as checking concept subsumption or query answering, can be reduced to satisfiability checking using well-known transformations [2].

Without loss of generality, we can assume that the ABox of each $\mathcal{SHIQ}$ knowledge base $\mathcal{K}$ is *extensionally reduced*—that is, its assertions contain only possibly negated atomic concepts and atomic roles. This is so because we can rewrite each assertion $R^-(a, b)$ as $R(b, a)$, and we can rewrite each assertion $C(a)$ where C is not a possibly negated atomic concept as $A_C(a)$ and $A_C \sqsubseteq C$ for A_C a fresh concept. This transformation clearly preserves satisfiability of $\mathcal{K}$ and is linear in the size of $\mathcal{K}$.

The *negation-normal form* of a concept C, written $\mathsf{nnf}(C)$, is the concept equivalent to C containing negations only in front of atomic concepts; $\dot{\neg} C$ is an abbreviation for $\mathsf{nnf}(\neg C)$. Each concept can be brought into negation-

[3] This restriction is necessary to make reasoning decidable [14].

Table 1: Model-Theoretic Semantics of $\mathcal{SHIQ}$

Semantics of Roles and Concepts		
$(R^-)^I$	$=$	$\{\langle y, x\rangle \mid \langle x, y\rangle \in R^I\}$
$\top^I$	$=$	$\triangle^I$
$\bot^I$	$=$	$\emptyset$
$(\neg C)^I$	$=$	$\triangle^I \setminus C^I$
$(C \sqcap D)^I$	$=$	$C^I \cap D^I$
$(C \sqcup D)^I$	$=$	$C^I \cup D^I$
$(\forall R.C)^I$	$=$	$\{x \mid \forall y : \langle x, y\rangle \in R^I \rightarrow y \in C^I\}$
$(\exists R.C)^I$	$=$	$\{x \mid \exists y : \langle x, y\rangle \in R^I \wedge y \in C^I\}$
$(\leq n\, S.C)^I$	$=$	$\{x \mid \sharp\{y \mid \langle x, y\rangle \in S^I \wedge y \in C^I\} \leq n\}$
$(\geq n\, S.C)^I$	$=$	$\{x \mid \sharp\{y \mid \langle x, y\rangle \in S^I \wedge y \in C^I\} \geq n\}$

Semantics of Axioms		
$I \models C \sqsubseteq D$	if	$C^I \subseteq D^I$
$I \models R \sqsubseteq S$	if	$R^I \subseteq S^I$
$I \models \mathsf{Trans}(R)$	if	$\langle x, y\rangle \in R^I \wedge \langle y, z\rangle \in R^I \rightarrow \langle x, z\rangle \in R^I$
$I \models C(a)$	if	$a^I \in C^I$
$I \models R(a, b)$	if	$\langle a^I, b^I\rangle \in R^I$
$I \models a \approx b$	if	$a^I = b^I$
$I \models a \not\approx b$	if	$a^I \neq b^I$

normal form in linear time using well-known transformations [2]. The DL $\mathcal{ALCHIQ}$ is obtained from $\mathcal{SHIQ}$ by disallowing transitive roles.

2.2 Rule Extensions of DLs

The basic principles for extending DLs with rules were laid down in [16, 7, 12]. Let $\Sigma = (N_C, N_R, N_I)$ be a $\mathcal{SHIQ}$ signature, and let N_V be a countably infinite set of *variables* disjoint from N_I. A *term* is an individual or a variable. A *predicate* is a concept, a role, or the *equality predicate* $\approx$. Concepts have arity one, and roles and $\approx$ have arity two. An *atom* over Σ has the form $P(t_1, \ldots, t_n)$, where P is a predicate of arity n, and t_i, $1 \leq i \leq n$ are terms. Atoms involving the equality predicate are usually written in the infix notation as $t_1 \approx t_2$. A rule r is an expression of the form

$$B_1 \wedge \ldots \wedge B_n \rightarrow H_1 \vee \ldots \vee H_m \tag{1}$$

where $n \geq 0$, $m \geq 0$, and B_i and H_j are atoms. The set of atoms $\{B_1, \ldots, B_n\}$ is called the *antecedent*, and the set of atoms $\{H_1, \ldots, H_m\}$ is called the *consequent*. A *program* $\mathcal{P}$ is a finite set of rules. A rule of the form (1) with $m = 0$ is usually written as $B_1 \wedge \ldots \wedge B_n \rightarrow \bot$ and is said to

Table 2: The Semantics of Rules

$I, \mu \models C(s)$	if $s^{I,\mu} \in C^I$
$I, \mu \models R(s,t)$	if $\langle s^{I,\mu}, t^{I,\mu} \rangle \in R^I$
$I, \mu \models s \approx t$	if $s^{I,\mu} = t^{I,\mu}$
$I, \mu \models \bigwedge_{i=1}^m U_i \rightarrow \bigvee_{j=1}^n V_n$	if $I, \mu \models V_j$ for some $1 \leq j \leq n$ whenever $I, \mu \models U_i$ for each $1 \leq i \leq m$
$I \models \bigwedge_{i=1}^m U_i \rightarrow \bigvee_{j=1}^n V_n$	if $I, \mu \models \bigwedge_{i=1}^m U_i \rightarrow \bigvee_{j=1}^n V_n$ for all mappings μ
$I \models \mathcal{P}$	if $I \models r$ for each rule $r \in \mathcal{P}$

have the *empty consequent*. A rule is *safe* if each variable in the rule occurs in some antecedent atom.

Let $I = (\triangle^I, \cdot^I)$ be an interpretation and $\mu : N_V \rightarrow \triangle^I$ a mapping of variables to elements of $\triangle^I$. Furthermore, let $a^{I,\mu} = a^I$ for an individual a and $x^{I,\mu} = \mu(x)$ for a variable x. Satisfaction of an atom, a rule, and a program $\mathcal{P}$ in I and μ is defined as shown in Table 2.

Without loss of generality, we can assume that no rule contains an atom $x \approx s$ in the antecedent, since such a rule is equivalent to the one obtained by replacing x with s. Finally, $A \wedge s \not\approx t \rightarrow B$ is equivalent to $A \rightarrow B \vee s \approx t$ and $A \rightarrow B \vee s \not\approx t$ is equivalent to $A \wedge s \approx t \rightarrow B$; therefore, to simplify the presentation, we do not allow the rules to contain inequality atoms.

3 Modeling Structured Objects using Logic

To understand the limitations of modeling structured objects in OWL, let us consider modeling the anatomy of the heart shown in Figure 1. This example has been derived by reconstructing the intention behind the axioms describing the heart in GALEN. We next consider possibilities for a logical interpretation of the figure.

Figure 1 could be represented in OWL using an ABox $\mathcal{A}$. ABox assertions, however, represent concrete data; thus, $\mathcal{A}$ would represent the structure of *one particular* heart. In this paper, we are concerned with modeling structured objects *at the schema level*—that is, we want to describe the general structure of *all* hearts. We should be able to instantiate such a description many times. For example, if we say that each patient has a heart, then, for each concrete patient, we should instantiate a *different* heart, each of the structure shown in Figure 1. This clearly cannot be achieved if we describe the structure of the heart using ABox assertions. Consequently, GALEN, SNOMED CT, and NCI contain only schema-level axioms and no ABox assertions.

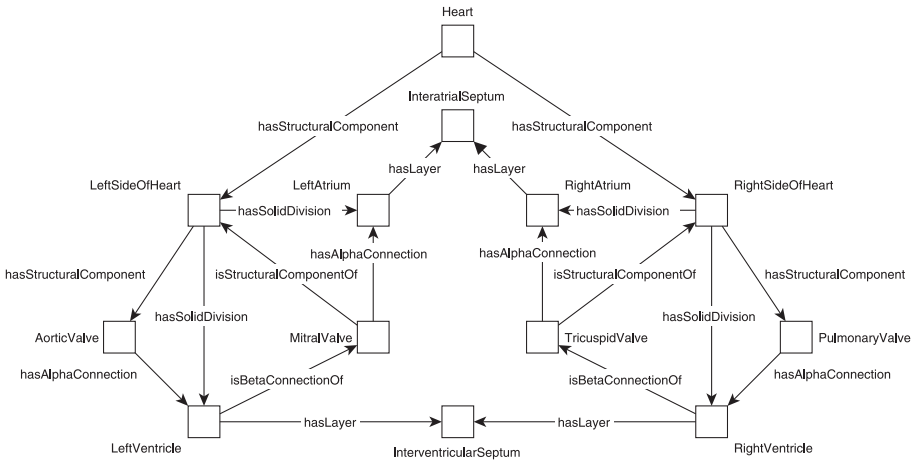

Figure 1: The Structure of the Heart

We can give a logical, schema-level interpretation to Figure 1 by treating vertices as concepts and arrows as *participation constraints* specifying relationships between concepts. For example, *LeftSideOfHeart* and *AorticValve* are concepts and the arrow between them states that each left side of the heart has an aortic valve as a structural component. Participation constraints can be represented using existential quantification, which can be encoded in OWL using axioms of the form (2). Let $\mathcal{K}$ be a DL knowledge base containing the following axioms.

$$\mathit{LeftSideOfHeart} \sqsubseteq \exists \mathit{hasStructuralComponent}.\mathit{AorticValve} \tag{2}$$

$$\mathit{AorticValve} \sqsubseteq \exists \mathit{hasAlphaConnection}.\mathit{LeftVentricle} \tag{3}$$

$$\mathit{LeftSideOfHeart} \sqsubseteq \exists \mathit{hasSolidDivsion}.\mathit{LeftVentricle} \tag{4}$$

Let I be an interpretation that corresponds to Figure 1 in the obvious way. Clearly, I is a model of $\mathcal{K}$, which justifies the formalization of Figure 1 by axioms (2)–(4).

Such a schema-level representation of a heart can be put to use in many ways. We might represent knowledge about various heart conditions; for example, if the aortic valve suffers from aortic regurgitation (AR), then the left ventricle suffers from left ventricular hypertrophy (LVH):

$$\mathit{AorticValve} \sqcap \mathit{HasAR} \sqsubseteq \forall \mathit{hasAlphaConnection}.\mathit{HasLVH} \tag{5}$$

We might expect to derive from (2)–(5) that, if the aortic valve of the left side of the heart suffers from aortic regurgitation, then the left ventricle

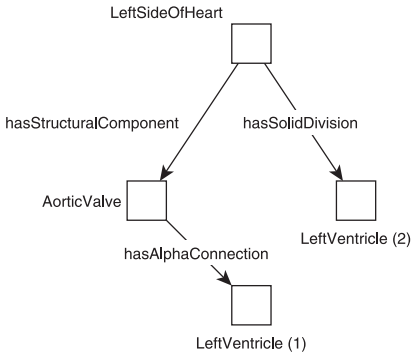

Figure 2: An Unintended Tree Model of $\mathcal{K}$

suffers from hypertrophy:

$$\begin{aligned} &\mathit{LeftSideOfHeart} \sqcap \\ &\quad \exists \mathit{hasStructuralComponent}.(\mathit{AorticValve} \sqcap \mathit{HasAR}) \sqsubseteq \\ &\qquad \exists \mathit{hasSolidDivision}.\mathit{HasLVH} \end{aligned} \tag{6}$$

Unfortunately, (6) does not follow from $\mathcal{K}$: axioms (3) and (4) imply the existence of two left ventricles, but no axiom in $\mathcal{K}$ states that these two ventricles are necessarily the same object. Thus, an interpretation I' corresponding to Figure 2 is also a model of $\mathcal{K}$. In I', even if the aortic valve has aortic regurgitation, the second left ventricle is unaffected. Hence, $I' \not\models (6)$, so $\mathcal{K} \not\models (6)$ as well.

The knowledge base $\mathcal{K}$ is thus underconstrained: some models of $\mathcal{K}$ do not correspond to the actual structure of the heart shown in Figure 1. This discrepancy can prevent us from drawing some quite reasonable conclusions, such as (6). Furthermore, it can also cause problems with the performance of reasoning. For example, we might use axioms (4) and (7)–(8) to describe the relationships between the left side of the heart, the left ventricle, and the mitral valve.

$$\mathit{LeftVentricle} \sqsubseteq \exists \mathit{isBetaConnectionOf}.\mathit{MitralValve} \tag{7}$$

$$\mathit{MitralValve} \sqsubseteq \exists \mathit{isStructuralComponentOf}.\mathit{LeftSideOfHeart} \tag{8}$$

While admitting a model corresponding to Figure 1, these axioms do not state that the mitral valve in (7) is a structural component of the "initial" left side of the heart. Hence, the interpretation from Figure 3 is also a model of these axioms. In fact, the latter model is "canonical" in the sense that it reflects the least amount of information derivable from the axioms. In order to disprove an entailment from these axioms, an OWL reasoner will try to construct such a "canonical" model. In practice, such models can be highly

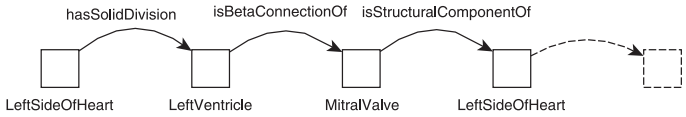

Figure 3: An Unintended Infinite Model of $\mathcal{K}$

repetitive and much larger than the intended models, which, according to our experience, is the main reason why OWL reasoners still cannot process ontologies such as FMA and certain versions of GALEN.

To avoid such problems, we need to extend $\mathcal{K}$ with additional axioms that make *all* models of $\mathcal{K}$ correspond as much as possible to the intended conceptualization shown in Figure 1. Such axioms, however, cannot be stated in OWL, for reasons we explain next. OWL can represent *unbounded* or even *infinite* domains, which is appropriate in many cases. For example, in the domain of people, we should not make any assumptions about the number of people in the world. In other words, the domain of all people does not exhibit a *natural bound* on its size. Thus, we can represent the fact that every person has exactly two parents who are persons:

$$\mathit{Person} \sqsubseteq \geq 2\, \mathit{hasParent}.\mathit{Person} \sqcap \leq 2\, \mathit{hasParent}.\top \tag{9}$$

Reasoning with such axioms is not straightforward. A model containing one person γ must contain two parents δ_1 and δ_2, each of which requires the existence of two additional parents and so on. Effectively, we obtain a model that is similar to the one shown in Figure 3.

To ensure termination of the model construction outlined in the previous paragraph, the structure of the axioms allowed in OWL is restricted such that the language exhibits (a variant of) the *tree model property* [32]: whenever a knowledge base $\mathcal{K}$ has a model, it also has a model of a certain tree shape. The relationship between the left side of the heart, the aortic valve, and the left ventricle in Figure 1 is, however, triangular and cannot be represented as a tree. Hence, if we want to ensure that the ventricles whose existence is implied by (3) and (4) are the same in *every* model of $\mathcal{K}$, we must leave the confines of OWL and DLs.

Certain rule formalisms can axiomatize nontree structures. For example, the following SWRL [12] rule can be used to make the two ventricles from Figure 2 the same:

$$\begin{aligned} &\mathit{LeftSideOfHeart}(x) \wedge \mathit{hasStructuralComponent}(x, y) \wedge \\ &\quad \mathit{hasAlphaConnection}(y, z) \wedge \mathit{LeftVentricle}(z) \wedge \\ &\quad \mathit{hasSolidDivsion}(x, w) \wedge \mathit{LeftVentricle}(w) \rightarrow z \approx w \end{aligned} \tag{10}$$

This, however, has significant drawbacks. From the standpoint of modeling, such a solution is quite complex, as it requires the modeler to anticipate which objects need to be made the same. The fact that the two left ventricles are the same follows from the complex interaction between axioms (2)–(4) and (10), and is thus not represented explicitly. Clearly, such a modeling formalism is likely to be hard to use and susceptible to modeling errors. From the standpoint of automated reasoning, the extension of OWL with SWRL is undecidable [12], which is a significant impediment to the adoption of SWRL in practice.

SWRL-like rules can, however, naturally express certain conditional aspects of structured objects. For example, if the septum has a ventricular septal defect, then there is a blood flow from the left to the right ventricle:

$$\begin{aligned} &\mathit{IntraventricularSeptum}(x) \wedge \mathit{HasVSD}(x) \wedge \\ &\quad \mathit{hasLayer}(y_1, x) \wedge \mathit{LeftVentricle}(y_1) \wedge \\ &\quad \mathit{hasLayer}(y_2, x) \wedge \mathit{RightVentricle}(y_2) \rightarrow \mathit{hasBloodFlow}(y_1, y_2) \end{aligned} \tag{11}$$

The variables in the antecedent of this rule are connected in a non-tree-like way, so such a rule cannot be expressed in OWL. If we, however, deal with arbitrarily connected structures, such as the one shown in Figure 1, non-tree-like antecedents are essential for drawing the correct inferences.

Various decidable combinations of DLs and rules cannot be used for schema modeling. For example, the DL-safe rules [20] are syntactically restricted such that they apply only to the explicitly named objects. Role-safe [16] and weakly safe [23] rules also impose restrictions that prevent the application of the rules to arbitrary elements of the domain, and similar restrictions are also employed by various nonmonotonic rule extensions of OWL [8, 23, 19]. While these are quite useful in query answering, they cannot be used to derive new conclusions from the schema.

The DL $\mathcal{SROIQ}$ [15] and the OWL 1.1 extension of OWL DL extend OWL with *complex role inclusions* of the form $R_1 \circ \ldots \circ R_n \sqsubseteq S$, restricted appropriately to ensure decidability. Such axiom solve some of the problems; however, they still cannot axiomatize arbitrary structures such as the one in Figure 1 or express axioms such as (11).

4 The Formalism for Structured Objects

We now present an extension of OWL that addresses the problems outlined in Section 3. In Section 4.1, we present the basic principles behind our idea, which we then formalize in Section 4.2.

4.1 Basic Principles

The main aspect of a description of a structured object is the connection between the object's parts, which can naturally be represented as a graph.

Hence, we introduce the notion of a *description graph* $G = (V, E, \lambda)$—a directed graph in which each vertex $i \in V$ is labeled with a set of atomic concepts $\lambda\langle i \rangle$ and each edge $\langle i, j \rangle \in E$ is labeled with a set of atomic roles $\lambda\langle i, j \rangle$. For example, Figure 1 can be understood as a description graph that describes the heart.

Semantically, $G = (V, E, \lambda)$ should be understood as a "template" for a fragment of a model. Let I be a model and A an atomic concept labeling some graph vertex $i \in V$. If I contains an object γ such that $\gamma \in A^I$, then I must also contain an instance of G in which γ corresponds to i. For example, if I contains an instance γ of the *Heart* concept, then I must contain a relational structure corresponding to Figure 1 in which γ corresponds to the top-most vertex.

As discussed in Section 3, extending DLs with constructs that allow the description of arbitrarily connected structures of unbounded size easily leads to undecidability. In practice, structured objects are usually modeled up to a certain level of granularity, which naturally determines this bound. For example, a human body consists of a certain number of organs. These organs might be decomposed into smaller parts; however, each such decomposition is bounded, so the entire model of human anatomy requires a bounded number of objects. Even though the number of required objects may be large and difficult to determine by hand, the fact that the domain is bounded is intrinsic to the modeling problem. The reasoning algorithm presented in Section 5 uses this bound to ensure termination even on arbitrarily connected, non-tree-like structures.

We assume that the set of atomic roles is divided into a set of *atomic tree roles* N_{R_t} and a set of *atomic graph roles* N_{R_g}. A *graph-extended DL knowledge base* is a 4-tuple $\mathcal{K} = (\mathcal{T}, G, \mathcal{P}, \mathcal{A})$ where $\mathcal{T}$ is a DL TBox, G is a description graph, $\mathcal{P}$ is a set of rules, and $\mathcal{A}$ is an ABox. Furthermore, $\mathcal{T}$ is allowed to refer only to tree roles, G and $\mathcal{P}$ are allowed to refer only to the graph roles, and $\mathcal{A}$ is allowed to refer to both graph and tree roles.

For example, let $\mathcal{K} = (\mathcal{T}, G, \mathcal{P}, \mathcal{A})$ be a graph-extended DL knowledge base with the following components. Let $\mathcal{T}$ contain the axioms (12)–(14). Intuitively, axiom (12) says that each person has a parent and a heart; axiom (13) ensures that the heart of each sufferer from aortic regurgitation is an instance of *HasAR*; and axiom (14) says that, on each aortic valve suffering from aortic regurgitation, some person is performing a surgery on it.

$$\mathit{Person} \sqsubseteq \exists \mathit{hasParent}.\mathit{Person} \sqcap \exists \mathit{hasHeart}.\mathit{Heart} \tag{12}$$

$$\mathit{AR\_Sufferer} \sqsubseteq \forall \mathit{hasHeart}.\mathit{HasAR} \tag{13}$$

$$\mathit{AorticValve} \sqcap \mathit{HasAR} \sqsubseteq \exists \mathit{performsSurgeryOn}^{-}.\mathit{Person} \tag{14}$$

Let G correspond to Figure 1, and let $\mathcal{P}$ contain the rule that propagates the *HasAR* concept over the structural components of the heart.

$$\mathit{HasAR}(x) \wedge \mathit{hasStructuralComponent}(x, y) \rightarrow \mathit{HasAR}(y) \tag{15}$$

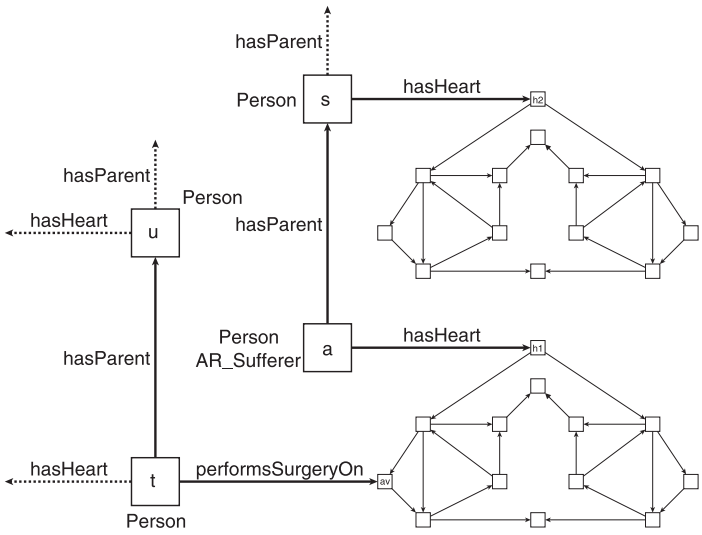

Figure 4: A Typical Model of $\mathcal{K}$

Let $\mathcal{A}$ contain the assertions $Person(a)$ and $AR\_Sufferer(a)$.

The semantics of graph-extended knowledge bases ensures that each model I of $\mathcal{K}$ is of the form shown in Figure 4. The model I consists of two distinct parts. The *tree backbone* consists of objects (shown as large squares) connected through tree roles (shown using thick lines), and it is constructed using the standard DL axioms in $\mathcal{T}$. As discussed in Section (3), the number of persons is not naturally bounded so, if we want a decidable formalism, we must employ standard DL restrictions. Apart from the tree backbone, I also contains arbitrarily connected but naturally bounded *graph instances*, such as the structure of the heart of each person. Unlike in the case of axioms (2)–(4) and Figure 2, each graph instance is *necessarily* of the form as specified by G in each such model I. Note that the tree backbone of I need not be contiguous: the bottom-most *AorticValve* object av can be connected to other objects through tree roles. To summarize, for a graph-extended knowledge base $\mathcal{K}$, we can consider only models that consist of graph instances, connected among themselves and with other objects through tree roles.

Decidability of the formalism is now ensured by the separation of the roles into tree and graph ones. The axioms in $\mathcal{T}$ can propagate constraints across tree roles just like in standard DLs; however, we can adapt the blocking technique [14] to ensure termination of model construction. Furthermore, the rules in $\mathcal{P}$ can propagate constraints within a graph; however, the size of the graph is naturally bounded, so this does not cause termination

problems either.

Our way of obtaining decidability is related to fusions of abstract description systems (ADSs) [3], which provide for the combination of different modal and description logics. The component ADSs can share concepts; however, the interaction between them through roles is restricted to ensure decidability. Our separation of roles into graph and tree ones is similar in spirit. Bounded structures and rules, however, cannot directly be expressed as an ADS. In addition, we present a practical decision procedure for our formalism.

This example also demonstrates a minor modeling problem. The intention behind axioms (13) and (15) is to mark the aortic valve of each sufferer from aortic regurgitation with the *HasAR* concept. Doing this in two separate axioms seems unnatural, and the following, more compact DL axiom can be used for this purpose:

$$\begin{aligned} AR\_Sufferer \sqsubseteq \forall hasHeart.\forall hasStrucutralComponent. \\ \forall hasStrucutralComponent.HasAR \end{aligned} \tag{16}$$

This is currently not possible because DL axioms are not allowed to contain graph roles. This problem, however, can be solved by appropriate syntactic sugar. For example, we might extend the definition of $\mathcal{T}$ to allow for graph roles to occur under $\forall$ quantifier; then, for reasoning, we might replace each subconcept concept $\forall R.C$ where R is a graph role using a new name that we axiomatize appropriately using rules. Such extensions, however, do not change the essence of our formalism, so we do not discuss them further.

Our way of obtaining obtaining decidability is related to fusions of abstract description systems (ADSs) [3], which allow one to combine different modal and description logics. The component ADSs can share concepts; however, the interaction between them through roles is restricted to ensure decidability. Our separation of roles into graph and tree ones is similar in spirit. Bounded structures and rules, however, cannot directly be expressed as an ADS (an encoding might exist, but it is not obvious and is unlikely to be suitable for practical usage). In addition, we present a practical decision procedure for our formalism.

4.2 Formalization

We now present the formal definition of our formalism. We start by separating the tree form the graph roles.

Definition 1 (Graph-Extended DL Signature)**.** *A* graph-extended DL signature *is a 4-tuple* $\Sigma = (N_C, N_{R_t}, N_{R_g}, N_I)$ *consisting of pair-wise disjoint sets of atomic concepts* N_C*,* atomic tree roles N_{R_t}*,* atomic graph roles N_{R_g}*, and individuals* N_I*.*

All subsequent definitions in this paper are implicitly parameterized with a graph-extended DL signature. Next, we formalize the notion of description graphs. We make the technical assumption that the vertices of the description graph are integers, as this allows us to use vertices as indices.[4]

Definition 2 (Description Graph)**.** *A* description graph $G = (V, E, \lambda)$ *is a directed labeled graph where*

- $V = \{1, \ldots, \ell\}$ *is a finite set of integers called* vertices*,*
- $E \subseteq V \times V$ *is a set of* edges*, and*
- λ *is a labeling function that assigns a set of atomic concepts* $\lambda\langle i\rangle \subseteq N_C$ *to each vertex* $i \in V$*, and a set of atomic graph roles* $\lambda\langle i, j\rangle \subseteq N_{R_g}$ *to each edge* $\langle i, j\rangle \in E$*.*

For an atomic concept A*, let* $V_A = \{k \in V \mid A \in \lambda\langle k\rangle\}$*.*

Finally, we define the notion of graph-extended DL knowledge bases. The definition of graph-regular rules ensures that each such rule can become applicable only to objects from the same instance of the description graph G. As discussed in the previous section, this is required for decidability of the formalism.

Definition 3 (Graph-Extended KBs)**.** *A* graph-extended DL knowledge base *is a 4-tuple* $\mathcal{K} = (\mathcal{T}, G, \mathcal{P}, \mathcal{A})$*, where different components have the following structure:*

- $\mathcal{T}$ *is a TBox over the signature* (N_C, N_{R_t}, N_I) *expressed in some description logic* $\mathcal{DL}$*.*
- G *is a description graph.*
- $\mathcal{P}$ *is a program consisting of a finite number of* graph-regular rules*. A rule* r *is graph-regular if it uses only atomic concepts and graph roles and for each two variables* x_1 *and* x_2 *occurring in* r*, some antecedent atom of* r *contains both* x_1 *and* x_2*.*
- $\mathcal{A}$ *is an extensionally-reduced ABox whose assertions can refer to all elements in the signature* $(N_C, N_{R_t}, N_{R_g}, N_I)$*; furthermore, in addition to the standard ABox assertions of* $\mathcal{DL}$*, the ABox* $\mathcal{A}$ *can contain* graph assertions *of the form* $G(a_1, \ldots, a_\ell)$*, where* $a_i \in N_I$ *and* ℓ *is the number of vertices of the description graph* G*.*

[4]For example, given a vector of variables $x_1, \ldots, x_\ell$ and a vertex $i \in V$, we can refer to the variable x_i.

Graph-regular rules can express conjunctive queries over G, so we do not consider query answering separately. We now formalize the semantics of description graphs.

Definition 4 (Semantics)**.** *An interpretation $I = (\triangle^I, \cdot^I)$ interprets a description graph $G = (V, E, \lambda)$ with ℓ vertices as an ℓ-ary relation $G^I \subseteq (\triangle^I)^\ell$. Moreover, I satisfies G, written $I \models G$, if all of the following conditions hold.*

- *The i-*key *property must hold for each $1 \leq i \leq \ell$:*

$$\begin{aligned}&\forall x_1, \dots, x_\ell, y_1, \dots, y_\ell \in \triangle^I : \\ &\quad \langle x_1, \dots, x_\ell \rangle \in G^I \wedge \langle y_1, \dots, y_\ell \rangle \in G^I \wedge x_i = y_i \rightarrow \bigwedge_{1 \leq j \leq \ell} x_j = y_j\end{aligned}$$

- *The* disjointness *property must hold:*

$$\begin{aligned}&\forall x_1, \dots, x_\ell, y_1, \dots, y_\ell \in \triangle^I : \\ &\quad \langle x_1, \dots, x_\ell \rangle \in G^I \wedge \langle y_1, \dots, y_\ell \rangle \in G^I \rightarrow \bigwedge_{1 \leq i < j \leq n} x_i \neq y_j\end{aligned}$$

- *The A-*start *property must hold for each atomic concept A with $V_A \neq \emptyset$:*

$$\begin{aligned}&\forall x \in \triangle^I : \\ &\quad x \in A^I \rightarrow \exists x_1, \dots, x_\ell \in \triangle^I : \langle x_1, \dots, x_\ell \rangle \in G^I \wedge \bigvee_{k \in V_A} x = x_k\end{aligned}$$

- *The* vertex layout *property must hold for each $i \in V$ and $A \in \lambda\langle i \rangle$:*

$$\forall x_1, \dots, x_\ell \in \triangle^I : \langle x_1, \dots, x_\ell \rangle \in G^I \rightarrow x_i \in A^I$$

- *The* edge layout *property must hold for each $\langle i, j \rangle \in E$ and $R \in \lambda\langle i, j \rangle$:*

$$\forall x_1, \dots, x_\ell \in \triangle^I : \langle x_1, \dots, x_\ell \rangle \in G^I \rightarrow \langle x_i, x_j \rangle \in R^I$$

The intuition behind this definition is as follows. Each tuple in the ℓ-ary relation G^I corresponds to one instance of the description graph G.

The i-key properties and the disjointness property ensure that no two instances of G can share an object, which essentially captures the idea behind natural boundedness. Consider the axiom $B \sqsubseteq A$ and a graph G consisting of two vertices 1 and 2 such that $\lambda\langle 1 \rangle = \{A\}$, $\lambda\langle 2 \rangle = \{B\}$, and $\lambda\langle 1, 2 \rangle = \{R\}$. Without the i-key and the disjointness properties, we could build a model I of G where $\gamma_1 \in A^I$, $\langle \gamma_1, \gamma_2 \rangle \in R^I$, and $\gamma_2 \in B^I$; to ensure that $B^I \subseteq A^I$, we must also set $\gamma_2 \in A^I$; but then, we must instantiate G for γ_2, which clearly leads to a cyclic computation. The i-key and the disjointness properties ensure that such a model I cannot exist: each object occurring in a graph part of I occurs in *exactly* one tuple of G^I. Since this tuple is bounded in

size, each graph part of I is bounded as well, which can be used to ensure termination of model construction.

The A-start property ensures that I contains an appropriate instance of G whenever I contains an instance γ of a concept A labeling a vertex of G. If A labels more than one vertex of G, the A-start property "guesses" the vertex of G that γ should be matched to. Consider, for example, a graph containing a vertex labeled with *Hand* and five vertices labeled with *Finger*. If some object γ is an instance of *Finger*, without further information we cannot disambiguate which of the five fingers γ stands for. Therefore, we need to make a "guess" and examine all five possibilities independently. Note that no other property requires guessing; hence, unless we label two vertices in G with the same concept A, the semantic properties of graphs allow for deterministic reasoning.

Finally, the vertex and edge layout properties simply ensure that each instance of G indeed contains the appropriate relational structure of G.

A satisfaction of a graph-extended knowledge base $\mathcal{K} = (\mathcal{T}, G, \mathcal{P}, \mathcal{A})$ in an interpretation I is defined in the obvious way: $I \models \mathcal{K}$ iff $I \models \mathcal{T}$, $I \models G$, $I \models \mathcal{P}$, and $I \models \mathcal{A}$.

5 Reasoning with Graph-Extended KBs

We now present an algorithm for reasoning with a graph-extended knowledge base $\mathcal{K} = (\mathcal{T}, G, \mathcal{P}, \mathcal{A})$ with $\mathcal{T}$ expressed in $\mathcal{SHIQ}$. We extend the hypertableau algorithm, which has been implemented in the HermiT reasoner and has proven very effective in practice [21]: HermiT is currently the only reasoner that can classify the original version of GALEN—a long-standing open problem in DL reasoning. In the following section, we present an overview of our algorithm and, in the subsequent sections, we introduce the algorithm formally and prove its soundness, completeness, and termination.

5.1 Algorithm Overview

Our algorithm decides satisfiability of $\mathcal{K} = (\mathcal{T}, G, \mathcal{P}, \mathcal{A})$. All other interesting reasoning problems, such as subsumption and instance checking, can be reduced to satisfiability checking as in standard DLs [2, Chapter 2]. Satisfiability of $\mathcal{K}$ is checked in two main phases. The task of the *preprocessing phase* is to translate $\mathcal{T}$ and G into a set of rules of the form (1) that are equisatisfiable with $\mathcal{T}$ and G. In the *hypertableau phase*, our algorithm attempts to construct a model satisfying the rules obtained in the preprocessing phase, the rules in $\mathcal{P}$, and the ABox assertions in $\mathcal{A}$.

The TBox $\mathcal{T}$ is preprocessed into a set of rules with $\Xi(\mathcal{T})$ exactly as it is done in [21]. To make this paper self-contained, we repeat the translation in Section 5.2. The first preprocessing step is to remove transitivity axioms from $\mathcal{T}$. Thus, $\mathcal{T}$ is converted into a TBox $\Omega(\mathcal{T})$ that does not contain

transitivity axioms but is equisatisfiable with $\mathcal{T}$. The next step is to convert each concept inclusion from $\Omega(\mathcal{T})$ into a normalized form shown below, for A_i, B_i, C_i, and D_i atomic concepts:

$$\top \sqsubseteq \bigsqcup (\neg)A_i \sqcup \bigsqcup \forall R_i.(\neg)B_i \sqcup \bigsqcup \geq n_i\, S_i.(\neg)C_i \sqcup \bigsqcup \leq m_i\, R_i.(\neg)D_i \quad (17)$$

This greatly simplifies the structure of the axioms, as the normalized axioms do not contain implicit negations or complex nested subconcepts. Note that concepts of the form $\exists R.C$ are represented in the normalized form as $\geq 1\, R.C$. We denote the result of the normalization with $\Delta(\mathcal{T})$. The final preprocessing step is to translate $\Delta(\mathcal{T})$ into an equivalent set of rules $\Xi(\mathcal{T})$. This is done by expanding the $\forall R.C$ and $\geq n\, R.C$ concepts according to the standard semantics. For example, the axiom $\top \sqsubseteq \exists R.A \sqcup \forall R.\neg B$ is translated into the rule $R(x, y) \rightarrow (\exists R.A)(x) \vee B(y)$.

The description graph G is translated into a set of rules $\Xi(G)$ that encodes the conditions of Definition 4. The i-key, disjointness, vertex, and edge layout properties are encoded as rules in a straightforward way. To encode the A-start property, we first extend the rule consequents to allow for concepts of the form $\exists G|_k$; intuitively, $\exists G|_k(x)$ is true in a model I if x occurs in I in some instance of G at vertex k. For each concept A labeling vertices $i_1, \ldots, i_k$ of G, the following rule is added to $\Xi(G)$:

$$A(x) \rightarrow \exists G|_{i_1}(x) \vee \ldots \vee \exists G|_{i_k}(x) \quad (18)$$

For example, if G contains vertices 4 and 9 that are labeled with the *Ventricle* concept, the following rule is added to $\Xi(G)$:

$$\mathit{Ventricle}(x) \rightarrow \exists G|_4(x) \vee \exists G|_9(x) \quad (19)$$

Intuitively, each instance x of the *Ventricle* concept must occur in some instance of G either at vertex 4 or vertex 9. Hence, this rule encodes the "guessing" that we discussed in Section 4.2.

The preprocessing produces a set of rules $\mathcal{R} = \Xi(\mathcal{T}) \cup \Xi(G) \cup \mathcal{P}$ that is equisatisfiable with $(\mathcal{T}, G, \mathcal{P})$. Thus, satisfiability of $\mathcal{K} = (\mathcal{T}, G, \mathcal{P}, \mathcal{A})$ can be shown by proving satisfiability of $(\mathcal{R}, \mathcal{A})$, which can be done using our hypertableau algorithm. The algorithm is presented formally in Section 5.4. It attempts to construct a representation of a model of $(\mathcal{R}, \mathcal{A})$ by applying different *inference rules* to $\mathcal{A}$ and $\mathcal{R}$, which produce new ABoxes $\mathcal{A}_1, \ldots, \mathcal{A}_n$. All inference rules are such that at least one resulting ABox contains more information about a model for $\mathcal{A}$ and $\mathcal{R}$. For example, if $A(s) \in \mathcal{A}$ and $A(x) \rightarrow B(x) \vee C(x) \in \mathcal{R}$, an application of the *Hyp*-rule to $\mathcal{A}$ and $\mathcal{R}$ produces ABoxes $\mathcal{A} \cup \{B(s)\}$ and $\mathcal{A} \cup \{C(s)\}$; intuitively, in each model of $(\mathcal{R}, \mathcal{A})$, either $B(s)$ or $C(s)$ must be true as well. If $\geq 1\, R.C(s) \in \mathcal{A}$, an application of the $\geq$-rule produces an ABox $\mathcal{A} \cup \{R(s, t), C(t)\}$ with t a new individual; intuitively, in each model of $\mathcal{A}$, the individual s must be

connected to some individual which is also an instance of C. If $s \approx t \in \mathcal{A}$, an application of the $\approx$-rule produces an ABox in which, roughly speaking, s is replaced with t; intuitively, if s and t denote the same object, we can refer to that object using just one of the two names. Finally, if $s \not\approx s \in \mathcal{A}$ or $\{A(s), \neg A(s)\} \subseteq \mathcal{A}$, then $\mathcal{A}$ is obviously unsatisfiable, which is detected by the $\bot$-rule. Finally, the $\exists G$-rule is a new rule that is similar to the $\geq$-rule: if $\exists G|_i(s) \in \mathcal{A}$, this rule derives $\mathcal{A} \cup \{G(t_1, \ldots, t_{i-1}, s, t_{i+1}, \ldots, t_\ell)\}$ with t_j new individuals.

It is easy to see that, on axioms (4) and (7)–(8), our algorithm would generate a model shown in Figure 3, and would therefore not terminate. In standard (hyper)tableau algorithms, termination is achieved using blocking: roughly speaking, if two individuals s and s' occur in the same concepts in $\mathcal{A}$, then s "behaves" just like s'—that is, we do not expand s any further. Thus, to ensure decidability, we adapt the well-known pair-wise blocking [14] to our setting. Roughly speaking, we separate the individuals in $\mathcal{A}$ into named, tree, and graph individuals. The named individuals are the ones that occur in the original graph-extended knowledge base, the tree individuals are introduced by an application of the $\geq$-rules, and the graph individuals are introduced by an application of the $\exists G$ rule. We use this distinction in the definition of blocking: only tree individuals can be blocked, and the blocking individual must also be a tree individual.

5.2 Preprocessing the DL TBox into Rules

We now present the part of the preprocessing phase that translates a $\mathcal{SHIQ}$ TBox $\mathcal{T}$ into a set of equisatisfiable rules $\Xi(\mathcal{T})$.

Elimination of Transitivity Axioms. We first encode a $\mathcal{SHIQ}$ TBox $\mathcal{T}$ into an equisatisfiable $\mathcal{ALCHIQ}$ knowledge base $\Omega(\mathcal{T})$. Roughly speaking, an axiom $\mathsf{Trans}(S)$ is replaced with axioms $\forall R.C \sqsubseteq \forall S.(\forall S.C)$, for each R with $S \sqsubseteq^* R$ and C a "relevant" concept from $\mathcal{T}$. This encoding is polynomial and has been presented several times for various description [31] and modal [25] logics. Therefore, we omit the details of the transformation and refer the reader to [17, Section 5.2]. After this transformation, there is no distinction between simple and complex roles, so, without loss of generality, in the rest of this paper we treat $\exists R.C$ as a syntactic shortcut for $\geq 1\, R.C$.

Structural Transformation. Axioms are next brought into a certain normalized form, defined as follows:

Definition 5. *For A an atomic concept, the concepts A, $\neg A$, $\top$, and $\bot$ are called* literal concepts. *A GCI is* normalized *if it is of the form $\top \sqsubseteq \bigsqcup_{i=1}^{n} C_i$, where each C_i is of the form B, $\forall R.B$, $\geq n\, R.B$, or $\leq n\, R.B$, and B is a literal concept. A TBox $\mathcal{T}$ is* normalized *if all GCIs in it are normalized.*

Table 3: The Structural Transformation

$$\Delta(\mathcal{T}) = \bigcup_{C_1 \sqsubseteq C_2 \in \mathcal{T}} \Delta(\top \sqsubseteq \mathsf{nnf}(\neg C_1 \sqcup C_2))$$

$$\Delta(\top \sqsubseteq \mathbf{C} \sqcup C') = \Delta(\top \sqsubseteq \mathbf{C} \sqcup \alpha_{C'}) \cup \bigcup_{i=1}^{n} \Delta(\top \sqsubseteq \dot{\neg}\alpha_{C'} \sqcup C_i) \text{ for } C' = \prod_{i=1}^{n} C_i$$

$$\Delta(\top \sqsubseteq \mathbf{C} \sqcup \forall R.D) = \Delta(\top \sqsubseteq \mathbf{C} \sqcup \forall R.\alpha_D) \cup \Delta(\top \sqsubseteq \dot{\neg}\alpha_D \sqcup D)$$

$$\Delta(\top \sqsubseteq \mathbf{C} \sqcup \geq n\, R.D) = \Delta(\top \sqsubseteq \mathbf{C} \sqcup \geq n\, R.\alpha_D) \cup \Delta(\top \sqsubseteq \dot{\neg}\alpha_D \sqcup D)$$

$$\Delta(\top \sqsubseteq \mathbf{C} \sqcup \leq n\, R.D) = \Delta(\top \sqsubseteq \mathbf{C} \sqcup \leq n\, R.\dot{\neg}\alpha_{D'}) \cup \Delta(\top \sqsubseteq \dot{\neg}\alpha_{D'} \sqcup D') \text{ for } D' = \dot{\neg}D$$

$$\Delta(\beta) = \{\beta\} \text{ for any other axiom } \beta$$

$$\alpha_C = \begin{cases} Q_C & \text{if } \mathsf{pos}(C) = \mathsf{true} \\ \neg Q_C & \text{if } \mathsf{pos}(C) = \mathsf{false} \end{cases} \quad \text{where } Q_C \text{ is a fresh atomic concept unique for } C$$

$$\mathsf{pos}(\top) = \mathsf{false} \qquad \mathsf{pos}(\bot) = \mathsf{false}$$

$$\mathsf{pos}(A) = \mathsf{true} \qquad \mathsf{pos}(\neg A) = \mathsf{false}$$

$$\mathsf{pos}(C_1 \sqcap C_2) = \mathsf{pos}(C_1) \vee \mathsf{pos}(C_2) \qquad \mathsf{pos}(C_1 \sqcup C_2) = \mathsf{pos}(C_1) \vee \mathsf{pos}(C_2)$$

$$\mathsf{pos}(\forall R.C_1) = \mathsf{pos}(C_1)$$

$$\mathsf{pos}(\geq n\, R.C_1) = \mathsf{true}$$

$$\mathsf{pos}(\leq n\, R.C_1) = \begin{cases} \mathsf{pos}(\dot{\neg}C_1) & \text{if } n = 0 \\ \mathsf{true} & \text{otherwise} \end{cases}$$

Note: A is an atomic concept, C_i are arbitrary concepts, $\mathbf{C}$ is a possibly empty disjunction of arbitrary concepts, and D is not a literal concept. In $\mathbf{C} \sqcup C$, the concept C should be understood as any, and not just the right-most disjunct of $\mathbf{C} \sqcup C$.

A TBox $\mathcal{T}$ can be brought into normalized form $\Delta(\mathcal{T})$ as follows:

Definition 6. *For $\mathcal{T}$ an $\mathcal{ALCHIQ}$ TBox, $\Delta(\mathcal{T})$ is the TBox computed as shown in Table 3.*

Intuitively, $\mathsf{pos}(C)$ is used to determine whether C should be replaced with a positive or a negative literal concept α_C. If $\mathsf{pos}(C) = \mathsf{false}$, then C can be converted into clauses with only negative literals, so we rename C by a negative literal concept $\neg Q_C$; otherwise, the clausification of C requires at least one positive literal, so we rename C by a positive literal concept Q_C. This prevents the introduction of new disjunctions due to normalization [21]. We now show that normalization does not affect satisfiability.

Lemma 1. *A graph-extended knowledge base $(\mathcal{T}, G, \mathcal{P}, \mathcal{A})$, where $\mathcal{T}$ is an $\mathcal{ALCHIQ}$ TBox, is satisfiable if and only if $(\Delta(\mathcal{T}), G, \mathcal{P}, \mathcal{A})$ is satisfiable; $\Delta(\mathcal{T})$ can be computed in polynomial time; and $\Delta(\mathcal{T})$ is normalized.*

Proof. It is easy to see that our transformation is a syntactic variant of the structural transformation from [22], from which the first two claims follow. Observe that Δ essentially rewrites each GCI into a form $\top \sqsubseteq \bigsqcup_{i=1}^{n} C_i$ and then keeps replacing nested subconcepts of C_i as long as the GCI is not normalized. □

Translation into Rules. We now show how to transform a normalized TBox into an equivalent set of rules.

Table 4: Translation of Normalized GCIs to Rules

$$\Xi(\mathcal{T}) = \{\left[\bigwedge_{i=1}^{n} \mathsf{lhs}(C_i)\right] \rightarrow \left[\bigvee_{i=1}^{n} \mathsf{rhs}(C_i)\right] \mid \text{ for each } \top \sqsubseteq \bigsqcup_{i=1}^{n} C_i \text{ in } \mathcal{T}\} \cup$$
$$\{\mathsf{ar}(R,x,y) \rightarrow \mathsf{ar}(S,x,y) \mid \text{ for each } R \sqsubseteq S \text{ in } \mathcal{T}\}$$

$$\mathsf{ar}(R,s,t) = \begin{cases} R(s,t) & \text{if } R \text{ is an atomic role} \\ S(t,s) & \text{if } R \text{ is an inverse role and } R = S^- \end{cases}$$

Note: Whenever $\mathsf{lhs}(C_i)$ or $\mathsf{rhs}(C_i)$ is undefined, it is omitted in the rule.

C	$\mathsf{lhs}(C)$	$\mathsf{rhs}(C)$
A		$A(x)$
$\neg A$	$A(x)$	
$\geq n\, R.A$		$\geq n\, R.A(x)$
$\geq n\, R.\neg A$		$\geq n\, R.\neg A(x)$
$\forall R.A$	$\mathsf{ar}(R,x,y_C)$	$A(y_C)$
$\forall R.\neg A$	$\mathsf{ar}(R,x,y_C) \wedge A(y_C)$	
$\leq n\, R.A$	$\bigwedge_{i=1}^{n+1}[\mathsf{ar}(R,x,y_C^i) \wedge A(y_C^i)]$	$\bigvee_{i=1}^{n+1}\bigvee_{j=i+1}^{n+1} y_C^i \approx y_C^j$
$\leq n\, R.\neg A$	$\bigwedge_{i=1}^{n+1} \mathsf{ar}(R,x,y_C^i)$	$\bigvee_{i=1}^{n+1} A(y_C^i) \vee \bigvee_{i=1}^{n+1}\bigvee_{j=i+1}^{n+1} y_C^i \approx y_C^j$

Note: Each variable $y_C^{(i)}$ is unique for C (and i), and it is different from x.

Definition 7. *For a normalized $\mathcal{ALCHIQ}$ TBox $\mathcal{T}$, the set of rules $\Xi(\mathcal{T})$ is obtained as shown in Table 4.*

To simplify the inference rules presented in Section 5.4, the role atoms in $\Xi(\mathcal{T})$ involve only atomic roles. Thus, the function ar from Table 4 is used to convert inverse role atoms $R^-(s,t)$ in $\Xi(\mathcal{T})$ into atomic role atoms $R(t,s)$. An inverse role can occur only in concepts of the form $\geq n\, R^-.C$, and the ar function is used when expanding such concepts to produce atoms containing atomic roles.

We now show that translation of a TBox into rules does not affect its satisfiability.

Lemma 2. *Let $\mathcal{T}$ be a normalized $\mathcal{ALCHIQ}$ TBox. Then, $I \models \mathcal{T}$ if and only if $I \models \Xi(\mathcal{T})$.*

Proof. The following equivalences between DLs and first-order logic are known:

$$\forall R.C(x) \equiv \forall y : \neg R(x,y) \vee C(y)$$
$$\leq n\, R.C(x) \equiv \forall y_1, \ldots, y_{n+1} : \bigvee_{1=i}^{n+1} [\neg R(x,y_i) \vee \neg C(y_i)] \vee \bigvee_{1 \leq i < j \leq n+1} y_i \approx y_j$$

Clearly, $\Xi(\mathcal{T})$ is obtained from normalized GCIs by expanding the concepts $\forall R.C$ and $\leq n\, R.C$ according to these equivalences, and then moving all negative atoms into the antecedent and all positive atoms into the consequent of the rule. □

5.3 Preprocessing the Description Graph into Rules

We now present the part of the preprocessing phase that translates a description graph G into a set of graph-regular rules $\Xi(G)$. In Definition 8 we introduce some auxiliary notions; then, we present the actual translation in Definition 9.

Definition 8. *A* graph existence concept *is an expression of the form $\exists G|_i$, where $G = (V, E, \lambda)$ is a description graph with ℓ vertices and $i \in V$. It is interpreted in an interpretation I as follows:*

$$(\exists G|_i)^I = \{s \mid \exists t_1, \ldots, t_\ell : \langle t_1, \ldots, t_\ell \rangle \in G^I \wedge s = t_i\}$$

*In the rest of this paper, we extend the definition of graph-regular rules (see Definition 3) to allow for (*i*) graph atoms $G(x_1, \ldots, x_n)$ in antecedents, and (*ii*) graph existence atoms $\exists G|_i(x)$ in consequents, where $x_{(i)}$ are variables. The semantics of such rules is defined in the obvious way.*

Definition 9. *Given a description graph $G = (V, E, \lambda)$ with ℓ vertices, the set $\Xi(G)$ consists of the following rules:*

1. *The following rule is instantiated for each $i, j \in V$ such that $j \neq i$:*

$$G(x_1, \ldots, x_\ell) \wedge G(y_1, \ldots, y_{i-1}, x_i, y_{i+1}, \ldots, y_\ell) \rightarrow x_j \approx y_j;$$

2. *The following rule is instantiated for each $1 \leq i < j \leq \ell$:*

$$G(x_1, \ldots, x_\ell) \wedge G(y_1, \ldots, y_{j-1}, x_i, y_{j+1}, \ldots, y_\ell) \rightarrow \bot;$$

3. *The following rule is instantiated for each atomic concept A such that $V_A \neq \emptyset$:*

$$A(x) \rightarrow \bigvee_{k \in V_A} \exists G|_k(x);$$

4. *The following rule is instantiated for each $i \in V$ and $A \in \lambda\langle i \rangle$:*

$$G(x_1, \ldots, x_\ell) \rightarrow A(x_i);$$

5. *The following rule is instantiated for each $\langle i, j \rangle \in E$ and $R \in \lambda\langle i, j \rangle$:*

$$G(x_1, \ldots, x_\ell) \rightarrow R(x_i, x_j).$$

By taking into account the semantics of the concepts $\exists G|_k$ from Definition 8, the rules in $\Xi(G)$ obviously encode the conditions of Definition 4, which implies the following lemma.

Lemma 3. *Let $G = (V, E, \lambda)$ be a description graph and I an interpretation. Then, $I \models G$ if and only if $I \models \Xi(G)$.*

5.4 The Hypertableau Calculus

We now present a calculus for checking satisfiability of $(\mathcal{R}, \mathcal{A})$, for $\mathcal{R}$ a set of rules, G a description graph, and $\mathcal{A}$ an ABox. To check satisfiability of a graph-extended DL knowledge base $\mathcal{K} = (\mathcal{T}, G, \mathcal{P}, \mathcal{A})$, we shall apply the calculus to $\mathcal{R} = \Xi(\mathcal{T}) \cup \Xi(G) \cup \mathcal{P}$. Our algorithm, however, can handle to any set of rules $\mathcal{R}$ that satisfies certain conditions. These conditions are more general than what is strictly necessary to handle $\Xi(\mathcal{T}) \cup \Xi(G) \cup \mathcal{P}$; for example, they allow for rules such as $A(x) \wedge R(x, y) \rightarrow S(x, y)$, which can be useful in practice.

Definition 10. *A set of rules $\mathcal{R}$ is* admissible *if it can be represented as a disjoint union of two subsets $\mathcal{R}_t$ and $\mathcal{R}_g$ satisfying the following conditions.*

The set $\mathcal{R}_g$ can contain only graph-regular rules, and it must contain all the rules specified in items 1 and 2 of Definition 9 for each description graph G occurring in $\mathcal{R}$ and $\mathcal{A}$.

Each rule $r \in \mathcal{R}_t$ must be tree-like*—that is, it must be possible to separate the variables of r into one* center variable *x and the set of* leaf variables *$\{y_i\}$ such that the following conditions are satisfied, for R an atomic tree role, A an atomic concept, and C a concept of the form $\geq n\, S.A$ or $\geq n\, S.\neg A$ with S a (not necessarily atomic) tree role:*

- *Each atom in the antecedent of r is of the form $A(x)$, $A(y_i)$, $R(x, y_i)$, or $R(y_i, x)$.*
- *Each atom in the consequent is of the form $A(x)$, $A(y_i)$, $C(x)$, $C(y_i)$, $R(x, y_i)$, $R(y_i, x)$, or $y_i \approx y_j$.*
- *Each variable y_i in the rule occurs in some binary atom in the antecedent.*

By inspecting the types of rules generated in Definitions 7 and 9, it is easy to see that, for an arbitrary graph-extended DL knowledge base $\mathcal{K} = (\mathcal{T}, G, \mathcal{P}, \mathcal{A})$ where $\mathcal{T}$ is expressed in the DL $\mathcal{SHIQ}$, the set of rules $\mathcal{R} = \Xi(\mathcal{T}) \cup \Xi(G) \cup \mathcal{P}$ is admissible. We are now ready to formally define our hypertableau calculus.

Definition 11. ***Generalized Individuals.*** *Let* T *and* Γ *be two disjoint countably infinite sets of* tree *and* graph symbols*. A* generalized individual *is a finite string of symbols $\alpha_0.\alpha_1.\ \ldots\ .\alpha_n$ such that $\alpha_0 \in N_I$, $\alpha_i \in \mathrm{T} \cup \Gamma$ for $1 \leq i \leq n$, and $\alpha_{i-1} \in \Gamma$ implies $\alpha_i \notin \Gamma$. If $\alpha_n \in N_I$, the individual is* named*; if $\alpha_n \in \mathrm{T}$, the individual is a* tree individual*; and if $\alpha_n \in \Gamma$, the individual is a* graph individual*.*

Successors and Predecessors. *A generalized individual $x.\alpha$ is a* successor *of x,* predecessor *is the inverse of successor, and* descendant *and* ancestor *are the transitive closures of successor and predecessor, respectively.*

Graph Cluster. *Generalized individuals s and t* are from the same graph clusters *if either (*i*) s is either a named individual or a graph successor of a named individual, and t is also either a named individual or a graph successor of a named individual, (*ii*) both s and t are graph successors of the same tree individual, or (*iii*) one individual is a graph successor of the other individual.*

Generalized ABox. *We generalize the notion of ABoxes by allowing them to contain generalized individuals in the assertions. Furthermore, since $\approx$ and $\not\approx$ are symmetric relations, we take $a \approx b$ and $a \not\approx b$ to also stand for the assertions $b \approx a$ and $b \not\approx a$. Finally, we allow an ABox to contain a special assertion $\bot$ that is false in all interpretations. Unless otherwise noted, all ABoxes in the rest of this paper are generalized.*

Initial ABox. *An ABox is said to be* initial *if it contains only named individuals, it is extensionally reduced, and it is not empty.*

Pairwise Anywhere Blocking. *A concept is* blocking-relevant *if it is of the form A, $\geq n\, R.A$, $\geq n\, R.\neg A$, or $\exists G|_i$, for A an atomic concept, R a (not necessarily atomic) role, and G a description graph. The* labels *of an individual and of an individual pair in an ABox $\mathcal{A}$ are defined as follows:*

$$\mathcal{L}_{\mathcal{A}}(s) = \{C \mid C(s) \in \mathcal{A} \text{ and } C \text{ is a blocking-relevant concept}\}$$
$$\mathcal{L}_{\mathcal{A}}(s,t) = \{R \mid R(s,t) \in \mathcal{A}\}$$

Let $\prec$ be a strict ordering (i.e., a transitive and irreflexive relation) on the generalized individuals containing the ancestor relation—that is, if s' is an ancestor of s, then $s' \prec s$. By induction on $\prec$, we assign to each individual s in $\mathcal{A}$ a status as follows:

- *s is* directly blocked by *an individual s' iff all of the following is true, for t and t' the predecessors of s and s', respectively:*
  - *both s and s' are tree individuals,*
  - *s' is not blocked,*
  - *$s' \prec s$,*
  - *$\mathcal{L}_{\mathcal{A}}(s) = \mathcal{L}_{\mathcal{A}}(s')$ and $\mathcal{L}_{\mathcal{A}}(t) = \mathcal{L}_{\mathcal{A}}(t')$, and*
  - *$\mathcal{L}_{\mathcal{A}}(s,t) = \mathcal{L}_{\mathcal{A}}(s',t')$ and $\mathcal{L}_{\mathcal{A}}(t,s) = \mathcal{L}_{\mathcal{A}}(t',s')$.*
- *s is* indirectly blocked *iff its predecessor is blocked.*
- *s is* blocked *iff it is either directly or indirectly blocked.*

Pruning. *The ABox $\mathsf{prune}_{\mathcal{A}}(s)$ is obtained from $\mathcal{A}$ by removing all assertions that contain a descendant of s.*

Merging. *The ABox $\mathsf{merge}_{\mathcal{A}}(s \to t)$ is obtained from $\mathsf{prune}_{\mathcal{A}}(s)$ by replacing the individual s with the individual t in all assertions.*

Clash. *An ABox $\mathcal{A}$ contains a* clash *if and only if $\bot \in \mathcal{A}$; otherwise, $\mathcal{A}$ is* clash-free.

Derivation Rules. *Table 5 specifies* derivation rules *that, given a clash-free ABox $\mathcal{A}$ and a set of rules $\mathcal{R}$, derive the ABoxes $\langle \mathcal{A}_1, \ldots, \mathcal{A}_n \rangle$. In the Hyp-rule, σ is a mapping from the set of variables N_V to the individuals occurring in $\mathcal{A}$, and $\sigma(U)$ is obtained from U by replacing each variable x with $\sigma(x)$.*

Rule Priority. *The $\exists G$-rule is applicable only if no other rule is applicable.*

Derivation. *A* derivation *for a set of admissible rules $\mathcal{R}$ and an initial ABox $\mathcal{A}$ is a pair (T, ρ) where T is a finitely branching tree and ρ is a function that labels the nodes of T with ABoxes such that (*i*) $\rho(\epsilon) = \mathcal{A}$ for ϵ the root of the tree, and (*ii*) for each node t, if one or more derivation rules are applicable to $\rho(t)$ and $\mathcal{R}$, then t has children $t_1, \ldots, t_n$ such that the ABoxes $\langle \rho(t_1), \ldots, \rho(t_n) \rangle$ are the result of applying one applicable derivation rule chosen by respecting the rule priority. A derivation is* clash-free *if it contains a leaf node labeled with a clash-free ABox.*

The main difference to the algorithm presented in [21] is the distinction between named, tree, and graph individuals. To understand why this is important, assume that Figure 4 depicts the contents of an ABox $\mathcal{A}$, generated using a set of tree rules $\mathcal{R}_t$ corresponding to axioms (12)–(14) and a graph G shown in Figure 1. The individual a is clearly named. The individual h_1 is generated by deriving $\exists \mathit{hasHeart}.\mathit{Heart}(a)$ by (12) and then expanding it by the $\geq$-rule; hence, h_1 is a tree individual. All other individuals that correspond to the structure of the heart (including av) are created by instantiating G, so they are graph individuals. We say that h_1 and all these individuals are from the say graph cluster—that is, they are allowed to occur in one instance of the graph. Note that each graph cluster can contain arbitrarily many graph individuals, but at most one tree individual which is used to "enter" the graph.

Our calculus ensures that each application of an inference rule does not violate the structure of an ABox outlined in the previous paragraph. The main problem is with the $\approx$-rule: if we merged, say, the tree individual h_2 into h_1 by simply replacing h_2 with h_1, the upper instance of G (see Figure 1) would contain individual h_1 which is from the wrong graph cluster than as all other individuals in the graph instance. This, however, is prevented by pruning: before replacing h_2 with h_1, we prune h_2, which also removes the graph instance surrounding h_2. The following lemma proves that each ABox labeling a derivation node is indeed of the form outlined above.

Lemma 4. *The following properties hold for each ABox $\mathcal{A}'$ labeling a node in a derivation for an admissible set of rules $\mathcal{R}$ and an initial ABox $\mathcal{A}$, where a and b are named individuals, u is a generalized individual, $\gamma_i, \gamma_j \in \Gamma$, and $\tau_i, \tau_j \in \mathrm{T}$.*

Table 5: Derivation Rules of the Hypertableau Calculus

***Hyp*-rule**
If 1. $U_1 \wedge ... \wedge U_m \rightarrow V_1 \vee ... \vee V_n \in \mathcal{R}$, 2. a mapping $\sigma : N_V \rightarrow N_\mathcal{A}$ exists such that 2.1 $\sigma(U_i) \in \mathcal{A}$ for each $1 \leq i \leq m$ and 2.2 $\sigma(V_j) \notin \mathcal{A}$ for each $1 \leq j \leq n$, then $\mathcal{A}_1 = \mathcal{A} \cup \{\bot\}$ if $n = 0$; or $\mathcal{A}_j := \mathcal{A} \cup \{\sigma(V_j)\}$ for $1 \leq j \leq n$ if $n > 0$.
$\geq$-rule
If 1. $\geq n\, R.C(s) \in \mathcal{A}$, 2. s is not blocked in $\mathcal{A}$, and 3. there are no individuals $u_1, \ldots, u_n$ such that $\{\mathsf{ar}(R, s, u_i), C(u_i) \mid 1 \leq i \leq n\} \cup \{u_i \not\approx u_j \mid 1 \leq i < j \leq n\} \subseteq \mathcal{A}$, then $\mathcal{A}_1 := \mathcal{A} \cup \{\mathsf{ar}(R, s, t_i),\ C(t_i) \mid 1 \leq i \leq n\} \cup \{t_i \not\approx t_j \mid 1 \leq i < j \leq n\}$ where $t_1, \ldots, t_n$ are fresh pairwise distinct tree successors of s.
$\exists G$-rule
If 1. $\exists G|_i(s) \in \mathcal{A}$ for G a description graph with ℓ vertices, 2. s is not blocked in $\mathcal{A}$, and 3. there are no individuals $u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_\ell$ such that $G(u_1, \ldots, u_{i-1}, s, u_{i+1}, \ldots, u_\ell) \in \mathcal{A}$ then $\mathcal{A}_1 := \mathcal{A} \cup \{G(t_1, \ldots, t_{i-1}, s, t_{i+1}, \ldots, t_\ell)\}$ where $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_\ell$ are fresh pairwise distinct graph individuals from the same graph cluster as s.
$\approx$-rule
If $s \approx t \in \mathcal{A}$ and $s \neq t$ then $\mathcal{A}_1 := \mathsf{merge}_\mathcal{A}(s \rightarrow t)$ if t is a named individual or if s is a descendant of t; or $\mathcal{A}_1 := \mathsf{merge}_\mathcal{A}(t \rightarrow s)$ otherwise.
$\bot$-rule
If $s \not\approx s \in \mathcal{A}$ or $\{A(s), \neg A(s)\} \subseteq \mathcal{A}$ then $\mathcal{A}_1 := \mathcal{A} \cup \{\bot\}$.
Note: $\mathcal{A}$ is a generalized ABox, $\mathcal{R}$ is a set of admissible rules, and $N_\mathcal{A}$ is the set of individuals occurring in $\mathcal{A}$.

1. *Each $R(s,t) \in \mathcal{A}'$ with R a tree role is of the form $R(a,b)$, $R(u, u.\tau_i)$, or $R(u.\tau_i, u)$.*

2. *Each $s \approx t \in \mathcal{A}'$ is of the form $u \approx u$, $a \approx b$, $a \approx b.\tau_i$, $u.\tau_i \approx u.\tau_j$, $u \approx u.\tau_i.\tau_j$, $u \approx u.\gamma_i$, $u.\gamma_i \approx u.\gamma_j$, $a \approx b.\gamma_i$, or $a.\gamma_i \approx b.\gamma_j$.*

3. *In each graph assertion and each assertion involving a graph role, all individuals are from the same graph cluster.*

4. *For each tree individual t_n, a sequence$(s_0, t_0), \ldots, (s_n, t_n)$ of pairs of individuals—called a* link to root *for t_n of length n—exists such that (*i*) s_0 is a named individual, (*ii*) each t_i is a tree successor of s_i, (*iii*) t_{i-1} is from the same graph cluster as s_i for each $1 \leq i \leq n$,*

*(*iv*) for each* $0 \leq i \leq n$*, either* $R(s_i, t_i) \in \mathcal{A}'$ *or* $R(t_i, s_i) \in \mathcal{A}'$ *for some tree role* R*.*

Proof. We prove the claim by induction on the derivation depth. For the base case, note that the ABox $\mathcal{A}$ contains only named individuals, so all claims are trivially satisfied. For the induction step, assume that the claim holds for some ABox $\mathcal{A}'$ and consider an ABox $\mathcal{A}''$ obtained from $\mathcal{A}'$ by an application of a derivation rule.

An application of the *Hyp*-rule to a tree-like rule r can introduce an assertion of the form $R(s, t)$ if r contains an atom $R(x, y)$ in the consequent. Since r is tree-like, its antecedent contains an atom $S(x, y)$ or $S(y, x)$ for S a tree role. This atom is matched to an assertion $S(s, t)$ or $S(t, s)$ in $\mathcal{A}'$ that satisfies Property (1) by induction assumption. But then, $R(s, t)$ also satisfies Property (1), and so does $\mathcal{A}''$.

An application of the *Hyp*-rule to a tree-like rule r can introduce an equality assertion of the form $s \approx t$ if the consequent of r contains an atom $y_1 \approx y_2$. But then, the antecedent of r contains either $S(x, y_1) \wedge S(x, y_2)$ or $S(x, y_1) \wedge S(y_2, x)$. The following table summarizes the ways in which $S(x, y_1) \wedge S(x, y_2)$ can be matched to assertions in $\mathcal{A}'$ and the types of equalities that can be derived. The case for $S(x, y_1) \wedge S(y_2, x)$ is symmetric, so $\mathcal{A}''$ satisfies Property (2).

$S(x, y_1)$	$S(x, y_2)$	$y_1 \approx y_2$
$S(u, u.\tau_i)$	$S(u, u.\tau_j)$	$u.\tau_i \approx u.\tau_j$
$S(u.\tau_i, u.\tau_i.\tau_j)$	$S(u.\tau_i, u)$	$u.\tau_i.\tau_j \approx u$
$S(a, a.\tau_i)$	$S(a, b)$	$a.\tau_i \approx b$
$S(a, b)$	$S(a, c)$	$b \approx c$

An application of the *Hyp*-rule to a graph-regular rule r can introduce a concept assertion, a role assertion with a graph role, or an equality assertion. By Definitions 3 and 8, each atom in the antecedent of r is either a graph, an atomic concept, or a graph role atom. Furthermore, each two variables occurring in r occur together in some atom A in the antecedent of r. Hence, A is matched to an assertion in $\mathcal{A}'$ that satisfies Property (3) by induction assumption; thus, all variables of r are matched to individuals from the same graph cluster in the inference. It is then easy to see that the resulting assertion satisfies Properties (2) and (3).

An application of the $\geq$-rule can introduce an assertion of the form $R(s, t_i)$ or $R(t_i, s)$. But then, t_i is a tree successor of s, so $\mathcal{A}''$ satisfies Property (1). If s is a graph individual, let s' be the predecessor of s; otherwise, let $s' = s$. Then, s' is a named or a tree individual for which Property (4) holds, so Property (4) holds for t_i as well.

An application of the $\exists G$-rule introduces graph assertions of the form $G(u_1, \ldots, u_{i-1}, s, u_{i+1}, \ldots, u_\ell)$, where u_j are from the same graph cluster as s. Clearly, $\mathcal{A}''$ satisfies Property (3).

Consider an application of the $\approx$-rule to an equality of the form $a \approx b.\tau_i$, $u.\tau_i \approx u.\tau_j$, $u \approx u.\tau_i.\tau_j$. The following table summarizes the types of assertions that can be produced by merging:

Replacing $b.\tau_i$ with a in ...	produces ...	
$R(b, b.\tau_i)$	$R(b, a)$	
$R(b.\tau_i, b)$	$R(a, b)$	
$c \approx b.\tau_i$	$c \approx a$	
Replacing $u.\tau_j$ with $u.\tau_i$ in ...	**produces ...**	
$R(u, u.\tau_j)$	$R(u, u.\tau_i)$	
$R(u.\tau_j, u)$	$R(u.\tau_i, u)$	
$u.\tau_k \approx u.\tau_j$	$u.\tau_k \approx u.\tau_i$	
$v \approx v.\tau_k.\tau_j$	$v \approx v.\tau_k.\tau_i$	for $u = v.\tau_k$
Replacing $u.\tau_i.\tau_j$ with u in ...	**produces ...**	
$R(u.\tau_i, u.\tau_i.\tau_j)$	$R(u.\tau_i, u)$	
$R(u.\tau_i.\tau_j, u.\tau_i)$	$R(u, u.\tau_i)$	
$u.\tau_i.\tau_k \approx u.\tau_i.\tau_j$	$u.\tau_i.\tau_k \approx u$	
$u.\tau_i.\tau_k \approx u$	$u \approx u$	

All assertions containing a tree successor of the merged individual are removed, so $\mathcal{A}'$ satisfies Property (1). Furthermore, if the merged individual occurs in a graph assertion, an assertion with a graph role, or an equality assertion containing a graph individual, by Property (3) this assertion contains a successor of the merged individual, so it is pruned. Thus, $\mathcal{A}''$ satisfies Properties (2) and (3). Pruning removes all the successors of some individual, so $\mathcal{A}''$ satisfies Property (4).

Consider an application of the $\approx$-rule to an equality of the form $u \approx u.\gamma_j$, $u.\gamma_i \approx u.\gamma_j$, $a \approx b$, $a \approx b.\gamma_i$, $a.\gamma_i \approx b.\gamma_j$. The following table summarizes the types of assertions that can be produced by merging in the first case:

Replacing $u.\gamma_j$ with u in ...	produces ...	
$a \approx b.\gamma_i$	$a \approx b$	for $b = u$
$u.\gamma_k \approx u.\gamma_j$	$u.\gamma_k \approx u$	
$u \approx u.\gamma_j$	$u \approx u$	

The remaining four types of equalities produce similar types of assertions. This inference prunes one individual and replaces it with another one from the same graph cluster, so $\mathcal{A}''$ satisfies Properties (1)–(3). Pruning removes all the successors of some individual, so $\mathcal{A}''$ satisfies Property (4). □

The following lemma shows that our inference rules are sound.

Lemma 5 (Soundness)**.** *Let $\mathcal{R}$ be an admissible set of rules and $\mathcal{A}$ an ABox, and let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be obtained by applying a derivation rule to $\mathcal{R}$ and $\mathcal{A}$. If $(\mathcal{R}, \mathcal{A})$ is satisfiable, then $(\mathcal{R}, \mathcal{A}_i)$ is satisfiable for some $1 \leq i \leq n$.*

Proof. Let I be a model of $(\mathcal{R}, \mathcal{A})$, and let us consider all possible applications of derivation rules shown in Table 5.

(*Hyp*-rule) Since $\sigma(U_i) \in \mathcal{A}$, we have $I \models \sigma(U_i)$ for all $1 \leq i \leq m$. But then, $n > 0$ and $I \models \sigma(V_j)$ for some $1 \leq j \leq n$. Since $\mathcal{A}_j = \mathcal{A} \cup \{\sigma(V_j)\}$, we conclude that $I \models (\mathcal{R}, \mathcal{A}_j)$.

($\geq$-rule) Since $\geq n\, R.C(s) \in \mathcal{A}$, we have $I \models \geq n\, R.C(s)$, which means that $\alpha_1, \ldots, \alpha_n \in \triangle^I$ exist such that $\langle s^I, \alpha_i \rangle \in R^I$ and $\alpha_i \in C^I$ for $1 \leq i \leq n$, and $\alpha_i \neq \alpha_j$ for $1 \leq i < j \leq n$. Let I' be obtained from I by setting $t_i^{I'} = \alpha_i$. Clearly, $I' \models \mathsf{ar}(R, s, t_i)$, $I' \models C(t_i)$, and $I' \models t_i \not\approx t_j$ for $i \neq j$. Therefore, we have $I' \models (\mathcal{R}, \mathcal{A}_1)$.

($\exists G$-rule) Since $\exists G|_i(s) \in \mathcal{A}$, we have $I \models \exists G|_i(s)$, which means that individuals $\alpha_1, \ldots \alpha_\ell \in \triangle^I$ exist such that $\langle \alpha_1, \ldots \alpha_\ell \rangle \in G^I$ and $\alpha_i = s$. Let I' be obtained from I by setting $t_j^I = \alpha_j$ for $1 \leq j \leq \ell$ and $j \neq i$. Clearly, $I' \models (\mathcal{R}, \mathcal{A}_1)$.

($\approx$-rule) Since $s \approx t \in \mathcal{A}$, we have $I \models s \approx t$, so $s^I = t^I$. Pruning removes assertions, so I is a model of the pruned ABox by monotonicity. Merging simply replaces an individual with an individual that is interpreted in the same way so, clearly, $I \models (\mathcal{R}, \mathcal{A}_1)$.

($\bot$-rule) Since $I \models (\mathcal{R}, \mathcal{A})$, the precondition of the $\bot$-rule cannot be satisfied, so the rule cannot be applied to $\mathcal{R}$ and $\mathcal{A}$. □

The following lemma shows that our calculus is complete. Our proof adapts the well-known unraveling technique [14].

Lemma 6 (Completeness)**.** *Let $\mathcal{R}$ be an admissible set of rules and $\mathcal{A}$ an initial ABox. If a derivation for $\mathcal{R}$ and $\mathcal{A}$ exists in which a leaf node is labeled with a clash-free ABox $\mathcal{A}'$, then $(\mathcal{R}, \mathcal{A})$ is satisfiable.*

Proof. To construct a model of $(\mathcal{R}, \mathcal{A})$, we first introduce the following definitions. A *path* is a finite sequence of pairs of individuals $p = [\frac{x_0}{x'_0}, \ldots, \frac{x_n}{x'_n}]$ with $n \geq 0$. Let $\mathsf{tail}(p) = x_n$ and $\mathsf{tail}'(p) = x'_n$. Furthermore, let $q = [p \mid \frac{x_{n+1}}{x'_{n+1}}]$ be the path $[\frac{x_0}{x'_0}, \ldots, \frac{x_n}{x'_n}, \frac{x_{n+1}}{x'_{n+1}}]$; we say that q is a *successor* of p, and p is a *predecessor* of q. Furthermore, paths p and q are from the same *graph cluster* if either (*i*) $\mathsf{tail}(p)$ is either a named individual or a graph successor of a named individual, and $\mathsf{tail}(q)$ is also either a named individual or a graph successor of a named individual, (*ii*) both $\mathsf{tail}(p)$ and $\mathsf{tail}(q)$ are graph individuals and p and q are successors of the same path, or (*iii*) one path, say p, is a successor of the other path q and $\mathsf{tail}(p)$ is a graph individual. The set of all paths $\mathbb{P}(\mathcal{A}')$ is defined inductively as follows:

- $[\frac{a}{a}] \in \mathbb{P}(\mathcal{A}')$ if a is a named individual and it occurs in $\mathcal{A}'$;
- $[p \mid \frac{s'}{s'}] \in \mathbb{P}(\mathcal{A}')$ if $p \in \mathbb{P}(\mathcal{A}')$, and s' is a successor of $\mathsf{tail}(p)$, it occurs in an assertion of $\mathcal{A}'$, and it is not blocked in $\mathcal{A}'$; and

- $[p \mid \frac{s}{s'}] \in \mathbb{P}(\mathcal{A}')$ if $p \in \mathbb{P}(\mathcal{A}')$, and s' is a successor of $\mathsf{tail}(p)$, it occurs in an assertion of $\mathcal{A}'$, and it is directly blocked in $\mathcal{A}'$ by s.

The following property, which we denote with (*), follows immediately from the previous definition: for each blocking-relevant concept C and each path $p \in \mathbb{P}(\mathcal{A}')$, the individual $\mathsf{tail}(p)$ is not blocked in $\mathcal{A}'$, so $C(\mathsf{tail}(p)) \in \mathcal{A}'$ if and only if $C(\mathsf{tail}'(p)) \in \mathcal{A}'$.

Let I be the following interpretation, where A is an atomic concept, R is an atomic (tree or graph) role, G is a description graph with ℓ vertices, $p_{(i)}$ are paths from $\mathbb{P}(\mathcal{A}')$ and are all from the same graph cluster, and a is a named individual.

$$
\begin{aligned}
\triangle^I &= \mathbb{P}(\mathcal{A}') \\
a^I &= [\tfrac{a}{a}] \text{ if } a \text{ occurs in } \mathcal{A}' \\
a^I &= b^I \text{ if named individuals } a = c_0, c_1, \ldots, c_n = b \text{ exist such that each} \\
&\quad\ c_{i-1} \text{ was merged into } c_i \text{ in the derivation leading to } \mathcal{A}' \\
A^I &= \{p \mid A(\mathsf{tail}(p)) \in \mathcal{A}'\} \\
R^I &= \{\langle p_1, p_2\rangle \mid R(\mathsf{tail}(p_1), \mathsf{tail}(p_2)) \in \mathcal{A}'\} \cup \\
&\quad\ \{\langle p, [p \mid \tfrac{s}{s'}]\rangle \mid s' \text{ is a successor of } \mathsf{tail}(p) \text{ and } R(\mathsf{tail}(p), s') \in \mathcal{A}'\} \cup \\
&\quad\ \{\langle [p \mid \tfrac{s}{s'}], p\rangle \mid s' \text{ is a successor of } \mathsf{tail}(p) \text{ and } R(s', \mathsf{tail}(p)) \in \mathcal{A}'\} \\
G^I &= \{\langle p_1, \ldots, p_\ell]\rangle \mid G(\mathsf{tail}(p_1), \ldots, \mathsf{tail}(p_\ell)) \in \mathcal{A}'\}
\end{aligned}
$$

Since $\mathcal{A}$ is initial, the ABox $\mathcal{A}'$ contains at least one assertion, so $\triangle^I$ is not empty. We now show that, for each $p_s = [q_s \mid \frac{s}{s'}]$ and $p_t = [q_t \mid \frac{t}{t'}]$ from $\triangle^I$, the following claims hold (**):

- *If $s' \approx t' \in \mathcal{A}'$ and $q_s = q_t$, then $p_s = p_t$:* Since the $\approx$-rule is not applicable to $\mathcal{A}'$, we have $s' = t'$, which implies $p_s = p_t$.
- *If $s' \not\approx t' \in \mathcal{A}'$, then $p_s \neq p_t$:* Since the $\bot$-rule is not applicable to $s' \not\approx t'$, we have $s' \neq t'$, which implies $p_s \neq p_t$.
- *If $A(s') \in \mathcal{A}'$, then $p_s \in A^I$:* By (*), we have $A(s) \in \mathcal{A}'$, so $p_s \in A^I$.
- *If $\neg A(s') \in \mathcal{A}'$, then $p_s \notin A^I$.* Since the $\bot$-rule is not applicable to $\neg A(s')$, we have $A(s') \notin \mathcal{A}'$. By (*), this implies $A(s) \notin \mathcal{A}'$, so $p_s \notin A^I$.
- *If $\geq n\, R.C(s') \in \mathcal{A}'$, then $p_s \in (\geq n\, R.C)^I$:* Note first that R is a tree role, since only such roles can occur in concepts of the form $\geq n\, R.C$. By (*), $\geq n\, R.C(s) \in \mathcal{A}'$ and s is not blocked. The $\geq$-rule is not applicable to $\geq n\, R.C(s)$, so individuals $u_1, \ldots, u_n$ exist such that $\mathsf{ar}(R, s, u_i) \in \mathcal{A}'$ and $C(u_i) \in \mathcal{A}'$ for $1 \leq i \leq n$, and $u_i \not\approx u_j \in \mathcal{A}'$ for $1 \leq i < j \leq n$. By property (1) of Lemma 4, the following possibilities exist for each u_i:

– u_i can be a successor of s. If u_i is directly blocked by u'_i, then let $p_{u_i} = [p_s \mid \frac{u'_i}{u_i}]$; otherwise, let $p_{u_i} = [p_s \mid \frac{u_i}{u_i}]$.
– u_i can be a predecessor of s. Let $p_{u_i} = q_s$. If $\mathsf{tail}'(p_{u_i}) \neq u_i$, this is because s' is blocked, but then, by the conditions of blocking, $C(\mathsf{tail}'(p_{u_i})) \in \mathcal{A}'$ and $\mathsf{ar}(R, s', \mathsf{tail}'(p_{u_i})) \in \mathcal{A}'$.
– u_i can be neither a predecessor nor a successor of s. Then, both s and u_i are named individuals, so let $p_{u_i} = [\frac{u_i}{u_i}]$.

In all cases $\mathsf{ar}(R, s', \mathsf{tail}'(p_{u_i})) \in \mathcal{A}'$, which implies $\langle p_s, p_{u_i} \rangle \in R^I$, and $C(\mathsf{tail}'(p_{u_i})) \in \mathcal{A}'$, which implies $p_{u_i} \in C^I$. Consider now each pair of paths p_{u_i} and p_{u_j} with $i \neq j$. If $\mathsf{tail}'(p_{u_i}) \not\approx \mathsf{tail}'(p_{u_j}) \in \mathcal{A}'$, then clearly $p_{u_i} \neq p_{u_j}$. If $\mathsf{tail}'(p_{u_i}) \not\approx \mathsf{tail}'(p_{u_j}) \notin \mathcal{A}'$, this is because $\mathsf{tail}'(p_{u_i}) \neq u_i$, which is possible only if s' is directly blocked by s and u_i is a predecessor of s. Since s can have at most one predecessor, no u_j with $j \neq i$ is a predecessor of s, so $p_{u_i} \neq p_{u_j}$. Thus, $p_s \in (\geq n\, R.C)^I$.

- *If $\exists G|_i(s') \in \mathcal{A}'$, then $p_s \in (\exists G|_i)^I$:* By (*), $\exists G|_i(s) \in \mathcal{A}'$ and s is not blocked. The $\exists G$-rule is not applicable to $\exists G|_i(s)$, so individuals u_j exist such that $G(u_1, \ldots, u_{i-1}, s, u_{i+1}, \ldots, u_\ell) \in \mathcal{A}'$. By Lemma 4, all u_j are from the same graph cluster as s. If s is a graph individual, since it is not blocked, its tree predecessor is not blocked either; otherwise, s is a nonblocked predecessor of all u_j. Either way, none of the individuals u_j are blocked. But then, by the definition of I, we have $\langle u_1^I, \ldots, u_\ell^I \rangle \in (\exists G|_i)^I$, so $p_s \in (\exists G|_i)^I$.

Property (**) implies that $I \models \alpha'$ for each assertion $\alpha' \in \mathcal{A}'$ that contains only named individuals. Consider now each assertion $\alpha \in \mathcal{A}$. If $\alpha \notin \mathcal{A}'$, then some named individuals in α were merged into other individuals; but then, $\mathcal{A}'$ contains the assertion α' obtained by this merging, so $I \models \alpha$ by the definition of I. Hence, $I \models \mathcal{A}$, and it remains to be shown that $I \models \mathcal{R}$.

Consider first each tree-like rule $r \in \mathcal{R}_t$ and each mapping μ of variables of r to objects of $\triangle^I$. By Definition 10, the rule is of the following form, where R_i and S_i are (not necessarily atomic) tree roles.

$$\bigwedge A_i(x) \wedge \bigwedge \mathsf{ar}(R_i, x, y_i) \wedge \bigwedge B_i(y_i) \rightarrow \bigvee C_i(x) \vee \bigvee D_i(y_i) \vee \bigvee \mathsf{ar}(S_i, x, y_i) \vee \bigvee y_i \approx y_j$$

Let $p_x = \mu(x)$, $p_{y_i} = \mu(y_i)$, and $s' = \mathsf{tail}'(p_x)$. Assume now that each atom from the antecedent of r is true in I and μ—that is, $p_x \in A_i^I$, $p_{y_i} \in B_i^I$, and $\langle p_x, p_{y_i} \rangle \in R_i^I$.

If s' is not blocked, let $s = s'$ and $t_i = \mathsf{tail}'(p_{y_i})$. By the definition of I, we have $A_i(s) \in \mathcal{A}'$, $B_i(t_i) \in \mathcal{A}'$, and $\mathsf{ar}(R_i, s, t_i) \in \mathcal{A}'$.

If s' is blocked, let $s = \mathsf{tail}(p_x)$; that is, s is the individual that blocks s'. By the definition of I, since $p_x \in A_i^I$, we have $A_i(s) \in \mathcal{A}'$. If $\mathsf{tail}'(p_{y_i})$

is a successor of s, let $t_i = \mathsf{tail}'(p_{y_i})$; now $p_{y_i} \in B_i^I$ and $\langle p_x, p_{y_i}\rangle \in R_i^I$ imply $B_i(t_i) \in \mathcal{A}'$ and $\mathsf{ar}(R_i, s, t_i) \in \mathcal{A}'$. If $\mathsf{tail}'(p_{y_i})$ is not a successor of s, let t_i be the predecessor of s; this predecessor exists by the definition of blocking. Furthermore, $p_{y_i} \in B_i^I$ and $\langle p_x, p_{y_i}\rangle \in R_i^I$ imply $B_i(\mathsf{tail}'(p_{y_i})) \in \mathcal{A}'$ and $\mathsf{ar}(R_i, s', \mathsf{tail}'(p_{y_i})) \in \mathcal{A}'$; by the definition of blocking, we have $B_i(t_i) \in \mathcal{A}'$ and $\mathsf{ar}(R_i, s, t_i) \in \mathcal{A}'$ as well.

Let σ be a mapping such that $\sigma(x) = s$ and $\sigma(y_i) = t_i$. The *Hyp*-rule is not applicable to r and $\mathcal{A}'$, so some of the atoms from the consequent of $\sigma(r)$ are present in $\mathcal{A}'$. Assume first that $C_i(s) \in \mathcal{A}'$, $D_i(t_i) \in \mathcal{A}'$, or $\mathsf{ar}(S_i, s, t_i) \in \mathcal{A}'$. By the definition of blocking, then $C_i(\mathsf{tail}'(p_x)) \in \mathcal{A}'$, $D_i(\mathsf{tail}'(p_{y_i})) \in \mathcal{A}'$, or $\mathsf{ar}(S_i, \mathsf{tail}'(p_x), \mathsf{tail}'(p_{y_i})) \in \mathcal{A}'$, respectively; by (**), this implies $p_x \in C_i^I$, $p_{y_i} \in D_i^I$, or $\langle p_x, p_y\rangle \in S_i^I$,respectively, so $I, \mu \models r$. Assume now that $t_i \approx t_j \in \mathcal{A}'$. If p_{y_i} and p_{y_j} are both predecessors of p_x, then $p_{y_i} = p_{y_j}$ so $I, \mu \models r$. If p_{y_i} is a predecessor and p_{y_j} is a successor of p_x, then $\mathsf{tail}'(p_{y_j}) = t_j$; furthermore, since t_i is not blocked, $\mathsf{tail}'(p_{y_j}) \neq t_i$, which contradicts the assumption that $t_i = t_j$. Finally, if both p_{y_i} and p_{y_j} are successors of p_x, then $p_{y_i} = p_{y_j}$ by (**), so $I, \mu \models r$.

Consider now $r \in \mathcal{R}_g$ to be a graph-regular rule of the form

$$A_1 \wedge \ldots \wedge A_n \rightarrow B_1 \vee \ldots \vee B_m$$

and let μ be a variable mapping such that $I, \mu \models A_i$ for $1 \leq i \leq n$. For each variable x, let $p_x = \mu(x)$ and $\sigma(x) = \mathsf{tail}(p_x)$. By the definition of I, none of $\sigma(x)$ is blocked, so $\sigma(A_i) \in \mathcal{A}'$. Since the *Hyp*-rule is not applicable to r and $\mathcal{A}'$, we have $\sigma(B_j) \in \mathcal{A}'$ for some $1 \leq j \leq m$. But then, by the definition of I, we have $I, \mu \models B_j$, so $I, \mu \models r$. □

We next prove that our calculus terminates. Using the standard argument, we show that each tree individual can have exponentially many tree predecessors. Thus, the only thing that might prevent the calculus from terminating is a situation described after Definition 4: the calculus might create infinitely many instances of G. The $\exists G$-rule, however, is applied with the lowest priority. Thus, if we instantiate G twice, before any further application of the $\exists G$-rule, either the rules from item 1 of Definition 9 will have merged these two instances of G, or the rules from item 2 of Definition 9 will have detected a clash. Thus, these rules and the rule priority guarantee that we never have more than two instances of G in the same graph cluster, which can be used to prove termination.

Lemma 7 (Termination)**.** *Let $\mathcal{R}$ be an admissible set of rules and $\mathcal{A}$ an initial ABox. Then, each derivation for $\mathcal{R}$ and $\mathcal{A}$ is finite.*

Proof. We prove the claim by showing that *(i)* each individual can cause only a limited number of rule applications and *(ii)* the number of new individuals introduced on each path of a derivation is finite.

We first prove *(i)*—that is, that each derivation rule can be applied a limited number of times to a fixed set of individuals on every path of every derivation. Observe that the supply of individuals is infinite; hence, if an individual s is pruned in some ABox in the derivation, we can assume that s is not introduced later in the derivation. We now show that (i) holds for each derivation rule.

- An application of the *Hyp*-rule to a rule $r \in R$ and a mapping σ introduces an assertion $\sigma(V_i)$ for some atom V_i from the consequent of r, which prevents repeated application of the *Hyp*-rule to the same r and σ. Merging and pruning can remove $\sigma(V_i)$ in subsequent derivation steps, but this would also remove some individual in σ, thus preventing the use of the same σ in future.

- An application of the $\geq$-rule to $\geq n\, R.C(s)$ extends the ABox with assertions $C(t_1), \ldots, C(t_n)$, $\mathsf{ar}(R, s, t_1), \ldots, \mathsf{ar}(R, s, t_n)$, and $t_i \not\approx t_j$ for $1 \leq i < j \leq n$. Thus, the individuals $u_1, \ldots, u_n$ from the precondition of the $\geq$-rule can be matched to $t_1, \ldots, t_n$, which prevents future applications of the $\geq$-rule to $\geq n\, R.C(s)$. If some t_j is merged into an individual v, then the assertions $C(v)$, $\mathsf{ar}(R, s, v)$, and $v \not\approx t_k$ are added to the ABox, so s retains a set of a neighbors which prevents subsequent application of the $\geq$-rule to $\geq n\, R.C(s)$.

- An application of the $\exists G$-rule to $\exists G|_i(s)$ extends the ABox with an assertion $G(t_1, \ldots, t_{i-1}, s, t_{i+1}, \ldots, t_\ell)$. Thus, the individuals u_j from the precondition of the the $\exists G$-rule can be matched to t_j, which prevents future applications of the $\exists G$-rule to $\exists G|_i(s)$. If some t_j is merged into an individual v, then the appropriate graph assertion is added to the ABox that prevents subsequent application of the $\exists G$-rule to $\exists G|_i(s)$.

- The $\approx$-rule is never applied twice to the same assertion $s \approx t$ since either s or t is removed from the ABox.

- If the $\bot$-rule is applied, then the resulting ABox labels a leaf of the derivation.

Next we prove *(ii)*—that is, that the total number of individuals introduced on a derivation path is finite.

Let c be the number of concepts and r the number of atomic roles that occur in $\mathcal{R}$ and $\mathcal{A}$. For an individual s and its predecessor t, the number of different labels $\mathcal{L}_\mathcal{A}(s)$ and $\mathcal{L}_\mathcal{A}(t)$ is 2^c, and the number of different labels $\mathcal{L}_\mathcal{A}(s, t)$ and $\mathcal{L}_\mathcal{A}(t, s)$ is 2^r. Hence, if there are more than $\delta = 2^{2c+2r} + 1$ such pairs of individuals, at least two pairs have the same labels; we denote this observation with (*). By Lemma 4, each tree individual u has a link to root of some length n. Since the ordering $\prec$ used in the definition of blocking contains the predecessor relationship and because of (*), u is blocked if

$n = \delta$. Since $\geq$- and $\exists G$-rules are not applied to blocked individuals, each individual u in $\mathcal{A}'$ has at most δ tree predecessors; since each tree individual has at most one graph predecessor, u can have at most 2δ predecessors. Furthermore, the number of the applications of the $\geq$-rule to each individual with k predecessors is bounded by (i), which limits the overall number of tree individuals with $k+1$ predecessors introduced in a derivation.

To complete the proof, we show that, for each tree individual s, the number of graph individuals occurring in $\mathcal{A}'$ that are from the same graph cluster as s is bounded. The set of rules $\mathcal{R}$ is admissible, so it contains all the rules from items 1 and 2 of Definition 9. Furthermore, the *Hyp*-, $\bot$-, and $\approx$-rules are applied with higher priority than the $\exists G$-rule. Therefore, whenever an ABox $\mathcal{A}'$ in a derivation contains graph assertions $G(t_1, \ldots, t_\ell)$ and $G(t'_1, \ldots, t'_\ell)$ such that $t_i = t'_j$, the applications of the *Hyp*-, $\bot$-, and $\approx$-rule will either derive $\bot$ or will merge one graph assertion into the other. Thus, whenever the $\exists G$-rule is applicable to some assertion $\exists G|_i(t)$ with t from the same graph cluster as s, the ABox $\mathcal{A}'$ contains at most one instance of G containing t. The $\exists G$-rule can be applied to each of the vertices in that instance, so the number of graph individuals from the same graph cluster as s is at most ℓ^2. Similar reasoning applies if s is a named individual. But then, exactly as in the previous paragraph, we conclude that the total number of graph individuals is bounded.

The total number of individuals introduced in a derivation is bounded, and the number of applications of derivation rules to each individual is bounded as well, which gives us a bound on the maximum length in a derivation. Since each derivation is finitely branching, it is finite as well. □

The following theorem follows immediately from Lemmas 1, 2, 3, 5, 6, and 7.

Theorem 1. *Let $\mathcal{K} = (\mathcal{T}, G, \mathcal{P}, \mathcal{A})$ be a graph-extended DL knowledge base where $\mathcal{T}$ is a $\mathcal{SHIQ}$ TBox and $\mathcal{A}$ is an initial ABox; furthermore, let $\mathcal{R} = \Xi(\mathcal{T}) \cup \Xi(G) \cup \mathcal{P}$.*

- *If $\mathcal{K}$ is satisfiable, then each derivation for $\mathcal{R}$ and $\mathcal{A}$ is clash-free.*
- *If a clash-free derivation for $\mathcal{R}$ and $\mathcal{A}$ exists, then $\mathcal{K}$ is satisfiable.*
- *Each derivation for $\mathcal{R}$ and $\mathcal{A}$ is finite.*

6 Transforming OWL Ontologies into Graphs

The evaluation of the adequacy of our approach is rather difficult due to lack of adequate test data. Furthermore, remodeling existing ontologies using a new modeling paradigm may require considerable effort. In order to both obtain test data for our reasoner and make the adoption of our approach in

practice easier, we have developed an algorithm that automatically transforms a TBox $\mathcal{T}$ into a graph-extended knowledge base $\mathcal{K}$. For example, our algorithm can automatically construct the graph shown in Figure 1 from the axioms such as (2)–(4). Clearly, the resulting graph-extended knowledge base can only be taken as a rough approximation; however, it can be used as a starting point for a more comprehensive remodeling of $\mathcal{T}$ into a proper graph-extended knowledge base. We applied our algorithm to GALEN and FMA, and domain experts have assured us that the resulting description graph correctly reflects many aspects of human anatomy.

In Section 6.1 we describe the intuition behind our algorithm and present the algorithm's pseudo-code. Then, in Section 6.2 we discuss the results we got by applying the algorithm to GALEN and FMA.

6.1 The Transformation Algorithm

Our transformation of a TBox $\mathcal{T}_1$ into a graph-extended knowledge base $\mathcal{K} = (\mathcal{T}, G, \mathcal{P}, \mathcal{A})$ is based on two assumptions.

The first assumption is that only some concepts and roles from $\mathcal{T}_1$ are *relevant* for G. For example, the *Heart* concept is clearly relevant to the description graph of human anatomy; in contrast, the *Disease* concept is not relevant because it does not represent the structure of a human body. Similarly, the *hasStructuralComponent* role clearly belongs to the graph, while the *hasAge* role does not.

Our second assumption is that each concept relevant to G should be represented by one vertex in G, and that edges in G can be decoded from axioms of the form $A \sqsubseteq \exists R.B$. Our assumption is that, by writing axioms such as (2)–(4), modelers actually wanted to say "the aortic valve has an alpha connection to the left ventricle, and the left side of heart has *the same* left ventricle as a solid division."

We use these two assumptions in the core part of our algorithm, which is parameterized with a DL TBox $\mathcal{T}_1$, a set of relevant concepts N_{C_g}, and a set of relevant roles N_{R_g}. The latter set actually defines the set of graph roles, and all other roles are considered to be tree roles. Our algorithm first computes $\Delta(\mathcal{T}_1)$ and thus normalizes the input TBox; as in Section 5, this has the benefit of making all negations explicit. Then, the algorothm creates a vertex i in V for each concept $A \in N_{C_g}$ and sets $\lambda\langle i\rangle = \{A\}$. Then, it processes each axiom $\alpha \in \mathcal{T}_1$ as follows:

- If α is of the form $A \sqsubseteq \exists R.B$ where $\{A, B\} \subseteq N_{C_g}$ and $R \in N_{R_g}$, then, for i and j vertices such that $\lambda\langle i\rangle = \{A\}$ and $\lambda\langle j\rangle = \{B\}$, the algorithm adds the edge $\langle i, j\rangle$ to E and extends λ such that $R \in \lambda\langle i, j\rangle$.
- If α does not contain a role from N_{R_g}, the algorithm simply copies α to the resulting TBox $\mathcal{T}$.

- If α contains only roles from N_{R_g} and no existential quantifier, the algorithm translates α into a graph-regular rule and adds it to $\mathcal{P}$.
- If α is not of the above form, then either it involves a graph and a tree role simultaneously, or it is of the form $A \sqsubseteq \exists R.B$ but some of A, B, or R are not relevant for the graph. Such an axiom either invalidates the syntactic restrictions of our formalism or it does not have a natural interpretation; hence, our algorithm simply ignores such an axiom α.

Our translation cannot correctly handle axioms of the form $A \sqsubseteq \geq n\, R.B$ with $n \geq 2$. Intuitively, such axioms might be handled by creating n vertices in G, labeling all of them with B, and then connecting the vertex of A with all the vertices of B using the role R. The situation, however, is not so simple if, in addition, we also have the axiom $B \sqsubseteq \geq m\, R.A$. It is now not clear which vertices of the description graph labeled with A to "reuse" to satisfy this axiom. Therefore, we decided to ignore such axioms. This is partly justified by the fact that GALEN and FMA—our main sources of inspiration and test data—do not contain $\geq n\, R.B$ concepts with $n \geq 2$. In human anatomy, different objects of the domain are naturally given different names. For example, instead of an axiom

$$\mathit{Heart} \sqsubseteq \geq 2\, \mathit{hasStructuralComponent}.\mathit{SideOfHeart}, \tag{20}$$

GALEN introduces explicit names for the left and the right side of the heart:

$$\mathit{Heart} \sqsubseteq \exists \mathit{hasStructuralComponent}.\mathit{LeftSideOfHeart} \tag{21}$$

$$\mathit{Heart} \sqsubseteq \exists \mathit{hasStructuralComponent}.\mathit{RightSideOfHeart} \tag{22}$$

On ontologies with at-least restrictions, our algorithm simply treats each $\geq n\, R.B$ as $\exists R.B$. It is natural to use number restrictions for modeling symmetric organs such as the kidney. On such an ontology, our algorithm produces a description graph containing just one copy of the object, and the graph can then be corrected by the modeler.

Determining the sets N_{C_g} and N_{R_g} manually is not easy. According to our experience with GALEN and FMA, a good strategy is to manually identify a set of roles N'_{R_g} that naturally belong to the graph, and then to take N_{R_g} as the closure of N'_{R_g} w.r.t. the explicit role inclusions from $\mathcal{T}_1$. Then, we take N_{C_g} as the set of all concepts A and B occurring in an axiom $A \sqsubseteq \exists R.B \in \mathcal{T}_1$ such that $R \in N_{R_g}$. Intuitively, if A and B are connected by a role that should be included into the graph, then it is likely that A and B should be included into the graph as well.

This idea, however, requires some refinement. For example, GALEN contains the following axioms:

$$\mathit{LeftVentricle} \sqsubseteq \mathit{Ventricle} \tag{23}$$

$$\mathit{RightVentricle} \sqsubseteq \mathit{Ventricle} \tag{24}$$

Let us assume that N_{C_g} contains the concepts *Ventricle*, *LeftVentricle*, and *RightVentricle*. The core transformation then generates a description graph G containing three different vertices, each labeled with one of these concepts. It is, however, counterintuitive for G to contain a *Ventricle* vertex: no ventricle as such exists on its own; rather, each concrete ventricle is either the left of the right ventricle. In fact, such a description graph G is unsatisfiable. Assume that an object x as instance of *LeftVentricle*; due to (23), x is also an instance of *Ventricle*. To satisfy the A-start property for *LeftVentricle*, x must correspond to the i-th vertex of some instance of G; similarly, to satisfy the A-start property for *Ventricle*, x must also correspond to the j-th vertex of some instance of G. Finally, because *LeftVentricle* and *Ventricle* label different vertices of G, we have $i \neq j$, which then invalidates the disjointness property of Definition 4. The concept *Ventricle* is thus an *abstract concept*: it is not meant to be instantiated directly, but only through a subclass. Such concepts clearly do not belong into a description graph. Hence, after computing N_{C_g} as described in the previous paragraph, our algorithm classifies the input TBox $\mathcal{T}_1$ using standard DL reasoning; then, it removes from N_{C_g} all concepts that are not leaves in the resulting classification. Intuitively, if A is not a leaf concept in the classification of $\mathcal{T}_1$, then A is likely to be an abstract concept, so it should not be added to G.

The pseudo-code of the transformation is shown in Algorithm 1. The algorithm is given a DL TBox $\mathcal{T}_1$ and a set of graph roles N'_{R_g}. For the same reasons as in Section 5, the algorithm first normalizes the input TBox axioms (line 2). Then, it closes the set N'_{R_g} according to the explicitly specified role inclusion relationships in $\mathcal{T}_1$ (line 3). Next, it computes the set N_{C_g} as outlined in the previous paragraphs (lines 4–9). The main part of the algorithm then processes all the TBox axioms as explained previously (lines 10–27). Finally, the algorithm outputs the resulting graph-extended knowledge base $\mathcal{K}$ (line 28). The transformation tool implementing this algorithm can be downloaded from HermiT's Web page.

6.2 Applying the Transformation to GALEN and FMA

We applied the algorithm from Section 6.1 to the original version of GALEN; furthermore, FMA is a very large ontology, so we have applied our algorithm to a fragment of FMA that describes the heart. Both ontologies can be downloaded from HermiT's Web page. Table 6 summarizes information about the original and the transformed ontologies.

Our transformation clearly leads to a change in the semantics of the ontology, and some information is lost in the process. Many parts of the resulting description graph, however, correspond with the intuitive descrip-

Algorithm 1 The Transformation Algorithm

procedure ToGraphExtendedKB($\mathcal{T}_1, N'_{R_g}$)
  $\mathcal{T}' \leftarrow \Delta(\mathcal{T}_1)$ ▷ Normalize $\mathcal{T}_1$
  $N_{R_g} \leftarrow \{S \mid \exists R \in N'_{R_g}$ such that $S \sqsubseteq^*_{\mathcal{T}_1} R\}$
  $N_{C_g} \leftarrow \emptyset$
  for each $\top \sqsubseteq \neg A \sqcup \exists R.B \in \mathcal{T}'$ where $R \in N_{R_g}$ **do**
    $N_{C_g} \leftarrow \{A, B\}$
  end for
  compute the subsumption hierarchy $\mathcal{H}(\mathcal{T}_1)$
  remove from N_{C_g} all nonleaf concepts from $\mathcal{H}(\mathcal{T}_1)$
  $V \leftarrow \emptyset, \quad E \leftarrow \emptyset, \quad \lambda \leftarrow \emptyset, \quad \mathcal{T} \leftarrow \emptyset, \quad \mathcal{P} \leftarrow \emptyset$
  for each $\alpha \in \mathcal{T}'$ **do**
    if $\alpha = \top \sqsubseteq \neg A \sqcup \exists R.B$ with $\{A, B\} \subseteq N_{C_g}$ and $R \in N_{R_g}$ **then**
      let i be the vertex of V such that $A \in \lambda\langle i \rangle$
      if no such i exists **then**
        $i \leftarrow 1 + |V|, \quad V \leftarrow V \cup \{i\}, \quad \lambda\langle i \rangle \leftarrow \{A\}$
      end if
      let j be the vertex of V such that $B \in \lambda\langle j \rangle$
      if no such j exists **then**
        $j \leftarrow 1 + |V|, \quad V \leftarrow V \cup \{j\}, \quad \lambda\langle j \rangle \leftarrow \{B\}$
      end if
      $E \leftarrow E \cup \{\langle i, j \rangle\}, \quad \lambda\langle i, j \rangle \leftarrow \lambda\langle i, j \rangle \cup \{R\}$
    else if α does not contain a role from N_{R_g} **then**
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{\alpha\}$
    else if α contains only roles from N_{R_g} and no $\exists R.B$ **then**
      $\mathcal{P} \leftarrow \mathcal{P} \cup \Xi(\alpha)$
    end if
  end for
  return $\mathcal{K} = (\mathcal{T}, G = (V, E), \mathcal{P}, \emptyset)$
end procedure

tions of the anatomy of the body. For example, the graph shown in Figure 1 has been extracted from the transformed ontology.

7 Evaluation and Discussion

To evaluate our approach, we have classified the original ontologies using HermiT, transformed them using the algorithm from Section 6 into graph-extended KBs, and classified the resulting KBs using the reasoning algorithm presented in Section 5. We now present the performance results and discuss the classification results.

Table 6: Information about Test Ontologies

	GALEN	FMA
Total number of concepts:	2748	430
Total number of roles:	413	38
Total number of GCIs:	6962	3479
GCIs discarded in the transformation:	320	328
With both a tree and a graph role:	74	0
With existentials on abstract concepts:	246	328
Translated GCIs:	6642	3151
Into the description graph:	680	2966
Into rules over the graph:	155	1
Into the DL TBox:	5807	184
With existentials on tree roles:	1741	16
With universals on tree roles:	952	0
Involving concept names only:	3114	168
Vertices in the description graph:	325	342
Edges in the description graph:	667	1076

7.1 Performance Results

We performed the experiments using a standard laptop with 1 GB of RAM. The classification of the original version of GALEN and the fragment of FMA took 129 s and 57 s, respectively; furthermore, the classification of the transformed ontologies took 781 s and 6 s, respectively.

The increase in the classification time for GALEN is partly due to the fact that our implementation of the reasoning algorithm in Section 5 is still very prototypical. In the case of FMA, the classification times are substantially lower because most of the original ontology is translated into the graph, so the generated models are much smaller.

Our performance results show that, even with a very prototypical implementation, we can process complex ontologies, which we take as indication that our approach is practically feasible.

7.2 Changes in the Semantics

The transformed ontologies are more constrained than the original ones, so we expect to obtain new entailments.

In the case of GALEN, we discovered a concept that is satisfiable in the original version of the ontology, but is unsatisfiable in the transformed ontology, which revealed a modeling error in GALEN. The problem occurs in the representation of the patella—a bone that is connected to certain ten-

dons through two retinacula; the retinacula are represented using the concepts *LateralPatellaRetinaculum* and *MedialPatellaRetinaculum*. GALEN describes the relationship between the patella and the retinacula as follows:

$$\mathit{LateralPatellaRetinaculum} \equiv \exists \mathit{hasOtherEndAt}.\mathit{Patella} \sqcap (\ldots) \tag{25}$$

$$\mathit{MedialPatellaRetinaculum} \equiv \exists \mathit{hasOtherEndAt}.\mathit{Patella} \sqcap (\ldots) \tag{26}$$

$$\mathit{hasOtherEndAt} \equiv \mathit{isAtOtherEndOf}^{-} \tag{27}$$

$$\top \sqsubseteq \leq 1\, \mathit{isAtOtherEndOf} \tag{28}$$

According to these axioms, each patella is connected to both the lateral and the medial retinacula, but due to functionality of *isAtOtherEndOf*, the two must be the same objects. Intuitively, this is an undesirable consequence, since the two retunaculae are in reality different objects; in other words, *isAtOtherEndOf* should probably not have been declared functional. Since GALEN is underconstrained, this does not cause the inconsistency of either concept, so this error has not been detected so far. The description graph produced by our transformation, however, contains one node for the patella and one for each retinaculum; furthermore, both retinacula are connected through *isAtOtherEndOf* to the same patella. Since *isAtOtherEndOf* is functional, the retinacula should be the same, which invalidates the disjointness property for the graph (see Definition 4) and makes *Patella* unsatisfiable.

In the case of FMA, we did not obtain any new subsumption relationships. This is due to the fact that most of the subsumption relationships in FMA are represented explicitly as axioms of the form $A \sqsubseteq B$ where A and B are atomic concepts. For example, the fact that the heart is an organ is represented explicitly with the axiom $\mathit{Heart} \sqsubseteq \mathit{Organ}$, and it is not derived from the structure of the heart; clearly, such inferences are performed in the same way on both tree-like and nontree structures.

As explained in Section 6, our algorithm discards some axioms from the ontology. We compared the class hierarchies of the original and the graph-extended versions of GALEN. In total, 361 subsumption relationships were lost, such as $\mathit{Femur} \sqsubseteq \mathit{BodySpace}$ (the femur is a body space), and

$$\mathit{InteratrialSeptum} \sqsubseteq \mathit{TwoAndAHalfDimensionalStructure}$$

(the interatrial septum of the heart is a structure with two and a half dimensions). All these entailments involve an abstract concept, so their loss is not surprising since the transformation algorithm discards GCIs that involve an abstract concept and an existential on a graph role. No information about concrete concepts has been lost, though.

In contrast, in the case of FMA we did not lose any subsumption relationships. As explained before, the reason is that the structural information in FMA largely does not influence subsumption.

7.3 Discussion

Our experience with GALEN and the discussions we had with the authors of GALEN lead us to conclude that our formalism represents the anatomical structures in the human body in a way that is closer to the modelers' intention than the original OWL axioms.[5] The fact that we found a modeling error in GALEN leads us to believe that our formalism and its semantics are based on "reasonable" assumptions.

Furthermore, capturing the semantics of abstract concepts and axioms involving them properly is likely to be the most important open problem. We briefly discuss possibilities for addressing it. Consider the following axiom in GALEN that is eliminated by the transformation algorithm because both *AtrioventricularValve* and *Ventricle* are abstract concepts:

$$AtrioventricularValve \sqsubseteq \exists hasAlphaConnection.Ventricle \tag{29}$$

Since both concepts in (29) are abstract, this axiom does not say anything about the structure of the concrete objects (i.e., the objects that are likely to be included into a description graph). Thus, one might expect the actual relationship between valves and ventricles to be described for the concrete subclasses of *AtrioventricularValve* and *Ventricle*. Axiom (29) can then be interpreted as a check which makes sure that this abstract relationship is concretized at a lower level. In [18], a formalism has been presented that might be useful for this purpose. Another possibility is to interpret *Ventricle* disjunctively over its subclasses: each valve is connected to either left or the right ventricle, but we do not know which. Currently, it is not clear which interpretation is appropriate; in fact, the proper interpretation of abstract concepts is made more difficult by the fact that whether a concept is abstract or concrete depends on the level of granularity.

8 Conclusion

We have extended OWL with description graphs, which can be used to describe structured objects—that is, objects consisting of parts connected in a complex, arbitrary way. We also allow for arbitrary SWRL-like rules over description graphs. Unlike most existing combinations of DLs and rules in which rules can be used only for query answering [16, 20, 23, 8, 19], our rules also fully participate in schema reasoning. Based on an observation that many structured objects exhibit a natural bound on their size, we derived a hypertableau reasoning algorithm for our formalism, which we implemented in the HermiT reasoner. To obtain suitable test data, we extracted description graphs out of GALEN and FMA medical terminologies. We successfully classified the resulting ontologies and even detected a modeling error.

[5] Thanks to Alan Rector and Sebastian Brandt.

We see three open problems for future research. First, graph-extended KBs should provide for several and not just one description graph, as this would allow breaking up a large graph into several more manageable parts. The main challenge is to identify an appropriate paradigm for specifying relationships between different description graphs. Second, an adequate semantics for modeling abstract concepts at different levels of granularity is needed. Third, to allow for a wider users' community, we would need to extend ontology editors such as Protégé with description graphs.

Acknowledgments

We than Alan Rector and Sebastian Brandt for providing us with invaluable comments from the users' perspective.

References

[1] A. Artale, E. Franconi, N. Guarino, and L. Pazzi. Part-whole relations in object-centered systems: An overview. *Data Knowledge & Engineering*, 20(3):347–383, 1996.

[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2nd edition, August 2007.

[3] F. Baader, C. Lutz, H. Sturm, and F. Wolter. Fusions of Description Logics and Abstract Description Systems. *Journal of Artificial Intelligence Research*, 16:1–58, 2002.

[4] D. Calvanese, G. De Giacomo, and M. Lenzerini. Structured Objects: Modeling and Reasoning. In T. W. Ling, A. O. Mendelzon, and L. Vieille, editors, *Proc. of the 4th Int. Conf. on Deductive and Object-Oriented Databases (DOOD '95)*, volume 1013 of *LNCS*, pages 229–246, Singapore, December 4–7 1995. Springer.

[5] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the Decidability of Query Containment under Constraints. In *Proc. of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '98)*, pages 149–158, Seattle, WA, USA, June 1–3 1998. ACM Press.

[6] S. Derriere, A. Richard, and A. Preite-Martinez. An Ontology of Astronomical Object Types for the Virtual Observatory. In *Proc. of the 26th meeting of the IAU: Virtual Observatory in Action: New Science,*

New Technology, and Next Generation Facilities, pages 17–18, Prague, Czech Republic, August 21–22 2006.

[7] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. AL-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.

[8] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining Answer Set Programming with Description Logics for the Semantic Web. In D. Dubois, C. A. Welty, and M.-A. Williams, editors, *Proc. of the 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 141–151, Whistler, Canada, June 2–5, 2004 2004. AAAI Press.

[9] C. Golbreich, S. Zhang, and O. Bodenreider. The Foundational Model of Anatomy in OWL: Experience and Perspectives. *Journal of Web Semantics*, 4(3):181–195, 2006.

[10] J. Goodwin. Experiences of using OWL at the Ordnance Survey. In *Proc. of the OWL: Expreiences and Directions Workshop (OWLED 2005)*, volume 188 of *CEUR WS Proceedings*, Galway, Ireland, November 11–12 2005.

[11] F. W. Hartel, S. de Coronado, R. Dionne, G. Fragoso, and J. Golbeck. Modeling a description logic vocabulary for cancer research. *Journal of Biomedical Informatics*, 38(2):114–129, 2005.

[12] I. Horrocks and P. F. Patel-Schneider. A Proposal for an OWL Rules Language. In *Proc. of the 13th Int. World Wide Web Conference (WWW 2004)*, pages 723–731, New York, NY, USA, May 17–22 2004. ACM Press.

[13] I. Horrocks and U. Sattler. A Tableaux Decision Procedure for $\mathcal{SHOIQ}$. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 448–453, Edinburgh, UK, July 30–August 5 2005. Morgan Kaufmann Publishers.

[14] I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8(3):239–263, 2000.

[15] O. Kutz, I. Horrocks, and U. Sattler. The Even More Irresistible SROIQ. In P. Doherty, J. Mylopoulos, and C. A. Welty, editors, *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 68–78, Lake District, UK, June 2–5 2006. AAAI Press.

[16] A. Y. Levy and M.-C. Rousset. Combining Horn Rules and Description Logics in CARIN. *Artificial Intelligence*, 104(1–2):165–209, 1998.

[17] B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Univesität Karlsruhe, Germany, 2006.

[18] B. Motik, I. Horrocks, and U. Sattler. Bridging the Gap Between OWL and Relational Databases. In *Proc. of the 16th International World Wide Web Conference (WWW 2007)*, pages 807–816, Banff, AB, Canada, May 8–12 2007. ACM Press.

[19] B. Motik and R. Rosati. A Faithful Integration of Description Logics with Logic Programming. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007)*, pages 477–482, Hyderabad, India, January 6–12 2007. Morgan Kaufmann Publishers.

[20] B. Motik, U. Sattler, and R. Studer. Query Answering for OWL-DL with Rules. *Journal of Web Semantics*, 3(1):41–60, 2005.

[21] B. Motik, R. Shearer, and I. Horrocks. Optimized Reasoning in Description Logics using Hypertableaux. In F. 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.

[22] D. A. Plaisted and S. Greenbaum. A Structure-Preserving Clause Form Translation. *Journal of Symbolic Logic and Computation*, 2(3):293–304, 1986.

[23] R. Rosati. $\mathcal{DL} + log$: A Tight Integration of Description Logics and Disjunctive Datalog. In P. Doherty, J. Mylopoulos, and C. A. Welty, editors, *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 68–78, Lake District, UK, June 2–5 2006. AAAI Press.

[24] C. Rosse and J. V. L. Mejino. A reference ontology for biomedical informatics: the Foundational Model of Anatomy. *Journal of Biomedical Informatics*, 36:478–500, 2003.

[25] R. A. Schmidt and U. Hustadt. A Principle for Incorporating Axioms into the First-Order Translation of Modal Formulae. In F. Baader, editor, *Proc. of the 19th Int. Conf. on Automated Deduction (CADE-19)*, volume 2741 of *LNAI*, pages 412–426, Miami Beach, FL, USA, July 28–August 2 2003. Springer.

[26] J. Seidenberg and A. L. Rector. Representing Transitive Propagation in OWL. In D. W. Embley, A. Olivé, and S. Ram, editors, *Proc. of the 25th Int. Conf. on Conceptual Modeling (ER 2006)*, volume 4215 of *LNCS*, pages 255–266, Tucson, AZ, USA, November 6–9 2006. Springer.

[27] A. Sidhu, T. Dillon, E. Chang, and B. Singh Sidhu. Protein Ontology Development using OWL. In *Proc. of the OWL: Expreiences and Directions Workshop (OWLED 2005)*, volume 188 of *CEUR WS Proceedings*, Galway, Ireland, November 11–12 2005.

[28] D. Soergel, B. Lauser, A. Liang, F. Fisseha, J. Keizer, and S. Katz. Reengineering Thesauri for New Applications: The AGROVOC Example. *Journal of Digital Information*, 4(4), 2004.

[29] W.D. Solomon, A. Roberts, J. E. Rogers, C. J. Wroe C.J., and A. L. Rector. Having our cake and eating it too: How the GALEN Intermediate Representation reconciles internal complexity with users' requirements for appropriateness and simplicity. In *Proc. of the Annual Fall Symposium of American Medical Informatics Association*, pages 819–823, Los Angeles, CA, USA, November 4–8 2000.

[30] K. A Spackman. SNOMED RT and SNOMEDCT. Promise of an international clinical terminology. *M.D. Computing*, 17(6):29, 2000.

[31] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation.* PhD thesis, RWTH Aachen, Germany, 2001.

[32] M. Y. Vardi. Why Is Modal Logic So Robustly Decidable? In N. Immerman and P. Kolaitis, editors, *Proc. of a DIMACS Workshop on Descriptive Complexity and Finite Models*, volume 31 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 149–184, Princeton University, USA, January 14–17 1996. American Mathematical Society.