Representing Structured Objects using Description Graphs

Boris Motik, Bernardo Cuenca Grau, Ian Horrocks
University of Oxford
Oxford, UK

Ulrike Sattler
University of Manchester
Manchester, UK

Abstract

State-of-the-art ontology languages are often not sufficiently expressive to accurately represent domains consisting of objects connected in a complex way. As a possible remedy, in our previous work we have proposed an extension of ontology languages with *description graphs*. In this paper, we extend this formalism by allowing for multiple graphs that can be combined in complex ways, thus obtaining a powerful language for modeling structured objects. By imposing a particular *acyclicity* restriction on the relationships between the graphs, we ensure that checking satisfiability of knowledge bases expressed in our language is decidable. We also present a practical reasoning algorithm.

Introduction

Ontologies are currently used for conceptual modeling in a wide range of applications. The Web Ontology Language (OWL) is a commonly used ontology language, the formal underpinning of which is provided by description logics (DLs) (Baader *et al*. 2007). Most DLs are fragments of first-order logic that describe a domain using *concepts* (unary predicates), *roles* (binary predicates), and *individuals* (constants). DL axioms are organized into the schema (TBox) component that contains universal knowledge about the domain, and the data (ABox) component that contains facts. We assume the reader to be familiar with the syntax and semantics of standard DLs (Baader *et al*. 2007).

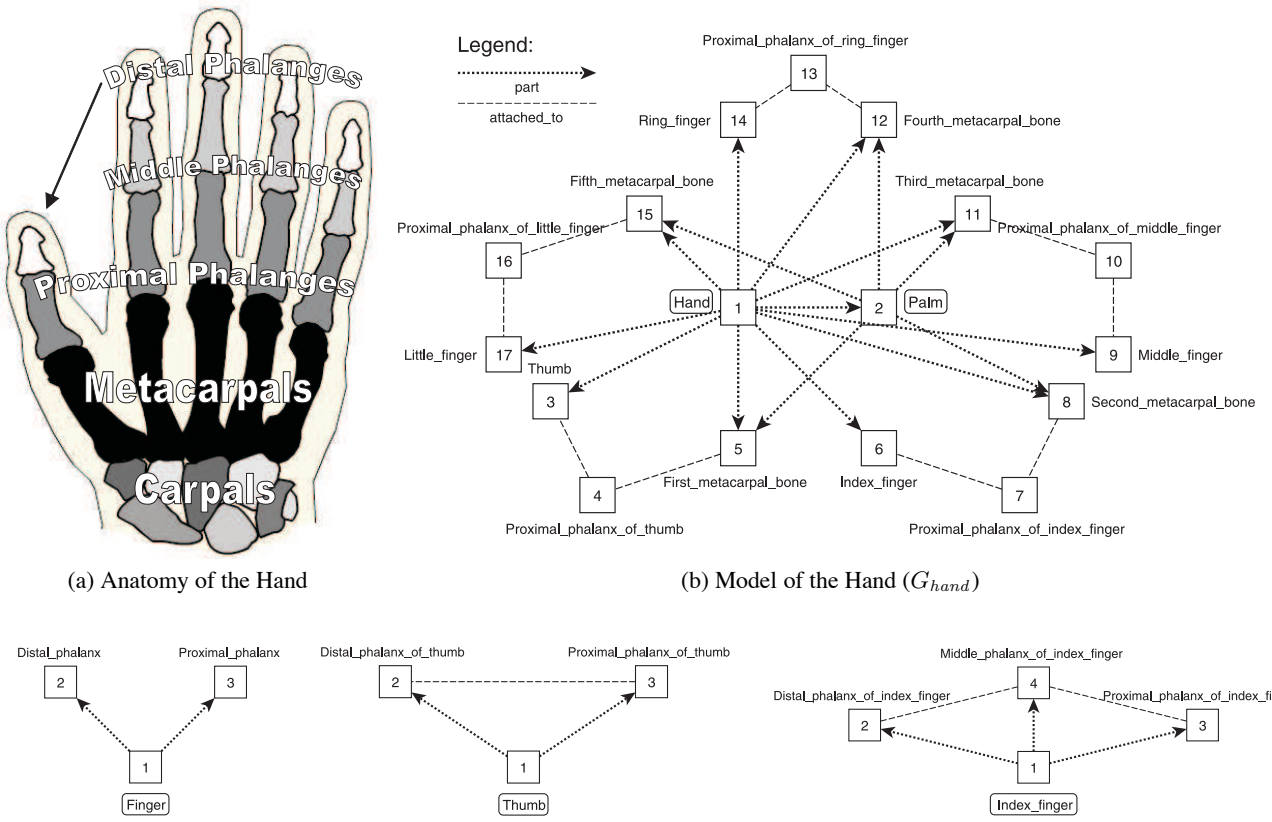
Ontologies often describe *structured objects*, which consist of many parts connected in complex ways. This is particularly the case in ontologies used in the clinical sciences, such as FMA (Rosse & Mejino 2003), GALEN (Rector, Nowlan, & Glowinski 1993), and SNOMED (Spackman 2000). For example, FMA models the human hand as consisting of the fingers, the palm, various bones, blood vessels, and so on, all of which are highly interconnected. The representation of such objects poses well-known problems to DLs, as DLs usually have a variant of the *tree model property* (Vardi 1996): each satisfiable DL knowledge base has a tree-like model. Thus, DLs cannot faithfully represent objects with nontree structures since they cannot enforce the existence of only non-tree-like models.

To address this problem, in our previous work we have proposed an extension of DLs with *description graphs* (Motik, Grau, & Sattler 2008), which can describe complex relations between objects in a direct and intuitive way. We have also shown that nontrivial ontologies can be semiautomatically remodeled as a DL KB extended with a description graph. To be able to focus on the core aspects of such an extension, however, we have made a number of simplifying assumptions: a knowledge base can contain only one description graph; this graph can neither specialize other axioms nor be specialized itself; and the roles in the DL axioms and the description graph must be strictly separated.

In this paper, we investigate possible ways of lifting these restrictions. We first present a general formalism that addresses all the limitations, but which is undecidable. We then identify a variant that allows for multiple graphs and graph specializations, but that requires the relationships between the graphs to satisfy a particular *acyclicity* condition. For the case where the roles are separated and the DL is $\mathcal{SHOQ}$, we provide a decision procedure based on hypertableau (Motik, Shearer, & Horrocks 2008). We believe that this formalism can support a range of practical applications; furthermore, the decision procedure can be easily extended to $\mathcal{SHOIQ}$ and thus cover all of OWL DL. Lifting the restriction on role separation, however, leads to undecidability if the DL provides for number restrictions (i.e., counting). For such cases, our algorithm can be modified to detect the inferences that can lead to nontermination and thus help users avoid "dangerous" knowledge bases.

Problems with Modeling Complex Structures

Consider the problem of modeling the skeleton of the human hand, shown in Figure 1a. The carpal bones form the base of the hand. The central part of the hand consists of the metacarpal bones, one leading to each finger. The fingers consist of phalanges: the proximal phalanges are connected to the metacarpal bones, and all fingers apart from the thumb contain a middle phalanx located between the proximal and the distal phalanx. This structure can be conceptualized as shown in Figures 1b–1e. Our goal is to describe this structure at the *schema level*, and thus obtain a "template" that can be instantiated for each particular hand. Thus, as discussed in our previous work, our description should be part of the TBox and not of the ABox.



(a) Anatomy of the Hand

(b) Model of the Hand (G_{hand})

(c) Model of a Finger (G_{finger})

(d) Model of the Thumb (G_{thumb})

(e) Model of the Index Finger (G_{index_finger})

Figure 1: The Anatomy of the Hand and its Models

Statements such as "the palm is a part of the hand" are typically represented in ontologies using DL axioms such as $Hand \sqsubseteq \exists part.Palm$. We have discussed in depth the limitations of such a style of modeling in our previous work. In short, most DLs enjoy a variant of the tree model property (Vardi 1996). Thus, as well as having a model that corresponds to the intended structure, DL axioms of the mentioned form also have an unintended model obtained by unraveling the intended structure into an infinite tree. This can prevent us from drawing conclusions that depend on the nontree connections in the structure; for example, if the thumb has a broken distal phalanx, then we should conclude that the phalanx adjacent to the proximal phalanx is broken (since this is the same broken phalanx). Furthermore, the unintended tree models can be large, which causes performance problems for reasoners that try to construct them.

Non-tree-like structures can be axiomatized using various extensions of DLs with rules (Levy & Rousset 1998); however, the schema-level integration of DLs with rules is undecidable even for basic DLs. The DL $\mathcal{SROIQ}$ (Kutz, Horrocks, & Sattler 2006) provides for complex role inclusions that can axiomatize a particular class of nontree structures; however, they cannot describe arbitrarily shaped structures.

In our previous work (Motik, Grau, & Sattler 2008), we have proposed to describe complex structures using *description graphs*, whose vertices and edges are labeled by concepts and roles, respectively. For example, Figure 1d is a description graph showing that each thumb has a proximal and a distal phalanx that are attached to each other.[1] Description graphs and DLs complement each other in expressive power: the former can be used to represent the structure of arbitrarily connected objects that are naturally bounded in size, whereas the latter can model possibly unbounded but tree-like structures. For example, up to a certain level of granularity, a human body can be decomposed into a finite number of subparts, the total number of which is naturally bounded by the decomposition; hence, we can represent the body using a description graph. In contrast, the statement that each person has two parents who are persons does not impose a natural bound on the number of people; hence, we can represent such relationships using DLs, provided the relationships are tree-shaped. To represent conditional aspects of the domain, we also allow for arbitrary first-order rules over the graph; for example, we can state that, if a bone in the hand is fractured, then the hand is fractured as well. This existing formalism, however, employs several simplifying assumptions that can limit its applicability.

[1] The role $attached\_to$ is symmetric, so we do not orient the edges labeled with it.

First, each knowledge base can contain only a single description graph. In our example, we would need to represent the hand and its fingers in a single graph, which might result in a description graph that is cluttered with detail and difficult to manage. In an extreme case, we would need to model the entire body as a single graph, which would clearly be cumbersome. Furthermore, reasoning with one monolithic graph can adversely affect the performance of reasoning as the reasoner must always consider the graph in its entirety.

Second, structured objects cannot be modeled at different levels of abstraction, which is often needed in practice. For example, we would like to describe the abstract structure common to all fingers as shown in Figure 1c, and then specialize the general structure for, say, the index finger and introduce the middle phalanx as shown in Figure 1e.

Third, our formalism requires the roles to be separated into tree and graph ones: the former can be used only in the DL axioms, whereas the latter can be used only in the graph and the rules. This requires users to decide in advance which parts of the domain will be modeled using graphs and which using DLs, and it prevents them from using the same role to represent both bounded and unbounded parts of the domain.

A Formalism for Complex Structures

We now present an extension of our previous work that addresses all three drawbacks outlined in the previous section. Let $\mathcal{DL}$ be a general DL language defined over a set of *atomic concepts* N_C, a set of *atomic roles* N_R, and a set of *named individuals* N_I. The set of *literal* concepts N_L is defined as $N_L = N_C \cup \{\neg A \mid A \in N_C\}$. A *TBox* $\mathcal{T}$ is a finite set of axioms expressed in $\mathcal{DL}$.

We start by extending the notion of a description graph.

Definition 1 (Description Graph)**.** *An ℓ-ary* description graph $G = (V, E, \lambda, M)$ *is a directed labeled graph where (*i*) $V = \{1, \ldots, \ell\}$ is a set of ℓ* vertices*, (*ii*) $E \subseteq V \times V$ is a set of* edges*, (*iii*) λ is a labeling function that assigns a set of literal concepts $\lambda\langle i\rangle \subseteq N_L$ to each vertex $i \in V$ and a set of atomic roles $\lambda\langle i, j\rangle \subseteq N_R$ to each edge $\langle i, j\rangle \in E$, and (*iv*) $M \subseteq N_C$ is a set of* main concepts *for G. For A an atomic concept, V_A is the set of vertices that contain A in their label; that is, $V_A = \{k \in V \mid A \in \lambda\langle k\rangle\}$.*

We define the vertices of G to be integers so that we can use them as indices. The main difference from the definition in our previous work is in the notion of a main concept. In Figure 1, main concepts are framed with rounded rectangles. Thus, the main concepts for the description graph in Figure 1b are $Hand$ and $Palm$, meaning that this graph defines the structure of the hand and the palm. Intuitively, an instance of a main concept implies the existence of a graph instance.

Definition 2 (Rule)**.** *Let N_V be a set of* variables *disjoint from N_I. An* atom *is an expression of the form $P(t_1, \ldots, t_k)$, where $t_i \in N_I \cup N_V$ and (*i*) P is an atomic concept and $k = 1$, or (*ii*) P is an atomic role and $k = 2$, or (*iii*) P is the equality predicate $\approx$ and $k = 2$, or (*iv*) P is an ℓ-ary graph G and $k = \ell$. An atom of the form $\approx(s, t)$ is written as $s \approx t$. A* rule *is an expression of the form* (1)*, where B_i and H_j are* body *and* head *atoms, respectively.*

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

W.l.o.g. we assume that the body does not contain $\approx$. Variables x and y are directly connected *in a rule r if they both occur in a body atom of r, and* connected *is the transitive closure of directly connected. A rule r is* connected *if each pair of variables x and y occurring in r is connected in r.*

Next, we introduce graph specializations to represent, for example, the fact that the graph for the thumb specializes the graph for the finger—that is, $G_{finger} \lhd G_{thumb}$.

Definition 3 (Graph Specialization)**.** *A* graph specialization *has the form $G_1 \lhd G_2$, for $G_1 = (V_1, E_1, \lambda_1, M_1)$ and $G_2 = (V_2, E_2, \lambda_2, M_2)$ description graphs with $V_1 \subseteq V_2$.*

Next, we introduce axioms that allow us to properly connect graph instances. For example, G_{hand} contains the vertices 3 and 4 that represent the thumb and its proximal phalanx, which correspond to the vertices 1 and 3 of G_{thumb}. We can specify this correspondence using a *graph alignment* of the form $G_{hand}[3, 4] \leftrightarrow G_{thumb}[1, 3]$. Intuitively, this ensures that it is not possible for G_{hand} and G_{thumb} to share the thumb without sharing the proximal phalanx as well.

Definition 4 (Graph Alignment)**.** *A* graph alignment *has the form $G_1[u_1, \ldots, u_n] \leftrightarrow G_2[w_1, \ldots w_n]$, where G_1 and G_2 are description graphs with sets of vertices V_1 and V_2, respectively, and $u_i \in V_1$ and $w_i \in V_2$ for $1 \leq i \leq n$.*

Finally, we define GBoxes and graph-extended KBs.

Definition 5 (Formalism)**.** *A* graph box *(GBox) is a tuple $\mathcal{G} = (\mathcal{G}_G, \mathcal{G}_S, \mathcal{G}_A)$ where $\mathcal{G}_G$, $\mathcal{G}_S$, and $\mathcal{G}_A$ are finite sets of description graphs, graph specializations over $\mathcal{G}_G$, and graph alignments over $\mathcal{G}_G$, respectively. An ABox is a finite set of* assertions *$C(a)$, $R(a_1, a_2)$, $a_1 \approx a_2$, $a_1 \not\approx a_2$, and $G(a_1, \ldots, a_\ell)$ (*graph assertion*), where $C \in N_L$, $R \in N_R$, $G \in \mathcal{G}_G$, and $a_{(i)} \in N_I$. A* graph-extended knowledge base *is a 4-tuple $\mathcal{K} = (\mathcal{T}, \mathcal{P}, \mathcal{G}, \mathcal{A})$ where $\mathcal{T}$ is a TBox, $\mathcal{P}$ is a program consisting of a finite number of connected rules, $\mathcal{G}$ is a GBox, and $\mathcal{A}$ is an ABox.*

Next, we define the semantics of the formalism.

Definition 6 (Semantics)**.** *An interpretation $I = (\triangle^I, \cdot^I)$ consists of a nonempty interpretation domain $\triangle^I$ and an interpretation function $\cdot^I$ that assigns to each atomic concept A, atomic role R, and ℓ-ary description graph G the sets $A^I \subseteq \triangle^I$, $R^I \subseteq \triangle^I \times \triangle^I$, and $G^I \subseteq (\triangle^I)^\ell$, respectively.*

We assume that $\mathcal{DL}$ defines a suitable notion of satisfaction of a TBox $\mathcal{T}$ in I, written $I \models \mathcal{T}$. Satisfaction of an ABox $\mathcal{A}$ in I, written $I \models \mathcal{A}$, is defined as usual. Satisfaction of a rule r in I, written $I \models r$, is defined by treating r as a universally quantified material implication. Satisfaction of a description graph, graph specialization, and graph alignment is defined in Table 1. A knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{P}, \mathcal{G}, \mathcal{A})$ is satisfied in I, written $I \models \mathcal{K}$, if all its components are satisfied in I.

Thus, each ℓ-ary graph G is interpreted as an ℓ-ary relation G^I in which each tuple corresponds to an instance of G. The key and disjointness properties ensure that no two distinct instances of G can share a vertex; for example, no two distinct instances of G_{hand} can share the vertex that represents the thumb. This assumption is required for decidability, and it seems reasonable in practical cases. The start

Table 1: Interpretation of GBox Elements

$I \models G$ for $G = (V, E, \lambda, M)$ an ℓ-ary graph if
Key property: $\forall x_1, \ldots, x_\ell, y_1, \ldots, y_\ell \in \triangle^I : \langle x_1, \ldots, x_\ell \rangle \in G^I \wedge \langle y_1, \ldots, y_\ell \rangle \in G^I \wedge \bigvee_{1 \le i \le \ell} x_i = y_i \rightarrow \bigwedge_{1 \le j \le \ell} x_j = y_j$
Disjointness property: $\forall x_1, \ldots, x_\ell, y_1, \ldots, y_\ell \in \triangle^I : \langle x_1, \ldots, x_\ell \rangle \in G^I \wedge \langle y_1, \ldots, y_\ell \rangle \in G^I \rightarrow \bigwedge_{1 \le i < j \le \ell} x_i \neq y_j$
Start property: for each atomic concept $A \in M$, $\forall x \in \triangle^I : x \in A^I \rightarrow \exists x_1, \ldots, x_\ell \in \triangle^I : \langle x_1, \ldots, x_\ell \rangle \in G^I \wedge \bigvee_{k \in V_A} x = x_k$
Layout property: $\forall x_1, \ldots, x_\ell \in \triangle^I : \langle x_1, \ldots, x_\ell \rangle \in G^I \rightarrow \bigwedge_{i \in V,\ B \in \lambda\langle i \rangle} x_i \in B^I \wedge \bigwedge_{\langle i,j \rangle \in E,\ R \in \lambda\langle i,j \rangle} \langle x_i, x_j \rangle \in R^I$
$I \models G_1 \lhd G_2$ for G_i an ℓ_i-ary description graph if $\forall x_1, \ldots, x_{\ell_2} \in \triangle^I : \langle x_1, \ldots, x_{\ell_1}, \ldots, x_{\ell_2} \rangle \in G_2^I \rightarrow \langle x_1, \ldots, x_{\ell_1} \rangle \in G_1^I$
$I \models G_1[u_1, \ldots, u_n] \leftrightarrow G_2[w_1, \ldots w_n]$ for G_i an ℓ_i-ary description graph if, for each $1 \le i \le n$, $\forall x_1, \ldots, x_{\ell_1}, y_1, \ldots, x_{\ell_2} \in \triangle^I : \langle x_1, \ldots, x_{\ell_1} \rangle \in G_1^I \wedge \langle y_1, \ldots, y_{\ell_2} \rangle \in G_2^I \wedge x_{u_i} = y_{w_i} \rightarrow \bigwedge_{1 \le j \le n} x_{u_j} = y_{w_j}$

property ensures that each instance of a main concept A of G occurs in an instance of G. For example, since *Hand* is a main concept for G_{hand}, each instance of *Hand* must occur as vertex 1 in an instance of G_{hand}. Similarly, vertex 3 of G_{hand} is labeled with *Thumb*, which is the main concept of G_{thumb}; hence, each vertex 3 in an instance of G_{hand} is also a vertex 1 in an instance of G_{thumb} (but not the other way around). The disjunction in the start property handles the case when a main concept labels multiple vertices. For example, if we were to describe the hand and the five fingers in a single graph without a distinction between the five fingers, then, given an instance of a *Finger*, we would have to guess which of the five fingers we are dealing with. Finally, the layout property ensures that each instance of G is labeled and connected as specified in the definition of G.

Graph specializations are interpreted as inclusions over the graph relations; for example, $G_{finger} \lhd G_{index\_finger}$ means that each instance of an index finger is also an instance of a finger. The two graphs share all the vertices of the more general graph, and the more specific graph can introduce additional vertices. This is not essential for our decidability results, but it simplifies the technical treatment.

Finally, graph alignments state that, whenever two graphs share some vertex from the specified list, then they share all other vertices from the list as well. For example, the alignment $G_{hand}[3, 4] \leftrightarrow G_{thumb}[1, 3]$ states that, if instances of G_{hand} and G_{thumb} share vertices 3 and 1, respectively, then they must also share vertices 4 and 3, respectively.

Note that our semantics of description graphs corresponds to implications of the form "if graph, then structure." In certain applications, however, the converse implication might be important in order to recognize graph instances in a structure; we call such inferences *graph recognition*. We do not explicitly support graph recognition because implications of the form "if structure, then graph" can be encoded in rules.

Decidability of Reasoning

The main reasoning problem for graph-extended KBs is satisfiability checking, as concept subsumption and instance checking can be reduced to satisfiability as usual. This problem is clearly undecidable: the combination of simple DLs with unrestricted Horn rules is already undecidable (Levy & Rousset 1998). Proposition 1 shows that, even without rules, the interaction between graphs and DL axioms leads to undecidability. The proposition can be proved in a simpler way and with $\mathcal{T}$ in $\mathcal{ALCF}$; however, the presented proof can be easily extended to acyclic GBoxes, defined shortly.

Proposition 1. *Checking satisfiability of $\mathcal{K} = (\mathcal{T}, \emptyset, \mathcal{G}, \emptyset)$ with $\mathcal{T}$ in $\mathcal{ALCIF}$ and $\mathcal{G} = (\mathcal{G}_G, \emptyset, \emptyset)$ is undecidable.*

Proof. Let $\mathcal{K}_{grid}$ be the following graph-extended KB. The GBox $\mathcal{G}$ contains the graphs $G_i = (V_i, E_i, \lambda_i, M_i)$, $1 \le i \le 4$, defined as follows. Each G_i contains nine vertices $V_i = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and the following labeled edges, where an edge between vertices i and j labeled with an atomic role R is represented as $i \xrightarrow{R} j$:

$$1 \xrightarrow{H} 2 \quad 2 \xrightarrow{H} 3 \quad 4 \xrightarrow{H} 5 \quad 5 \xrightarrow{H} 6 \quad 7 \xrightarrow{H} 8 \quad 8 \xrightarrow{H} 9$$
$$1 \xrightarrow{V} 4 \quad 4 \xrightarrow{V} 7 \quad 2 \xrightarrow{V} 5 \quad 5 \xrightarrow{V} 8 \quad 3 \xrightarrow{V} 6 \quad 6 \xrightarrow{V} 9$$

The labels of the following vertices of each G_i are not empty, and all other vertices are labeled with $\emptyset$:

$$\lambda_i\langle 2 \rangle = \{A_i\} \ \ \lambda_i\langle 4 \rangle = \{B_i\} \ \ \lambda_i\langle 3 \rangle = \{C_i\} \ \ \lambda_i\langle 7 \rangle = \{D_i\}$$

Finally, $M_i = \{A_i, B_i\}$. The $\mathcal{ALCIF}$ TBox $\mathcal{T}$ contains the following axioms:

$$\begin{array}{cccc} \top \sqsubseteq \le 1\, H & \top \sqsubseteq \le 1\, H^- & \top \sqsubseteq \le 1\, V & \top \sqsubseteq \le 1\, V^- \\ C_1 \sqsubseteq \exists H.A_2 & C_2 \sqsubseteq \exists H.A_1 & C_3 \sqsubseteq \exists H.A_4 & C_4 \sqsubseteq \exists H.A_3 \\ D_1 \sqsubseteq \exists V.B_3 & D_2 \sqsubseteq \exists V.B_4 & D_3 \sqsubseteq \exists V.B_1 & D_4 \sqsubseteq \exists V.B_2 \\ \top \sqsubseteq \exists R.A_1 & & & \end{array}$$

$\mathcal{K}_{grid}$ axiomatizes the existence of an infinite grid where horizontal and vertical links are represented using the roles H and V, respectively. By the last axiom in $\mathcal{T}$, the extension of A_1 is not empty, so an instance of G_1 exists in which vertices 3 and 7 is labeled with C_1 and D_1, respectively. By $C_1 \sqsubseteq \exists H.A_2$, vertex 3 of G_1 is connected with an instance of A_2, so an instance of G_2 exists. Thus, vertex 3 of G_1 is connected to vertex 2 of G_2 by H. Furthermore, vertex 1 of G_2 is also connected to vertex 2 of G_2 by H so, since H is inverse-functional, vertex 1 of G_2 must be the same as vertex 3 of G_1. But then, since V is functional, vertices 6 and 9 of G_1 must be the same as vertices 4 and 7 of G_2, respectively. Thus, instances of G_1 and G_2 are aligned into adjacent fragments of a grid. By applying the same argument inductively in the horizontal and vertical directions, one can see that the grid extends indefinitely in both directions.

For each instance of the undecidable DOMINO TILING problem (Börger, Grädel, & Gurevich 1996), $\mathcal{K}_{grid}$ can straightforwardly be extended with axioms that exactly encode the tiling of the grid, which implies our claim. □

Proposition 2 shows that, even without a DL TBox, the interaction between graphs and rules leads to undecidability.

Proposition 2. *Checking satisfiability of $\mathcal{K} = (\emptyset, \mathcal{P}, \mathcal{G}, \emptyset)$ with $\mathcal{P}$ a Horn program and $\mathcal{G} = (\mathcal{G}_G, \emptyset, \emptyset)$ is undecidable.*

Proof (Sketch). (Levy & Rousset 1998) have proved undecidability of the extension of a DL with rules by using a DL axiom to axiomatize the existence of an infinite R-chain and then encoding the HALTING problem using rules. Let $\mathcal{G}$ be a GBox containing the following description graphs:

$$G_1: \quad \begin{array}{rr} V_1 = \{1,2\} & M_1 = \{A_1\} \\ \lambda_1\langle 1\rangle = \{A_1\} & \lambda_1\langle 2\rangle = \{A_2\} \end{array} \quad 1 \xrightarrow{R} 2$$

$$G_2: \quad \begin{array}{rr} V_2 = \{1,2\} & M_2 = \{A_2\} \\ \lambda_2\langle 1\rangle = \{A_2\} & \lambda_2\langle 2\rangle = \{A_1\} \end{array} \quad 1 \xrightarrow{R} 2$$

If either A_1 or A_2 is not empty, $\mathcal{G}$ implies the existence of an infinite R-chain, which allows us to adapt the encoding by (Levy & Rousset 1998) with minor changes. □

We next explore ways of ensuring decidability. As the proof of Proposition 1 suggests, undecidability arises because, due to number restrictions, the structures whose existence is implied by DL axioms can interact with the structures whose existence is implied by description graphs. Definition 7 provides a way to restrict this interaction.

Definition 7 (Role-Separated KBs)**.** *A graph-extended KB $\mathcal{K} = (\mathcal{T}, \mathcal{G}, \mathcal{P}, \mathcal{A})$ is* role separated *if the set of atomic roles N_R can be split into disjoint subsets N_{R_t} and N_{R_g} of* tree *and* graph roles*, respectively, such that description graphs in $\mathcal{G}$ and rules in $\mathcal{P}$ refer only to graph roles, and axioms in $\mathcal{T}$ refer only to tree roles.*

In a role-separated knowledge base $\mathcal{K}$, the structures constructed using graphs and DLs are strictly separated. Therefore, if $\mathcal{K}$ is satisfiable, it has a model consisting of a *tree backbone* and *graph instances*: the former is a tree-shaped structure that is axiomatized using DL axioms, whereas the latter are arbitrarily connected fragments embedded into the backbone (Motik, Grau, & Sattler 2008).

Proposition 2 suggests that undecidability is also partly due to the fact that the GBox alone can axiomatize existence of an unbounded sequence of graphs. As we observed in our previous work, however, structured objects often exhibit a natural bound on their size. For example, the hand can be decomposed in a finite number of parts, each of which can be further finitely decomposed into subparts. Effectively, we obtain a hierarchy of parts, the leaves of which determine the total number of objects that we need to represent. This intuition suggests the following definition.

Definition 8 (Acyclic GBox)**.** *A GBox $\mathcal{G} = (\mathcal{G}_G, \mathcal{G}_S, \mathcal{G}_A)$ is* acyclic *if a strict (i.e., an irreflexive and transitive, but not necessarily total) order $\prec$ on $\mathcal{G}_G$ exists such that, for each $G = (V, E, \lambda, M)$ and $G' = (V', E', \lambda', M')$ in $\mathcal{G}_G$, if $G \not\preceq G'$, then, for each $A \in M'$ and $\lhd$ the reflexive–transitive closure of $\lhd$ in $\mathcal{G}_S$,*

- *if $G' \lhd G$, then $\neg A \in \lambda\langle i\rangle$ for each $i \in V \setminus V'$;*
- *if $G' \not\lhd G$, then $\neg A \in \lambda\langle i\rangle$ for each $i \in V$.*

Intuitively, $G_1 \prec G_2$ means that G_2 is subordinate to G_1. In our example, we would have $G_{hand} \prec G_{finger}$ and $G_{hand} \prec G_{thumb}$, since the structures of the finger and the thumb are subordinate to the structure of a hand, respectively. We would also have $G_{finger} \prec G_{thumb}$, since a finger is more general than the thumb. The conditions in Definition 8 state that, if G_2 is subordinate to G_1, then the existence of G_2 cannot imply the existence of G_1. For example, since the thumb is subordinate to the hand, no vertex in an instance of G_{thumb} should ever become labeled with a main concept of G_{hand} and thus imply a cycle.

Adding $\neg Hand$ and $\neg Palm$ to all vertices of G_{thumb} can be tedious and impractical. The problem can be addressed in practice by letting users specify the graph hierarchy $\prec$ in an ontology editor, which would then generate the required negative assertions automatically.

The proof of Proposition 1 holds even if $\mathcal{G}$ is made acyclic by adding $\neg A_i$ and $\neg B_i$, $1 \leq i \leq 4$ to each vertex of a graph G_j with $j \neq i$. This suggests that the interaction between number restrictions and graphs is a fundamental problem. Therefore, in the following section we present a reasoning algorithm for acyclic role-separated graph-extended KBs.

We also show that, if $\mathcal{DL}$ does not allow for number restrictions and transitivity, our algorithm provides a decision procedure for *rule separated* KBs (c.f. Definition 9), in which $\mathcal{T}$ and $\mathcal{G}$ can share roles, provided that $\mathcal{G}$ is acyclic.

Definition 9 (Rule-Separated KBs)**.** *A graph-extended KB $\mathcal{K} = (\mathcal{T}, \mathcal{G}, \mathcal{P}, \mathcal{A})$ is* rule separated *if the set of atomic roles N_R can be split into disjoint subsets $N_{R_{DL}}$ and N_{R_r} of* DL-roles *and* rule roles*, respectively, such that $\mathcal{T}$ refers only to DL-roles, and $\mathcal{P}$ refers only to rule roles.*

Finally, if no weakening of $\mathcal{DL}$ is acceptable, our algorithm can be used as a semidecision procedure. Moreover, the algorithm can notify the user when it detects an interaction between the tree backbone and graph instances, thus signalling that no termination guarantee can be given.

Reasoning Algorithm

In our previous work, we have presented a satisfiability checking algorithm for graph-extended KBs with a single description graph and with $\mathcal{DL}$ being $\mathcal{SHIQ}$. The DL underpinning OWL, however, is $\mathcal{SHOIQ}$—an extension of $\mathcal{SHIQ}$ with singleton concepts called *nominals*. The hypertableau algorithm for $\mathcal{SHOIQ}$ without description graphs (Motik, Shearer, & Horrocks 2008) is technically involved. Therefore, we focus here on the case when $\mathcal{DL}$ is $\mathcal{SHOQ}$, as this allows us to discuss the novel aspects due to multiple graphs and nominals without overloading the presentation with technical detail. This algorithm can be easily extended to $\mathcal{SHOIQ}$ by combining the mentioned existing results.

Role-Separated Acyclic KBs

Let $\mathcal{K} = (\mathcal{T}, \mathcal{P}, \mathcal{G}, \mathcal{A})$ be a role-separated graph-extended KB in which $\mathcal{T}$ is expressed in $\mathcal{SHOQ}$ and $\mathcal{G}$ is acyclic. Our algorithm first preprocesses $\mathcal{T}$ into a set of rules $\Xi_{\mathcal{T}}(\mathcal{T})$ and

an ABox $\Xi_{\mathcal{A}}(\mathcal{T})$. This step can be seen as an application of the structural transformation (Plaisted & Greenbaum 1986) adapted to DLs, where complex concepts are replaced with fresh atomic ones, followed by the translation of certain concepts into first-order logic. Due to lack of space, we leave the technical details to (Motik, Shearer, & Horrocks 2008, Section 4.1). In the rest of this paper, we assume that, for each named individual $a \in N_I$, the set of atomic concepts N_C contains a distinct *nominal guard concept* O_a. These concepts are used internally by our algorithm and are not allowed to occur in any input knowledge bases. The preprocessing produces HT-rules, which have the following form.

Definition 10 (HT-Rule)**.** *An* HT-rule *has the form* (2) *where R_i, S_i, and T_i are atomic roles, A_i and B_i are atomic concepts, O_{a_i} are nominal guard concepts, C_i and D_i are either atomic but not nominal guard concepts or they are of the form $\geq n\, R.A$ or $\geq n\, R.\neg A$ for A an atomic but not a nominal guard concept, and each y_i and y_{a_i} in the consequent occurs in an atom in the antecedent.*

$$\begin{array}{l} \bigwedge A_i(x) \wedge \bigwedge R_i(x,y_i) \wedge \bigwedge B_i(y_i) \wedge \bigwedge O_{a_i}(y_{a_i}) \rightarrow \\ \bigvee C_i(x) \vee \bigvee D_i(y_i) \vee \bigvee S_i(x,y_i) \vee \bigvee T_i(x,y_{a_i}) \vee \\ \bigvee x \approx y_{a_i} \vee \bigvee y_i \approx y_{a_j} \vee \bigvee y_i \approx y_j \end{array} \tag{2}$$

The atoms of the form $x \approx y_{a_i}$ and $y_i \approx y_{a_j}$ stem from nominals; for example, $C \sqsubseteq \{a\}$ is translated into a rule $C(x) \wedge O_a(y_a) \rightarrow x \approx y_a$ and an assertion $O_a(a)$; such a translation ensures that the rules do not contain individuals. The atoms $y_i \approx y_j$ stem from the translation of number restrictions; for example, $\top \sqsubseteq \leq 1\, R.\top$ is translated into $R(x,y_1) \wedge R(x,y_2) \rightarrow y_1 \approx y_2$. In the rest of this paper, we use $\exists R.C$ as an abbreviation for $\geq 1\, R.C$.

Our algorithm takes a set of rules $\mathcal{R}$, a GBox $\mathcal{G}$, and an ABox $\mathcal{A}$, and it decides satisfiability of $(\mathcal{R}, \mathcal{G}, \mathcal{A})$. Definition 11 specifies the conditions on $\mathcal{R}$ and $\mathcal{G}$ that ensure termination of the algorithm. It is straightforward to see that the set of rules $\mathcal{R} = \Xi_{\mathcal{T}}(\mathcal{T}) \cup \mathcal{P}$ is admissible.

Definition 11 (Admissibility)**.** *A set of rules $\mathcal{R}$ and a GBox $\mathcal{G}$ are* admissible *if $\mathcal{G}$ is acyclic, the set of atomic roles N_R can be split into disjoint subsets of* tree roles N_{R_t} *and* graph roles N_{R_g}*, and $\mathcal{R}$ can be split into disjoint subsets $\mathcal{R}_t$ and $\mathcal{R}_g$ of* tree *and* graph *rules such that (*i*) each $r \in \mathcal{R}_t$ is an HT-rule in which all roles are tree roles, (*ii*) each $r \in \mathcal{R}_g$ is connected and all roles in it are graph roles, and (*iii*) all roles in each graph in $\mathcal{G}$ are graph roles.*

We next describe the main aspects of our algorithm by means of an example. Let $\mathcal{K}_1 = (\mathcal{T}_1, \mathcal{P}_1, \mathcal{G}_1, \mathcal{A}_1)$ be the graph-extended KB where $\mathcal{T}_1 = \{C \sqsubseteq \exists R.A,\ B \sqsubseteq \{b\}\}$, $\mathcal{P}_1 = \emptyset$, $\mathcal{A}_1 = \{C(a)\}$, and $\mathcal{G}_1$ contains the following description graph G_1:

$$G_1: \quad \begin{array}{ccccc} V_1 = \{1,2,3\} & & M_1 = \{A\} & & \\ \lambda_1\langle 1\rangle = \{A\} & & \lambda_1\langle 2\rangle = \{B\} & & \lambda_1\langle 3\rangle = \{C\} \\ 1 \xrightarrow{S} 2 & & 2 \xrightarrow{T} 3 & & 1 \xrightarrow{U} 3 \end{array}$$

Preprocessing produces the ABox $\Xi_{\mathcal{A}}(\mathcal{T}_1) = \{O_b(b)\}$ and the following set of rules $\Xi_{\mathcal{T}}(\mathcal{T}_1)$:

$$C(x) \rightarrow (\exists R.A)(x) \tag{3}$$

$$B(x) \wedge O_b(y_b) \rightarrow x \approx y_b \tag{4}$$

Let $\mathcal{R}_1 = \Xi_{\mathcal{T}}(\mathcal{T}_1)$ and $\mathcal{A}_1^1 = \Xi_{\mathcal{A}}(\mathcal{T}_1) \cup \mathcal{A}_1$. It is easy to see that $\mathcal{R}_1$ and $\mathcal{G}_1$ are admissible: R is the only tree role; S, T, and U are the graph roles; and the rules in $\mathcal{R}_1$ are HT-rules. Note that a and b are named individuals.

The hypertableau algorithm consists of derivation rules shown in Table 2. By successively applying these rules to $\mathcal{R}_1, \mathcal{G}_1$, and $\mathcal{A}_1^1$, the algorithm tries to construct an abstraction of a model of $(\mathcal{R}_1, \mathcal{G}_1, \mathcal{A}_1^1)$.

The *Hyp*-rule tries to match all atoms from the body or a rule to assertions in an ABox; if this is successful, an assertion from the rule's head is then derived nondeterministically. Thus, given $C(a)$ and (3), the *Hyp*-rule derives

$$\mathcal{A}_1^2 = \mathcal{A}_1^1 \cup \{\exists R.A(a)\}. \tag{5}$$

To satisfy the assertion $\exists R.A(a)$, the $\exists$-rule introduces a fresh individual s; since R is a tree role, s is called a *tree successor* of a. To keep track of the successor relation, our algorithm represents individuals as finite strings; thus, s is represented as $a.\tau_1$ where τ_1 is a *tree symbol*. Thus, the application of the $\exists$-rule derives

$$\mathcal{A}_1^3 = \mathcal{A}_1^2 \cup \{R(a, a.\tau_1),\ A(a.\tau_1)\}. \tag{6}$$

A is a main concept of G_1, so the assertion $A(a.\tau_1)$ must occur in an instance of G_1 at vertex 1. Thus, the $G_\exists$-rule derives the ABox $\mathcal{A}_1^4$.

$$\mathcal{A}_1^4 = \mathcal{A}_1^3 \cup \{G_1(a.\tau_1, a.\tau_1.\gamma_1, a.\tau_1.\gamma_2)\} \tag{7}$$

Here, $a.\tau_1.\gamma_1$ and $a.\tau_1.\gamma_2$ are fresh *graph successors* of $a.\tau_1$ where γ_1 and γ_2 are *graph symbols*. The G_L-rule then connects all the vertices in the instance of G_1. For brevity, we do not show all the derived assertions; however, note that they include $B(a.\tau_1.\gamma_1)$ and $C(a.\tau_1.\gamma_2)$. Thus, the same inferences can be repeated: the *Hyp*-rule derives $\exists R.A(a.\tau_1.\gamma_2)$, the $\exists$-rule derives $R(a.\tau_1.\gamma_2, a.\tau_1.\gamma_2.\tau_1)$ and $A(a.\tau_1.\gamma_2.\tau_1)$, the $G_\exists$-rule derives the graph assertion $G_1(a.\tau_1.\gamma_2.\tau_1, a.\tau_1.\gamma_2.\tau_1.\gamma_1, a.\tau_1.\gamma_2.\tau_1.\gamma_2)$, and the G_L-rule connects the vertices. Let $\mathcal{A}_1^5$ be the resulting ABox.

Clearly, unrestricted application of the $\exists$- and $G_\exists$-rule would result in a nonterminating algorithm. To ensure termination, our algorithm applies *blocking* in the same say as the standard tableau algorithms. Roughly speaking, $a.\tau_1$ and $a.\tau_1.\gamma_2.\tau_1$ occur in $\mathcal{A}_1^5$ in the same concepts, so the former individual blocks the latter—that is, the $\exists$- and $G_\exists$-rule are not applied to (the successors of) the blocked individual.

A tree individual (e.g., $a.\tau_1$) and all of its graph successors (e.g., $a.\tau_1.\gamma_1$ and $a.\tau_1.\gamma_2$) are said to form a *cluster*; furthermore, all named individuals (e.g., a and b) and all of their graph successors (e.g., $a.\gamma_1$, $a.\gamma_2$, $b.\gamma_1$, and $b.\gamma_2$) form a single cluster as well. The ABox $\mathcal{A}_1^5$ can thus be seen as consisting of tree fragments with embedded clusters, where each graph assertion contains individuals from the same cluster. This property, formalized in Lemma 1, is a direct consequence of the separation of roles between the TBox and the GBox, and it holds the key to proving termination. Intuitively, the size of the tree part of each ABox is bounded due to blocking, and the size of each cluster is bounded due to acyclicity of the GBox; since the total number of individuals is bounded, the number applications of each derivation rule is bounded as well.

Table 2: 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 $\sigma(x)$ is not indirectly blocked for each $x \in N_V$,
2.2 $\sigma(U_i) \in \mathcal{A}$ for each $1 \leq i \leq m$, and
2.3 $\sigma(V_j) \notin \mathcal{A}$ for each $1 \leq j \leq n$,
then $\mathcal{A}_1 = \mathcal{A} \cup \{\bot\}$ if $n = 0$; and
$\mathcal{A}_j := \mathcal{A} \cup \{\sigma(V_j)\}$ for $1 \leq j \leq n$ otherwise.

$\approx$-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; and
$\mathcal{A}_1 := \mathsf{merge}_\mathcal{A}(t \rightarrow s)$ otherwise.

$G_\bot$-rule

If 1. $\{G(s_1, \ldots, s_\ell), G(t_1, \ldots, t_\ell)\} \subseteq \mathcal{A}$, and
2. $s_i = t_j$ for some $i \neq j$
then $\mathcal{A}_1 := \mathcal{A} \cup \{\bot\}$.

$G_\lhd$-rule

If 1. $G_1 \lhd G_2 \in \mathcal{G}_S$,
2. $G_2(s_1, \ldots, s_{\ell_2}) \in \mathcal{A}$, and
3. $G_1(s_1, \ldots, s_{\ell_1}) \notin \mathcal{A}$
then $\mathcal{A}_1 := \mathcal{A} \cup \{G_1(s_1, \ldots, s_{\ell_1})\}$.

$G_\leftrightarrow$-rule

If 1. $G_1[u_1, \ldots, u_n] \leftrightarrow G_2[w_1, \ldots w_n] \in \mathcal{G}_A$,
2. $\{G_1(s_1, \ldots, s_{\ell_1}), G_2(t_1, \ldots, t_{\ell_2})\} \subseteq \mathcal{A}$,
3. $s_{u_i} = t_{w_i}$ for some $1 \leq i \leq n$, and
4. $\{s_{u_j} \approx t_{w_j} \mid 1 \leq j \leq n\} \not\subseteq \mathcal{A}$
then $\mathcal{A}_1 := \mathcal{A} \cup \{s_{u_j} \approx t_{w_j} \mid 1 \leq j \leq n\}$.

$\geq$-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
$\{R(s, u_i), C(u_i) \mid 1 \leq i \leq n\} \cup \{u_i \not\approx u_j \mid 1 \leq i < j \leq n\} \subseteq \mathcal{A}$,
then $\mathcal{A}_1 := \mathcal{A} \cup \{R(s, t_i),\ C(t_i) \mid 1 \leq i \leq n\} \cup \{t_i \not\approx t_j \mid 1 \leq i < j \leq n\}$
where $t_1, \ldots, t_n$ are fresh pairwise distinct tree successors of s.

$\bot$-rule

If $s \not\approx s \in \mathcal{A}$ or $\{A(s), \neg A(s)\} \subseteq \mathcal{A}$
then $\mathcal{A}_1 := \mathcal{A} \cup \{\bot\}$.

$G_\approx$-rule

If 1. $\{G(s_1, \ldots, s_\ell), G(t_1, \ldots, t_\ell)\} \subseteq \mathcal{A}$, and
2. $s_i = t_i$ for some $1 \leq i \leq \ell$
3. $\{s_j \approx t_j \mid 1 \leq j \leq \ell\} \not\subseteq \mathcal{A}$
then $\mathcal{A}_1 := \mathcal{A} \cup \{s_j \approx t_j \mid 1 \leq j \leq \ell\}$.

$G_\exists$-rule

If 1. $A(s) \in \mathcal{A}$ such that $A \in M$ for some $G = (V, E, \lambda, M) \in \mathcal{G}_G$,
2. s is not blocked in $\mathcal{A}$, and
3. for each $v_i \in V_A$, no individuals $u_1, \ldots, u_\ell$ exists such that
$G(u_1, \ldots, u_\ell) \in \mathcal{A}$ and $u_{v_i} = s$
then given V_A of the form $\{v_1, \ldots, v_n\}$, for each $1 \leq i \leq n$ derive
$\mathcal{A}_i := \mathcal{A} \cup \{G(t_1, \ldots, t_\ell)\}$ where $t_{v_i} = s$ and all other
t_k are fresh graph individuals from the same cluster as s.

G_L-rule

If 1. $G(s_1, \ldots, s_\ell) \in \mathcal{A}$ with $G = (V, E, \lambda, M)$, and
2. $\{A(s_i) \mid A \in \lambda\langle i \rangle\} \cup \{R(s_i, s_j) \mid R \in \lambda\langle i, j \rangle\} \not\subseteq \mathcal{A}$
then $\mathcal{A}_1 := \mathcal{A} \cup \{A(s_i) \mid A \in \lambda\langle i \rangle\} \cup \{R(s_i, s_j) \mid R \in \lambda\langle i, j \rangle\}$.

Note: $\ell_{(i)}$ is the arity of $G_{(i)}$, $\mathcal{A}$ is a generalized ABox, and $N_\mathcal{A}$ is the set of individuals occurring in $\mathcal{A}$.

Nominals, however, introduce a slight complication. Consider again the ABox $\mathcal{A}_1^5$: from $B(a.\tau_1.\gamma_1)$, $O_b(b)$, and (4), the *Hyp*-rule derives $a.\tau_1.\gamma_1 \approx b$. The $\approx$-rule then *prunes* $a.\tau_1.\gamma_1$ (i.e., it removes all graph and tree successors of $a.\tau_1.\gamma_1$) and replaces it with b; pruning is necessary to avoid the so-called "yo-yo" problem (Baader & Sattler 2001). The resulting ABox thus contains the graph assertion $G_1(a.\tau_1, b, a.\tau_1.\gamma_2)$, in which b is not from the same cluster as $a.\tau_1$ and $a.\tau_1.\gamma_2$. This is remedied through *graph cleanup*: the mentioned assertion is replaced with $G_1(b.\gamma_1, b, b.\gamma_2)$, where $b.\gamma_1$ and $b.\gamma_2$ are fresh individuals from the cluster of b. The next time a graph assertion of the form $G_1(s, b, t)$ is derived, the key property allows us to reuse the individuals $b.\gamma_1$ and $b.\gamma_2$ for s and t in the cleanup. This allows us to establish a bound on the number of individuals introduced by the cleanup and prove termination.

Definition 12 (Hypertableau Algorithm)**.**

Generalized Individuals. *Let* T *and* Γ *be countably infinite sets of* tree *and* graph symbols*, respectively, such that* T, Γ, *and* N_I *are all mutually disjoint. A* generalized individual *is a finite string of symbols* $b.\alpha_1. \ldots .\alpha_n$ *such that* $n \geq 0$, $b \in N_I$, $\alpha_i \in \mathrm{T} \cup \Gamma$ *for* $1 \leq i \leq n$, *and* $\alpha_i \in \Gamma$ *implies* $\alpha_{i+1} \notin \Gamma$. *If* $n \geq 1$ *and* $\alpha_n \in \mathrm{T}$ *(resp.* $\alpha_n \in \Gamma$*), the individual is called a* tree *(resp.* graph*)* individual.

Successors and Predecessors. *A tree or graph 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.*

Cluster. *Individuals* s *and* t are from the same cluster *if (*i*) each individual in* $\{s, t\}$ *is either named or a graph successor of a named individual, or (*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. *In the rest of this paper, we allow ABoxes to contain generalized individuals and the assertion* $\bot$ *that is false in all interpretations, and we take* $a \approx b$ *(*$a \not\approx b$*) to also stand for* $b \approx a$ *(*$b \not\approx a$*).*

Input ABox. *An ABox that contains only named individuals is called an* input ABox.

Single Anywhere Blocking. *A concept is* blocking-relevant *if it is of the form* A, $\geq n\, R.A$, *or* $\geq n\, R.\neg A$, *for* A *an atomic concept and* R *an atomic role. The* label *of an individual* s *in an ABox* $\mathcal{A}$ *is defined as follows:*

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

We assume that we are given some (arbitrary) strict order $<$ *on the generalized individuals such that* $t < s$ *whenever* t *is an ancestor of* s.[2] *By induction on* $<$*, each individual* s *in* $\mathcal{A}$ *is assigned a status as follows: (*i*) a tree individual* s *is* directly blocked by *a tree individual* t *if* t *is not blocked,* $t < s$*, and* $\mathcal{L}_\mathcal{A}(s) = \mathcal{L}_\mathcal{A}(t)$*; (*ii*)* s *is* indirectly blocked *if it*

[2] In practice, this can be the order of individual introduction.

*has a predecessor that is blocked; and (*iii*) s is* blocked *if it is either directly or indirectly blocked.*

Pruning. *The result of* pruning *an individual s in an ABox $\mathcal{A}$ is the ABox obtained from $\mathcal{A}$ by removing all assertions that contain a descendant of s.*

Graph Cleanup. *Let $\mathcal{A}$ be an ABox containing an assertion $G(u_1, \ldots, u_\ell)$ where some u_i and u_j are not from the same cluster, u_i is of the form s or $s.\gamma_m$ for $\gamma_m \in \Gamma$ and s a tree or named individual, u_j is of the form t or $t.\gamma_n$ for $\gamma_n \in \Gamma$ and t a tree or named individual, and s is a named individual or an ancestor of t. A* cleanup *of u_j is obtained from $\mathcal{A}$ by pruning u_j and then replacing it everywhere in $\mathcal{A}$ with the individual t defined as follows:*

- *if $\mathcal{A}$ contains $G(v_1, \ldots, v_\ell)$ such that $u_i = v_i$ and v_j is from the same cluster as u_i, then $t = v_j$;*
- *otherwise, t is a fresh graph successor of s.*

A graph cleanup *of $\mathcal{A}$ is obtained from $\mathcal{A}$ by repeatedly applying cleanup to individuals in $\mathcal{A}$ as long as possible.*

Merging. *The ABox* $\mathsf{merge}_{\mathcal{A}}(s \to t)$ *is obtained from $\mathcal{A}$ by pruning s, replacing s with t in all assertions, and then applying a graph cleanup.*

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

Derivation Rules. *Table 2 specifies* derivation rules *that, given a clash-free ABox $\mathcal{A}$, a set of rules $\mathcal{R}$, and a GBox $\mathcal{G}$, derive the ABoxes $\langle \mathcal{A}_1, \ldots, \mathcal{A}_n \rangle$. In the Hyp-rule, σ maps N_V to the individuals in $\mathcal{A}$, and $\sigma(U)$ is obtained from U by replacing each variable x with $\sigma(x)$.*

Rule Precedence. *The $G_\exists$-rule is applicable to an ABox only if the $\bot$-, $\approx$-, $G_\bot$-, $G_\approx$-, $G_\lhd$-, and G_L-rule are not applicable to the ABox.*

Derivation. *A* derivation *for a set of admissible rules $\mathcal{R}$, a GBox $\mathcal{G}$, and an input ABox $\mathcal{A}$ is a pair (T, ρ) where T is a finitely branching tree and ρ 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 $\mathcal{R}$, $\mathcal{G}$, and $\rho(t)$, then t has children $t_1, \ldots, t_n$ such that the ABoxes $\langle \rho(t_1), \ldots, \rho(t_n) \rangle$ are exactly the results of applying one (arbitrarily chosen, but respecting the rule precedence) applicable derivation rule to $\mathcal{R}$, $\mathcal{G}$, and $\rho(t)$. The derivation is* successful *if T contains a leaf node labeled with a clash-free ABox.*

To show soundness, completeness, and termination of the hypertableau algorithm, we first prove the following lemma, which shows that all ABoxes labeling a node in a derivation are of a particular shape.

Lemma 1. *Each ABox $\mathcal{A}'$ labeling a node in a derivation for an admissible set of rules $\mathcal{R}$, GBox $\mathcal{G}$, and an input ABox $\mathcal{A}$ satisfies the following properties, for a and b named individuals, u a generalized individual, $\gamma_i, \gamma_j \in \Gamma$, and $\tau_i, \tau_j \in \mathrm{T}$.*

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 $R(u,a)$.*
2. *Each $s \approx t \in \mathcal{A}'$ is of the form $u \approx u$, $a \approx u$, $a.\gamma_i \approx b.\gamma_j$, $u.\tau_i \approx u.\tau_j$, $u \approx u.\gamma_i$, or $u.\gamma_i \approx u.\gamma_j$.*
3. *In each $G(s_1, \ldots, s_\ell) \in \mathcal{A}'$ and each $U(s_1, s_2) \in \mathcal{A}'$ with U a graph role, all individuals s_i are from the same cluster; in the latter case, s_1 and s_2 occur in some graph assertion in $\mathcal{A}'$.*
4. *In each $O_a(s) \in \mathcal{A}'$ for O_a a nominal guard concept, the individual s is named.*
5. *For each tree individual t_n occurring in $\mathcal{A}'$, we have $\{R_0(s_0, t_0), \ldots, R_n(s_n, t_n)\} \subseteq \mathcal{A}'$ such that (*i*) s_0 is a named individual, (*ii*) each t_i is a tree successor of s_i, (*iii*) for each $1 \leq i \leq n$, the individual s_i is from the same cluster as t_{i-1}, and (*iv*) R_i is a tree role.*

Proof. The proof is by induction on rule applications. The induction base is trivial. Assume that the claim holds for an ABox and consider the inferences deriving some $\mathcal{A}'$.

($\bot$-rule) The ABox $\mathcal{A}'$ trivially satisfies Conditions 1–5.

($G_\bot$-,$G_\lhd$-, $G_\leftrightarrow$-, $G_\approx$-, and G_L-rule) These rules are always applied to individuals in the same cluster, so $\mathcal{A}'$ satisfies Conditions 1–5.

($G_\exists$-rule) All t_i are from the same cluster as s, so $\mathcal{A}'$ satisfies Conditions 1–5.

($\geq$-rule) The t_i are tree successors of s and C is not a nominal guard concept, so $\mathcal{A}'$ satisfies Conditions 1–5.

(*Hyp*-rule) Consider an application of the *Hyp*-rule to a rule $r \in \mathcal{R}$. No rule contains nominal guard concepts in the consequent, so $\mathcal{A}'$ satisfies Condition 4. If r is a graph rule, it is connected, so all variables in r are matched to individuals in the same cluster and $\mathcal{A}'$ satisfies Conditions 1–5.

If $s \approx t$ is derived by instantiating $x \approx y_a$ or $y_i \approx y_a$ in a tree rule r, the antecedent of r contains $O_a(y_a)$. This atom is matched to an assertion $O_a(t)$ in which, by Condition 4, t is named. Hence, $s \approx t$ satisfies Condition 2.

If $s \approx t$ is derived by instantiating $y_i \approx y_j$ in a tree rule r, the antecedent of r contains atoms $R(x, y_i)$ and $S(x, y_j)$ that are matched to assertions $R(u, s)$ and $S(u, t)$ satisfying Condition 1. Clearly, $s \approx t$ then satisfies Condition 2.

If $R(s,t)$ is derived by instantiating $R(x, y_i)$ in a tree rule r, the antecedent of r contains an atom $S(x, y_i)$ that is matched to assertion $S(s,t)$ satisfying Condition 1. Clearly, $R(s,t)$ satisfies Condition 1 as well.

If $R(s,t)$ is derived by instantiating $R(x, y_{a_i})$ in a tree rule r, the antecedent of r contains an atom $O_{a_i}(y_{a_i})$ that is matched to an assertion $O_{a_i}(t)$ in which, by Condition 4, t is named. Hence, $R(s,t)$ satisfies Condition 1.

($\approx$-rule) Consider the types of equalities to which the rule can be applied. For $a.\gamma_i \approx b.\gamma_j$, $u \approx u.\gamma_i$, or $u.\gamma_i \approx u.\gamma_j$, the rule simply replaces an individual with another individual from the same cluster. For $u \approx u.\gamma_i$, the rule replaces $u.\gamma_i$ with u. For $u.\tau_i \approx u.\tau_j$, the rule prunes one individual, thus removing all individuals from its cluster, and then merges the pruned individual into the other individual. Clearly, $\mathcal{A}'$ satisfies Conditions 1–5.

If the $\approx$-rule is applied to $a \approx u$, then u is pruned and merged into a. Replacing u with a in some $R(b,u)$ or $b \approx u$ produces $R(b,a)$ or $b \approx a$, respectively, which satisfy Conditions 1 or 2. Replacing u with a in $G(s_1, \ldots, s_n)$ where $s_i = u$ produces at first an assertion that does not satisfy Condition 3; however, the graph cleanup then replaces each s_j with a graph individual from the same cluster as a. For $U(s_1, s_2)$ with U a graph role, s_1 and s_2 occur in a graph assertion, so graph cleanup is applied to s_1 and/or s_2. □

Theorem 1 summarizes the properties of our algorithm.

Theorem 1. *For an admissible set of rules $\mathcal{R}$ and GBox $\mathcal{G}$, and an input ABox $\mathcal{A}$,*

1. *if $(\mathcal{R}, \mathcal{G}, \mathcal{A})$ is satisfiable, then each derivation for $\mathcal{R}$, $\mathcal{G}$, and $\mathcal{A}$ is successful,*
2. *if a successful derivation for $\mathcal{R}$, $\mathcal{G}$, and $\mathcal{A}$ exists, then $(\mathcal{R}, \mathcal{G}, \mathcal{A})$ is satisfiable, and*
3. *each derivation for $\mathcal{R}$, $\mathcal{G}$, and $\mathcal{A}$ is finite.*

Proof of Claim 1. The claim follows from the following property: if $(\mathcal{R}, \mathcal{G}, \mathcal{A})$ is satisfiable and $\langle \mathcal{A}_1, \dots, \mathcal{A}_n \rangle$ are the result of applying a derivation rule to $\mathcal{R}$, $\mathcal{G}$, and $\mathcal{A}$, then $(\mathcal{R}, \mathcal{G}, \mathcal{A}_i)$ is satisfiable for some $1 \leq i \leq n$. The proof is straightforward for all but the $\approx$-rule, in which the graph cleanup step is nonstandard. Let I be a model of $(\mathcal{R}, \mathcal{G}, \mathcal{A})$ and consider an application of the $\approx$-rule to $s \approx t$, producing an ABox $\mathcal{A}_1$. Let $\mathcal{A}'$ be the ABox obtained from $\mathcal{A}$ by pruning s and then replacing it with t. Since $I \models s \approx t$, we have $s^I = t^I$, so clearly $I \models \mathcal{A}'$. The ABox $\mathcal{A}_1$ is obtained from $\mathcal{A}'$ by graph cleanup, which can additionally replace some individuals u_i with v_i. If v_i is fresh, we can extend I to obtain a model of $\mathcal{A}_1$; otherwise, v_i occurs in $\mathcal{A}'$ in a graph assertion for the same graph so, by the key property from Definition 6, $u_j^I = v_j^I$ for each j. Clearly, $(\mathcal{R}, \mathcal{G}, \mathcal{A}_1)$ is satisfiable. □

Proof of Claim 2. For $\mathcal{A}''$ a clash-free ABox labeling a leaf of a derivation for $\mathcal{R}$, $\mathcal{G}$, and $\mathcal{A}$, let $\mathcal{A}'$ be obtained from $\mathcal{A}''$ by removing (*i*) all assertions that contain an indirectly blocked individual and (*ii*) all assertions that contain a directly blocked individual and a graph role or a description graph. The ABox $\mathcal{A}'$ satisfies Lemma 1.

Let Λ be the set of all individuals in $\mathcal{A}'$. We define the function $[s]$ on each $s \in \Lambda$ as follows: if s is blocked in $\mathcal{A}'$ by s', then $[s] = s'$; otherwise, $[s] = s$. Furthermore, for each tree individual $s \in \Lambda$ that is blocked by a tree individual s' and for $u_1, \dots, u_k$ all graph individuals in $\mathcal{A}'$ from the same cluster as s', we introduce fresh graph individuals $v_1, \dots, v_k$ and define $[\cdot]$ on them as $[v_i] = u_i$. Let Υ be the set of all individuals introduced in this way.

We now define an interpretation I as follows, for each atomic concept A, tree role R, graph role U, and graph G:

$$
\begin{aligned}
\triangle^I &= \Lambda \cup \Upsilon \\
s^I &= s \text{ for each } s \in \triangle^I \\
A^I &= \{s \mid \text{for each } s \in \triangle^I \text{ such that } A([s]) \in \mathcal{A}'\} \\
R^I &= \{\langle s, t \rangle \mid \text{for all } s, t \in \triangle^I \text{ such that } R([s], t) \in \mathcal{A}'\} \\
U^I &= \{\langle s, t \rangle \mid \text{for all } s, t \in \triangle^I \text{ such that } U([s], [t]) \in \mathcal{A}'\} \\
G^I &= \{\langle s_1, \dots, s_\ell \rangle \mid \text{for all } s_1, \dots, s_\ell \in \triangle^I \text{ such that} \\
&\qquad\qquad G([s_1], \dots, [s_\ell]) \in \mathcal{A}'\}
\end{aligned}
$$

We now show that $I \models (\mathcal{R}, \mathcal{G}, \mathcal{A}')$. For each $s \approx t \in \mathcal{A}'$, since the $\approx$-rule is not applicable to $\mathcal{A}'$, we have $s = t$, so $I \models s \approx t$. For each $s \not\approx t \in \mathcal{A}'$, since the $\bot$-rule is not applicable to $\mathcal{A}'$, we have $s \neq t$, so $I \models s \not\approx t$.

Consider each $\geq n\, R.C(s) \in \mathcal{A}'$. By the definition of blocking, we have $\geq n\, R.C([s]) \in \mathcal{A}'$; since the $\geq$-rule is not applicable to $\mathcal{A}'$, individuals $u_1, \dots, u_n$ exists that satisfy the precondition of the rule; but then, by the definition of I, we have $\langle s, u_i \rangle \in R^I$ for each $1 \leq i \leq n$ and $u_i^I \neq u_j^I$ for each $1 \leq i < j \leq n$, so $I \models \geq n\, R.C(s)$.

Consider a tree rule r of the form (2) and σ a mapping of variables to $\triangle^I$ such that $I \models \sigma(B_i)$ for each body atom B_i of r. Let σ' be a mapping defined as $\sigma'(x) = [\sigma(x)]$, $\sigma'(y_i) = \sigma(y_i)$, and $\sigma'(y_{a_i}) = \sigma(y_{a_i})$. By the definition of I and the structure of r, then $\sigma'(B_i) \in \mathcal{A}'$. Since the *Hyp*-rule is not applicable to r, $\mathcal{A}'$, and σ', then $\sigma'(H_j) \in \mathcal{A}'$ for some head atom H_j of r. But then, by the definition of I and the structure of r, we have $I \models \sigma(H_j)$.

Consider a graph rule r of the form (1) and σ a mapping of variables to $\triangle^I$ such that $I \models \sigma(B_i)$ for each $1 \leq i \leq n$. Let σ' be a mapping defined as $\sigma'(z) = [\sigma(z)]$ for each variable z occurring in r. By the definition of I, then $\sigma'(B_i) \in \mathcal{A}'$. Since the *Hyp*-rule is not applicable to r, $\mathcal{A}'$, and σ', then $\sigma'(H_j) \in \mathcal{A}'$ for some $1 \leq j \leq m$. But then, by the definition of I, we have $I \models \sigma(H_j)$. The proof that I satisfies conditions of Definition 6 is completely analogous and we omit it for the sake of brevity.

If $\alpha \in \mathcal{A}$ but $\alpha \notin \mathcal{A}'$, then some named individuals in α have been merged into other named individuals, producing an assertion $\alpha' \in \mathcal{A}'$. Clearly, I can be extended to a model of $(\mathcal{R}, \mathcal{G}, \mathcal{A})$ by interpreting the merged individuals. □

Proof of Claim 3. Let (T, ρ) be a derivation for $\mathcal{R}$, $\mathcal{G}$, and $\mathcal{A}$. We show that, in the course of the derivation, (1) each derivation rule can be applied to a set of assertions only once; (2) the number of tree ancestors of each tree individual is bounded; (3) the $G_\exists$-rule can be applied for the same graph G to (different) assertions containing the same individual s at most twice; (4) the number of graph individuals introduced in each cluster is bounded; and (5) the number of graph individuals introduced by graph cleanup is bounded. Together, all these properties imply that (6) the number of individuals introduced in the course of a derivation is bounded. By (6), the number of rule applications is bounded as well, which implies our claim.

(1) This claim holds in exactly the same way as in the case of standard (hyper)tableau algorithms: if, for some derivation node $t \in T$, a derivation rule is applied to a subset of the assertions of $\rho(t)$, then the assertions are added to $\rho(t)$ that, for each descendant node t' of t, prevent the reapplication of the same derivation rule to the same assertions in $\rho(t')$.

(2) Let c be the number of atomic concepts occurring in $\mathcal{R}$, $\mathcal{G}$, and $\mathcal{A}$. By Condition 5 of Lemma 1, the ancestors of each tree individual are present in $\rho(t)$ for some $t \in T$. Thus, if a tree individual t has more than $\wp = 2^c$ tree ancestors, two ancestors with the same individual label must exist, so t is necessarily blocked in $\rho(t)$.

(3) If the $G_\exists$-rule is applied for the same G to two assertions containing s, two assertions of the form $G(\dots, s, \dots)$ are introduced in which s occurs at positions i and j, and $i \neq j$. But then, due to rule precedence, the $G_\bot$-rule derives $\bot$ before the $G_\exists$-rule is applied to s for the third time.

(4) Let $\prec$ be the order on the graphs in $\mathcal{G}$ that satisfies conditions of Definition 8. Assume that, whenever it introduces an individual t by an application of the rule to an individual s and a graph G, the $G_\exists$-rule assigns a finite

string of description graphs $\omega(t)$ to t such that (*i*) $\omega(t) = G$ if s is a tree or named individual, and (*ii*) $\omega(t) = \omega(s).G$ otherwise. By induction on the applications of the $G_\exists$-rule, we show that the following property (‡) holds: for each graph individual t occurring in an ABox $\mathcal{A}'$ labeling a derivation node, $\mathcal{A}'$ contains a graph assertion of the form $G_n(u_1, \ldots, u_{\ell_n})$ with $u_i = t$ for some i; furthermore, for $\omega(t) = G_1. \ldots .G_{n-1}.G_n$, we have $G_1 \prec \ldots \prec G_{n-1}$ and, if $G_{n-1} \not\prec G_n$, then $\mathcal{A}'$ contains a graph assertion $G_n(u'_1, \ldots, u'_{\ell_n})$ such that $u'_j = t$ and $i \neq j$. Property (4) then follows straightforwardly from (3), (‡), and the fact that $\prec$ is acyclic and finite.

Assume that (‡) holds for some ABox $\mathcal{A}'$, the $G_\exists$-rule is applied to an assertion $A(s) \in \mathcal{A}'$ and a description graph $G' = (V', E', \lambda', M')$ with $A \in M'$, and a fresh graph individual t is introduced. If s is a tree or named individual, (‡) holds trivially, so assume that s is a graph individual such that $\omega(s) = G_1. \ldots .G_{n-1}.G_n$. By the rule precedence, the $G_\bot$-rule is not applicable to $\mathcal{A}'$, so s does not occur in $\mathcal{A}'$ in two graph assertions involving G_n; thus, $G_{n-1} \prec G_n$. By (‡), a graph assertion $G_n(u_1, \ldots, u_{\ell_n}) \in \mathcal{A}'$ exists such that $u_i = s$ for some $1 \leq i \leq \ell_n$, where $G_n = (V_n, E_n, \lambda_n, M_n)$. If $G_n \prec G'$, then (‡) holds trivially, so assume that $G_n \not\preceq G'$. The G_L-rule is not applicable to $\mathcal{A}'$, so $\mathcal{A}'$ contains the layout of G_n for the vertices $u_1, \ldots, u_{\ell_n}$. Since the $G_\exists$-rule is applicable to $\mathcal{A}'$, we have $\bot \notin \mathcal{A}'$. Since the $\bot$-rule is not applicable to $\mathcal{A}'$, we have $\neg A(s) \notin \mathcal{A}'$, which implies $\neg A \notin \lambda_n\langle i\rangle$. If $G' \not\trianglelefteq G_n$, then the second condition of Definition 8 requires $\neg A \in \lambda_n\langle i\rangle$, which is a contradiction, so assume that $G' \trianglelefteq G_n$. The first condition of Definition 8 and $\neg A \notin \lambda_n\langle i\rangle$ imply $1 \leq i \leq \ell'$. The $G_\lhd$-rule is not applicable to $\mathcal{A}'$ by the rule precedence, so $G'(u_1, \ldots, u_{\ell'}) \in \mathcal{A}'$. But then, the $G_\exists$-rule introduces another graph assertion $G'(u'_1, \ldots, u'_{\ell'})$ such that $u'_j = s$ and $j \neq i$, so (‡) holds.

(5) From the proof of Lemma 1, we can see that graph cleanup can only be applied to a set of graph assertions $\Theta = \{G_i({u_1}^i, \ldots, {u_\ell}^i)\}$ when some ${u_j}^i$ is replaced with a named individual a. The total number of individuals in each Θ is bounded by (4), so the number of sets Θ different up to the renaming of individuals is bounded as well. By the first case in the definition of graph cleanup, for each different Θ, ${u_j}^i$, and a, fresh graph individuals can be introduced only once. Since the numbers of different Θ, ${u_j}^i$, and a are bounded, the number of graph individuals introduced by the graph cleanup is bounded as well.

(6) Because of (1), the $\geq$-rule can be applied to each individual at most once for each assertion $\geq n\, R.C(s)$. By (1) and (4), the number of successors of s introduced by the $\geq$- and $G_\exists$-rule is bounded. By (2) the number of descendants introduced by these rules is bounded as well. □

Let $\mathcal{K} = (\mathcal{T}, \mathcal{P}, \mathcal{G}, \mathcal{A})$ be a role-separated graph-extended knowledge base in which $\mathcal{T}$ is expressed in $\mathcal{SHOQ}$ and $\mathcal{G}$ is acyclic. Furthermore, let $\Xi_\mathcal{T}(\mathcal{T})$ and $\Xi_\mathcal{A}(\mathcal{T})$ be the set of rules and the ABox obtained by the preprocessing step, $\mathcal{R} = \Xi_\mathcal{T}(\mathcal{T}) \cup \mathcal{P}$, and $\mathcal{A}' = \mathcal{A} \cup \Xi_\mathcal{A}(\mathcal{T})$. Since preprocessing does not affect satisfiability, $\mathcal{K}$ is equisatisfiable with $(\mathcal{R}, \mathcal{G}, \mathcal{A}')$. Furthermore, by inspecting the preprocessing transformation, it is straightforward to see that $\mathcal{R}$ and $\mathcal{G}$ are admissible. This implies the following theorem.

Theorem 2. *Checking satisfiability of a role-separated graph-extended knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{P}, \mathcal{G}, \mathcal{A})$ in which $\mathcal{T}$ is expressed in $\mathcal{SHOQ}$ and $\mathcal{G}$ is acyclic is decidable.*

Rule-Separated Acyclic KBs

The proofs of Claims 2 and 3 of Theorem 1 crucially depend on the fact that each ABox labeling a derivation node satisfies Lemma 1—that is, it is tree-shaped. Role separation is one possible way of achieving this property; however, as we discuss next, it can also be achieved using Definition 9.

Theorem 3. *Checking satisfiability of a rule-separated graph-extended KB $\mathcal{K} = (\mathcal{T}, \mathcal{G}, \mathcal{P}, \mathcal{A})$ in which $\mathcal{T}$ is expressed in $\mathcal{ALCHO}$ and $\mathcal{G}$ is acyclic is decidable.*

Proof. Let $\mathcal{R} = \Xi_\mathcal{T}(\mathcal{T}) \cup \mathcal{P}$, and let $N_{R_{DL}}$ and N_{R_r} be the disjoint subsets of N_R satisfying Definition 9. Since $\mathcal{K}$ is rule-separated, $\mathcal{R}$ can be split into disjoint subsets $\mathcal{R}_{DL}$ and $\mathcal{R}_r$ such that the rules in each of them refer only to roles in $N_{R_{DL}}$ and N_{R_r}, respectively. Furthermore, since $\mathcal{T}$ does not allow for number restrictions, each rule in $\mathcal{R}_{DL}$ is of the form (2) but without atoms of the form $y_i \approx y_j$. Clearly, $\mathcal{K}$ and $(\mathcal{R}, \mathcal{G}, \mathcal{A} \cup \Xi_\mathcal{A}(\mathcal{T}))$ are equisatisfiable.

We now generalize Lemma 1 to the property (♮): each ABox $\mathcal{A}'$ labeling a node in a derivation for $\mathcal{R}$, $\mathcal{G}$, and $\mathcal{A} \cup \Xi_\mathcal{A}(\mathcal{T})$ satisfies the following properties, for $a, b \in N_I$, u a generalized individual, $\gamma_i, \gamma_j \in \Gamma$, and $\tau_i, \tau_j \in \mathrm{T}$.

1. Each $R(s,t) \in \mathcal{A}'$ with R a DL-role is of the form $R(a,b)$, $R(u, u.\tau_i)$, $R(u,a)$, or $R(s_1, s_2)$ for s_1 and s_2 from the same cluster.
2. Each equality $s \approx t \in \mathcal{A}'$ is of the form $u \approx u$, $a \approx u$, $a.\gamma_i \approx b.\gamma_j$, $u \approx u.\gamma_i$, or $u.\gamma_i \approx u.\gamma_j$.
3. In each $G(s_1, \ldots s_\ell) \in \mathcal{A}'$ and $U(s_1, s_2) \in \mathcal{A}'$ with U a rule role, all s_i are from the same cluster; in the latter case, s_1 and s_2 occur in a graph assertion in $\mathcal{A}'$.
4. Conditions 4 and 5 hold as in Lemma 1, with the difference that each R_i in Condition 5 is a DL-role.

The proof of (♮) is analogous to the proof of Lemma 1. One difference is that, since $\mathcal{R}_{DL}$ does not contain atoms of the form $y_i \approx y_j$, the ABox $\mathcal{A}'$ cannot contain atoms of the form $u.\tau_i \approx u.\tau_j$; thus, the "graph part" of $\mathcal{A}'$ cannot interact with the "tree part" of $\mathcal{A}'$ by the $\approx$-rule. Since the rules in $\mathcal{R}_r$ contain only atoms with roles in N_{R_r}, they can be applied only to the individuals in the same cluster; hence, each assertion derived by the *Hyp*-rule satisfies Conditions 2 and 3 of (♮). Unlike in Lemma 1, the rules in $\mathcal{R}_{DL}$ can be applied to both the "tree" and the "graph part" of $\mathcal{A}'$, but they still derive atoms satisfying Condition 1 of (♮).

Since $\mathcal{G}$ is acyclic and (♮) holds, the claims of Theorem 1 hold for $\mathcal{R}$ defined as above in essentially the same way as in Theorem 1; the only difference is in the proof of Claim 2 in the definition of the interpretation of a DL-role R:

$$\begin{aligned} R^I = \{\langle s,t\rangle \mid & \text{ for all } s,t \in \triangle^I \text{ such that } R([s],t) \in \mathcal{A}' \\ & \text{and } s \text{ and } t \text{ are not from the same graph cluster}\} \cup \\ \{\langle s,t\rangle \mid & \text{ for all } s,t \in \triangle^I \text{ such that } R([s],[t]) \in \mathcal{A}' \\ & \text{and } s \text{ and } t \text{ are from the same graph cluster}\} \end{aligned}$$

Satisfiability of $\mathcal{K}$ can thus be decided by applying the hypertableau algorithm to $(\mathcal{R}, \mathcal{G}, \mathcal{A} \cup \Xi_{\mathcal{A}}(\mathcal{T}))$. □

Rule-Separated KBs with an Expressive DL

Propositions 1 and 2 suggest that achieving decidability might be difficult if no weakening of $\mathcal{DL}$ is allowed. If $\mathcal{K}$ is rule-separated, however, the hypertableau algorithm provides a semidecision procedure. The following theorem relies on the standard notion of fair derivations. Intuitively, in a fair derivation, no application of an inference rule can be "postponed" infinitely often. Since derivations can now be infinite, we adjust the notion of a successful derivation.

Definition 13. *A derivation (T, ρ) for $\mathcal{R}$, $\mathcal{G}$, and $\mathcal{A}$ is* unfair *if a branch $t_1, t_2, \ldots$ of T exists such that, for infinitely many nodes $t_{i_1}, t_{i_2}, \ldots$ on that branch, the same derivation rule is applicable to the same assertions in each $\rho(t_{i_j})$.* Fair *is the opposite of unfair.*

A derivation (T, ρ) is successful *if T contains a branch $t_1, t_2, \ldots$ such that each $\rho(t_i)$ is clash-free.*

Theorem 4. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{G}, \mathcal{P}, \mathcal{A})$ be a rule-separated graph-extended KB with $\mathcal{T}$ expressed in $\mathcal{SHOQ}$, and let $\mathcal{R} = \Xi_{\mathcal{T}}(\mathcal{T}) \cup \mathcal{P}$. If $\mathcal{K}$ is satisfiable, then each derivation for $\mathcal{R}$, $\mathcal{G}$, and $\mathcal{A} \cup \Xi_{\mathcal{A}}(\mathcal{T})$ is successful. Conversely, if a fair and successful (but not necessarily finite) derivation for $\mathcal{R}$, $\mathcal{G}$, and $\mathcal{A} \cup \Xi_{\mathcal{A}}(\mathcal{T})$ exists, then $\mathcal{K}$ is satisfiable.*

Proof. The first claim holds in exactly the same way as Claim 1 of Theorem 1. For the second claim, assume that a successful derivation (T, ρ) for $\mathcal{R}$, $\mathcal{G}$, and $\mathcal{A} \cup \Xi_{\mathcal{A}}(\mathcal{T})$ exists. The main difference to the proof of Theorem 3 is that (T, ρ) is not necessarily finite. Let $t_1, t_2, \ldots$ be the branch of T such that each $\rho(t_i)$ is clash-free, and let $\mathcal{A}' = \bigcup_i \bigcap_{j \geq i} \rho(t_j)$. Since (T, ρ) is fair, no derivation rule is applicable to $\mathcal{A}'$. Since $\mathcal{K}$ is rule-separated, each ABox $\rho(t)$ labeling a node $t \in T$ satisfies property (♮) from the proof of Theorem 3; clearly, then $\mathcal{A}'$ satisfies (♮) as well. But then, a model of $\mathcal{R}$, $\mathcal{G}$, and $\mathcal{A} \cup \Xi_{\mathcal{A}}(\mathcal{T})$ can be constructed in exactly the same way as in Theorem 3. □

Let $\mathcal{K}$ be a graph-extended KB as specified in Theorem 4 in which $\mathcal{G}$ is acyclic. By Proposition 1, checking satisfiability of $\mathcal{K}$ is undecidable; however, we believe that the hypertableau algorithm is "likely" to terminate in practice even if $\mathcal{K}$ is satisfiable. The ABoxes generated by the algorithm satisfy property (♮) from the proof of Theorem 3, which enables the usage of blocking; thus, the algorithm can terminate even if $\mathcal{T}$ is cyclic. In contrast, general first-order model-building calculi are unlikely to terminate if $\mathcal{T}$ is cyclic. Nontermination can occur only due to merging of graph individuals from different clusters.

Conclusion and Future Work

We have presented an expressive formalism that extends DLs with description graphs, which allow one to model arbitrarily connected, and not just tree-like structures.

An open problem is to determine the computational complexity of graph-extended knowledge bases. Tableau algorithms generally do not provide worst-case behaviors, so a different approach will be needed. We conjecture that adding one description graph does not increase the complexity of EXPTIME-complete DLs, but adding several description graphs increases the complexity to NEXPTIME. Another open problem is to see whether the restrictions of Theorem 3 can be relaxed. We conjecture that the usage of inverse roles in Proposition 1 is strictly necessary for the undecidability result, and that Theorem 3 can be extended to $\mathcal{SHOQ}$. To confirm or refute these conjectures will be part of our future work.

The main practical challenge is to evaluate the utility of our formalism in applications. To facilitate this we will extend the remodeling algorithm from our previous work to support multiple graphs, extend the Protégé ontology editor[3] to support description graphs, and apply our formalism and tools in practical scenarios.

References

Baader, F., and Sattler, U. 2001. An Overview of Tableau Algorithms for Description Logics. *Studia Logica* 69:5–40.

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

Börger, E.; Grädel, E.; and Gurevich, Y. 1996. *The Classical Decision Problem*. Springer.

Kutz, O.; Horrocks, I.; and Sattler, U. 2006. The Even More Irresistible $\mathcal{SROIQ}$. In *Proc. KR 2006*, 68–78.

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

Motik, B.; Grau, B. C.; and Sattler, U. 2008. Structured Objects in OWL: Representation and Reasoning. In *Proc. WWW 2008*, 555–564.

Motik, B.; Shearer, R.; and Horrocks, I. 2008. Hypertableau Reasoning for Description Logics. Technical report, University of Oxford. See first author's Web page.

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

Rector, A. L.; Nowlan, W. A.; and Glowinski, A. 1993. Goals for concept representation in the galen project. In *Proc. SCAMC '93*, 414–418.

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

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

Vardi, M. Y. 1996. Why Is Modal Logic So Robustly Decidable? In *Proc. DIMACS Workshop*, volume 31 of *DIMACS Series*, 149–184.

[3] `http://protege.stanford.edu/`