Modelling Structured Domains Using Description Graphs and Logic Programming[⋆]

Despoina Magka, Boris Motik, and Ian Horrocks

Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, OX1 3QD, UK

Abstract. Although OWL 2 is widely used to describe complex objects such as chemical molecules, it cannot represent 'structural' features of chemical entities (e.g., having a ring). A combination of rules and *description graphs* (DGs) has been proposed as a possible solution, but it still exhibits several drawbacks. In this paper we present a radically different approach that we call Description Graph Logic Programs. Syntactically, our approach combines DGs, rules, and OWL 2 RL axioms, but its semantics is defined via a translation into logic programs under stable model semantics. The result is an expressive OWL 2 RL-compatible formalism that is well suited for modelling objects with complex structure.

1 Introduction

OWL 2 [7] is commonly used to represent objects with complex structure, such as complex assemblies in engineering applications [8], human anatomy [22], or the structure of chemical molecules [10]. In order to ground our discussion, we next present a concrete application of the latter kind; however, the problems and the solution that we identify apply to numerous similar scenarios.

The European Bioinformatics Institute (EBI) has developed the ChEBI ontology—a public dictionary of *molecular entities* used to ensure interoperability of applications supporting tasks such as drug discovery [17]. In order to automate the classification of molecular entities, ChEBI descriptions have been translated into OWL and then classified using state of the art Semantic Web reasoners. While this has uncovered numerous implicit subsumptions between ChEBI classes, the usefulness of the approach was somewhat limited by a fundamental inability of OWL 2 to precisely represent the structure of complex molecular entities. As we discuss in more detail in Section 3, OWL 2 exhibits a so-called *tree-model* property [23], which prevents one from describing non-tree-like relationships using OWL 2 schema axioms. For example, OWL 2 axioms can state that butane molecules have four carbon atoms, but they cannot state that the four atoms in a cyclobutane molecule are arranged in a ring. Please note that this applies to *schema* descriptions only: the structure of a particular cyclobutane molecule can be represented using class and property assertions, but the general definition of *all* cyclobutane molecules—a problem that terminologies such as ChEBI aim to solve—cannot be

⋆ This work was supported by the EU FP7 project SEALS and the EPSRC projects ConDOR, ExODA, and LogMap.

fully described in OWL 2. As we show in Section 3, an ontology may therefore fail to entail certain desired consequences.

A common solution to this problem is to extend OWL 2 with a rule-based formalism such as SWRL [11]. However, adding rules to even a very small fragment of OWL 2 makes the basic reasoning problems undecidable [14], which hinders practical usability. Decidability can be ensured by applying the rules only to the explicitly named individuals [21], or by restricting the shape of the rules [12]. Such restrictions, however, typically prevent the rules from axiomatising the required structures.

In our previous work, we considered a combination of OWL 2, rules, and *description graphs* (DGs) [20]—a graphical notation for describing non-tree-like structures. We ensured decidability of reasoning via a *property separation* condition and by requiring DGs to be *acyclic*. Intuitively, the latter means that DGs can describe structures of arbitrary shape, but bounded in size, while the former limits the interaction between the OWL and DG parts, thus preventing multiple DG structures from merging into one structure of (potentially) unbounded size. As reported in [10], DGs solved only some of the problems related to the representation of structured objects, and our subsequent discussions with EBI researchers have revealed the following drawbacks.

First, the DG approach [20] does not allow one to define structures based on the *absence* of certain characteristics. For example, an inorganic molecule is commonly described as 'a molecule *not* containing a carbon atom', which can then be used to classify water as an inorganic molecule. Designing an axiomatisation that produces the desired entailment is very cumbersome with the DG approach: apart from stating that 'each water molecule consists of one oxygen and two hydrogen atoms', one must additionally state that 'these three atoms are the only atoms in a water molecule' and that 'neither hydrogen nor oxygen atoms are carbon atoms'. Second, the separation conditions governing the interaction of the OWL 2 and DG components makes the combined language rather difficult to use, as no role can be used in both components. Third, the acyclicity condition from [20] is rather cumbersome: a modeller must add a number of negative class assertions to DGs so as to make any ontology with cyclic implications between DGs unsatisfiable. This solution fails to cleanly separate the semantic consequences of an ontology from the acyclicity check.

In response to this critique, in this paper we present a radically different approach to modelling complex objects via a novel formalism that we call Description Graph Logic Programs (DGLP). At the syntactic level, our approach combines DGs, rules, and OWL 2 RL axioms [19]. In order to overcome the first problem, we give semantics to our formalism via a translation into logic programs (which can be easily done for OWL 2 RL axioms [9]) interpreted under stable model semantics. As we show in Section 4, the resulting formalism can capture conditions based on the absence of information. Moreover, we address the second problem by ensuring decidability without the need for complex property separation conditions.

To address the third problem, in Section 5 we discuss existing syntactic acyclicity conditions, such as weak acyclicity [6] and super-weak acyclicity [16], and argue that they unnecessarily rule out some very simple and intuitively reasonable ontologies. As a remedy, we present a novel *semantic acyclicity* condition. Roughly speaking, a modeller is required to specify a precedence relation describing which DGs are allowed to imply other DGs; a cyclic ontology that is not compatible with this precedence relation entails a special propositional symbol. A cyclic ontology can still entail useful consequences, but termination of reasoning can no longer be guaranteed.

In Section 6 we consider the problem of reasoning with ontologies including only negation-free rules. We show that the standard bottom-up evaluation of logic programs can decide the relevant reasoning problems for semantically acyclic ontologies, and that it can also decide whether an ontology is semantically acyclic. Furthermore, in Section 7 we show that this result can be extended to ontologies with *stratified* negation.

In Section 8 we present the results of a preliminary evaluation of our formalism. We show that molecule descriptions from the ChEBI ontology can be translated into a DGLP ontology that entails the desired subsumption consequences. Furthermore, despite the very preliminary nature of our implementation, we show that reasoning with DGLP ontologies is practically feasible. Thus, in this paper we lay the theoretical foundations of a novel, expressive, and OWL 2 RL-compatible ontology language that is well suited to modelling objects with complex structure.

The proofs of all technical results presented in this paper are given in the appendix.

2 Preliminaries

We assume the reader to be familiar with OWL and description logics. For brevity, we write OWL axioms using the DL notation; please refer to [1] for an overview of the DL syntax and semantics. Let $\Sigma = (\Sigma_C, \Sigma_F, \Sigma_P)$ be a *first-order logic signature*, where Σ_C, Σ_F, and Σ_P are countably infinite sets of constant, function, and predicate symbols, respectively, and where Σ_P contains the 0-ary predicate $\bot$. The arity of a predicate A is given by $ar(A)$. A vector $t_1, \ldots, t_n$ of first-order terms is often abbreviated as $\vec{t}$. An *atom* is a first-order formula of the form $A(\vec{t})$, where $A \in \Sigma_P$ and $\vec{t}$ is a vector of the terms $t_1, \ldots, t_{ar(A)}$. A *rule* r is an implication of the form

$$B_1 \wedge \ldots \wedge B_n \wedge \mathbf{not}\ B_{n+1} \wedge \ldots \wedge \mathbf{not}\ B_m \rightarrow H_1 \wedge \ldots \wedge H_\ell \tag{1}$$

where $H_1, \ldots, H_\ell$ are atoms, $B_1, \ldots, B_m$ are atoms different from $\bot$, $m \geq 0$, and $\ell > 0$. Let $head(r) = \{H_i\}_{1 \leq i \leq \ell}$, $body^+(r) = \{B_i\}_{1 \leq i \leq n}$, $body^-(r) = \{B_i\}_{n < i \leq m}$, and $body(r) = body^+(r) \cup body^-(r)$. A rule r is *safe* if every variable that occurs in $head(r)$ also occurs in $body^+(r)$. If $body(r) = \emptyset$ and r is safe, then r is a *fact*. We denote with $head_P(r)$, $body^+_P(r)$, $body^-_P(r)$, and $body_P(r)$ the set of predicates that occur in $head(r)$, $body^+(r)$, $body^-(r)$, and $body(r)$, respectively. A rule r is *function-free* if no function symbols occur in r. A *logic program* P is a set of rules. A logic program P is *negation-free* if, for each rule $r \in P$, we have $body^-(r) = \emptyset$.

Given a logic program P, $HU(P)$ is the set of all terms that can be formed using the constants and the function symbols from P (w.l.o.g. we assume that P contains at least one constant). If no variables occur in an atom (rule), then the atom (rule) is *ground*. Given a logic program P, the set $HB(P)$ is the set of all ground atoms constructed using the terms in $HU(P)$ and the predicates occuring in P. The grounding of a rule r w.r.t. a set of terms T is the set of rules obtained by substituting the variables of r by the terms of T in all possible ways. Given a logic program P, the program $ground(P)$ is obtained from P by replacing each rule $r \in P$ with its grounding w.r.t. $HU(P)$.

Let $I \subseteq HB(P)$ be a set of ground atoms. Then, I *satisfies* a ground rule r if $\mathit{body}^+(r) \subseteq I$ and $\mathit{body}^-(r) \cap I = \emptyset$ imply $\mathit{head}(r) \subseteq I$. Furthermore, I is a model of a (not necessarily ground) program P, written $I \models P$, if $\bot \notin I$ and I satisfies each rule $r \in \mathit{ground}(P)$. Given a negation-free program P, set I is a *minimal model* of P if $I \models P$ and no $I' \subsetneq I$ exists such that $I' \models P$. The Gelfond-Lifschitz reduct P^I of a logic program P w.r.t I is obtained from $\mathit{ground}(P)$ by removing each rule r such that $\mathit{body}^-(r) \cap I \neq \emptyset$, and removing all atoms $\mathbf{not}\ B_i$ in all the remaining rules. A set I is a *stable* model of P if I is a minimal model of P^I. Given a fact A, we write $P \models A$ if $A \in I$ for each stable model I of P; otherwise, we write $P \not\models A$.

A substitution is a partial mapping of variables to ground terms. The result of applying a substitution θ to a term, atom, or a set of atoms M is written as $M\theta$ and is defined as usual. Let P be a logic program in which no predicate occurring in the head of a rule in P also occurs negated in the body of a (possibly different) rule in P. Operator T_P applicable to a set of facts X is defined as follows:

$$\begin{aligned} T_P(X) = X \cup \{h\theta \mid\ & h \in \mathit{head}(r),\ \ r \in P,\ \ \theta \text{ maps the variables of } r \text{ to} \\ & HU(P \cup X) \text{ such that } \mathit{body}^+(r)\theta \subseteq X \text{ and } \mathit{body}^-(r)\theta \cap X = \emptyset\} \end{aligned}$$

Let $T_P^0 = \emptyset$, let $T_P^i = T_P(T_P^{i-1})$ for $i \geq 1$, and let $T_P^\infty = \bigcup_{i=1}^{\infty} T_P^i$. Clearly, $T_P^i \subseteq T_P^{i+1}$ for each $i \geq 0$. Furthermore, such P has at most one stable model [3], and T_P^∞ is the stable model of P if and only if $\bot \notin T_P^\infty$.

3 Motivating Application

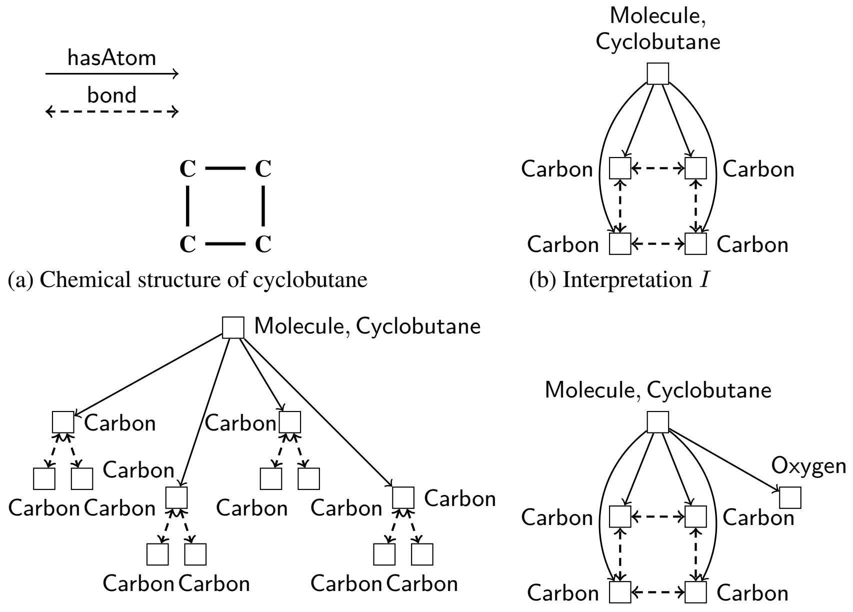
We next motivate our work using examples from the chemical Semantic Web application mentioned in the introduction. The goal of this application is to automatically classify chemical entities based on descriptions of their properties and structure. Unfortunately, as discussed in [20], OWL cannot describe cyclic structures with sufficient precision. This causes problems when modelling chemical compounds since molecules often have cyclic parts. For example, the cyclobutane molecule contains four carbon atoms connected in a ring,[1] as shown in Figure 1(a). One might try to represent this structure using the following OWL axiom:

$$\mathsf{Cyclobutane} \sqsubseteq \mathsf{Molecule}\ \sqcap = 4\,\mathsf{hasAtom}.[\mathsf{Carbon} \sqcap (= 2\,\mathsf{bond}.\mathsf{Carbon})]$$

This axiom is satisfied in first-order interpretations I and I' shown in Figures 1(b) and 1(c), respectively; however, only interpretation I correctly reflects the structure of cyclobutane. Furthermore, interpretation I' cannot be ruled out by writing additional axioms: OWL has a variant of the *tree-model property*, so each satisfiable TBox has at least one tree-shaped interpretation. This can prevent the entailment of certain desired consequences. For example, one cannot define the class of molecules containing four-membered rings that will be correctly identified as a superclass of cyclobutane.

The formalism from [20] addresses this problem by augmenting an OWL ontology with a set of rules and a set of *description graphs* (DGs), where each DG describes a

[1] In fact, cyclobutane also contains two hydrogens attached to each carbon; however, hydrogen atoms are commonly left implicit to simplify the presentation.



(a) Chemical structure of cyclobutane (b) Interpretation I

(c) Interpretation I' (d) Interpretation I''

Fig. 1. The chemical structure and the models of cyclobutane

complex object by means of a directed labeled graph. To avoid misunderstandings, we refer to the formalism from [20] as DGDL (Description Graph Description Logics), and to the formalism presented in this paper as DGLP (Description Graph Logic Programs). Thus, cyclobutane can be described using the DG shown in Figure 2(a). The first-order semantics of DGDL ontologies ensures that all models of an ontology correctly represent the DG structure; for example, interpretation I' from Figure 1(c) does not satisfy the DG in Figure 1(a). Nevertheless, the first-order models of DGDL ontologies can still be insufficiently precise. For example, the interpretation I'' shown in Figure 1(d) satisfies the definition of cyclobutane under the semantics of DGDL ontologies. We next show how the presence of models with excess information can restrict entailments.[2]

One might describe the class of hydrocarbon molecules (i.e., molecules consisting exclusively of hydrogens and carbons) using axiom (2). One would expect the definition of cyclobutane (as given in a DGDL ontology) and (2) to imply subsumption (3).

$$\mathsf{Molecule} \sqcap \forall \mathsf{hasAtom}.(\mathsf{Carbon} \sqcup \mathsf{Hydrogen}) \sqsubseteq \mathsf{Hydrocarbon} \tag{2}$$

$$\mathsf{Cyclobutane} \sqsubseteq \mathsf{Hydrocarbon} \tag{3}$$

This, however, is not the case, since interpretation I'' does not satisfy axiom (3). One might preclude the existence of extra atoms by adding cardinality restrictions requiring

[2] Krötzsch et al. [13] also suggest an extension of OWL 2 for the representation of graph-like structures. As we show next, the first-order semantics of this formalism exhibits the same problems as that of DGDL ontologies.

each cyclobutane to have exactly four atoms. Even so, axiom (3) would not be entailed because of a model similar to I, but where one carbon atom is also an oxygen atom. One could eliminate such models by introducing disjointness axioms for all chemical elements. Such gradual circumscription of models, however, is not an adequate solution, as one can always think of additional information that needs to be ruled out [18].

In order to address such problems, we present a novel expressive formalism that we call Description Graph Logic Programs (DGLP). DGLP ontologies are similar to DGDL ontologies in that they extend OWL ontologies with DGs and rules. In our case, however, the ontology is restricted to OWL 2 RL so that the ontology can be translated into rules [9]. We give semantics to our formalism by translating DGLP ontologies into logic programs with function symbols. As is common in logic programming, the translation is interpreted under *stable* models. Consequently, interpretations such as I'' are not stable models of the DG in Figure 2(a), and hence subsumption (3) is entailed.

Logic programs with function symbols can axiomatise infinite non-tree-like structures, so reasoning with DGLP ontologies is trivially undecidable [4]. Our goal, however, is not to model arbitrarily large structures, but to describe complex objects up to a certain level of granularity. For example, acetic acid has a carboxyl part, and carboxyl has a hydroxyl part, but hydroxyl does not have an acetic acid part (see Fig. 3(a)).

In Section 5 we exploit this intuition and present a condition that ensures decidability. In particular, we require the modeller to specify an ordering on DGs that, intuitively, describes which DGs are allowed to imply existence of other DGs. Using a suitable test, one can then check whether implications between DGs are acyclic and hence whether DGs describe structures of bounded size only. The resulting *semantic* acyclicity condition allows for the modelling of naturally-arising molecular structures, such as acetic acid, that would be ruled out by existing syntax-based acyclicity conditions [6, 16].

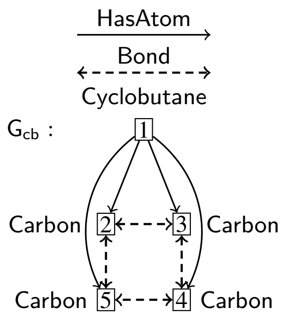
4 Description Graph Logic Programs

We now present the DGLP formalism in detail. We start by defining a slightly modified notion of description graphs; compared to the definition from [20], the new definition allows for only a single *start predicate* instead of a set of *main concepts*, and it includes a graph *mode* that allows for the automatic generation of 'recognition' rules—something that had to be introduced 'by hand' in [20].

Definition 1 (Description Graph). *A* description graph $G = (V, E, \lambda, A, m)$ *is a directed labeled graph where*

- $V = \{1, \ldots, n\}$ *is a nonempty set of vertices,*
- $E \subseteq V \times V$ *is a set of edges,*
- λ *assigns a set of unary predicates* $\lambda(v) \subseteq \Sigma_P$ *to each vertex* $v \in V$ *and a set of binary predicates* $\lambda(v_1, v_2) \subseteq \Sigma_P$ *to each edge* $(v_1, v_2) \in E$,
- $A \in \Sigma_P$ *is a* start predicate *for* G *such that* $A \in \lambda(1)$*, and*
- $m \in \{\Rightarrow, \Leftarrow, \Leftrightarrow\}$ *is a* mode *for* G.

A description graph (DG) abstracts the structure of a complex object by means of a directed labeled graph. For example, Figure 2 illustrates a DG that represents the

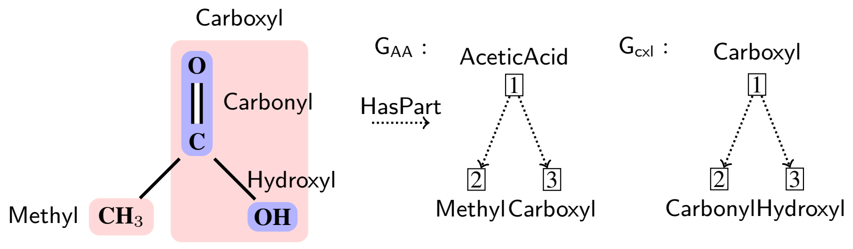


(a) Cyclobutane DG

$\mathsf{Cyclobutane(a)}$ $\mathsf{Molecule(a)}$
$\mathsf{G_{cb}(a, f_1^{G_{cb}}(a), f_2^{G_{cb}}(a), f_3^{G_{cb}}(a), f_4^{G_{cb}}(a))}$
$\mathsf{HasAtom(a, f_i^{G_{cb}}(a))}$ for each $\mathsf{i} \in \{1, 2, 3, 4\}$
$\mathsf{Carbon(f_i^{G_{cb}}(a))}$ for each $\mathsf{i} \in \{1, 2, 3, 4\}$
$\mathsf{Bond(f_i^{G_{cb}}(a), f_{i+1}^{G_{cb}}(a))}$ for each $\mathsf{i} \in \{1, 2, 3\}$
$\mathsf{Bond(f_4^{G_{cb}}(a), f_1^{G_{cb}}(a))}$
$\mathsf{MolWith4MemberedRing(a)}$
$\mathsf{Hydrocarbon(a)}$

(b) The stable model of $\mathsf{LP}(\mathcal{O})$

Fig. 2. Representing cyclobutane with DGLP



(a) Chemical graph of acetic acid (b) Acetic acid DG (c) Carboxyl DG

Fig. 3. The chemical graph of acetic acid and the $\mathsf{G_{AA}}$ and the $\mathsf{G_{cxl}}$ DGs

structure of a cyclobutane molecule. The start predicate of the graph (Cyclobutane in this case) corresponds to the name of the object that the graph describes. The mode determines whether a graph should be interpreted as an 'only if', 'if', or 'if and only if' statement. More precisely, $\Rightarrow$ means that each instance of the DG's start predicate implies the existence of a corresponding instantiation of the entire graph structure; $\Leftarrow$ means that an instantiation of a suitable graph structure is 'recognised' as an instance of the corresponding DG; and $\Leftrightarrow$ means both of the above.

Next we define *graph orderings*, which will play an important role in ensuring the decidability of DGLP.

Definition 2 (Graph Ordering). *A* graph ordering *on a set of description graphs DG is a transitive and irreflexive relation $\prec \; \subseteq DG \times DG$.*

Intuitively, a graph ordering specifies which DGs can imply the existence of instances of other DGs. For example, let $\mathsf{G_{AA}}$ be a graph that represents the structure of the acetic acid, and let $\mathsf{G_{cxl}}$ be a graph that represents the structure of the carboxyl group that is a substructure of acetic acid (see Figure 3); then, one might define $\prec$ such that $\mathsf{G_{AA}} \prec \mathsf{G_{cxl}}$, so that an acetic acid instance may imply the existence of a carboxyl group instance, but not vice versa. We are now ready to define DGLP ontologies.

Definition 3 (DGLP Ontology). *A* DGLP ontology $\mathcal{O} = \langle DG, \prec, R, F \rangle$ *is a quadruple where DG is a finite set of description graphs, $\prec$ is a graph ordering on DG, R is a finite set of function-free and safe rules, and F is a finite set of function-free facts.*

For the sake of simplicity, we do not explicitly include an OWL 2 RL TBox into the definition of DGLP ontologies: OWL 2 RL axioms can be translated into rules as shown in [9] and included in R, and datatypes can be handled as in [15]. Similarly, we could think of F as an OWL 2 ABox, as ABox assertions correspond directly to facts [9]. An example of a DGLP ontology is $\langle \{\mathsf{G_{AA}}, \mathsf{G_{cxl}}\}, \{(\mathsf{G_{AA}}, \mathsf{G_{cxl}})\}, \emptyset, \{\mathsf{AceticAcid(a)}\}\rangle$.

We next define the semantics of DGLP via a translation into logic programs. Since R and F are already sets of rules and $\prec$ serves only to check acyclicity, we only need to specify how to translate DGs into rules.

Definition 4 (Start, Layout, and Recognition Rule). *Let $G = (V, E, A, \lambda, m)$ be a description graph and let $f_1^G, \ldots, f_{|V|-1}^G$ be fresh distinct function symbols uniquely associated with G. The* start rule s_G*, the* layout rule ℓ_G*, and the* recognition rule r_G of *G are defined as follows:*

$$A(x) \rightarrow G(x, f_1^G(x), \ldots, f_{|V|-1}^G(x)) \qquad (s_\mathsf{G})$$

$$G(x_1, \ldots, x_{|V|}) \rightarrow \bigwedge_{i \in V, B \in \lambda(i)} B(x_i) \wedge \bigwedge_{\langle i,j \rangle \in E, R \in \lambda(i,j)} R(x_i, x_j) \qquad (\ell_\mathsf{G})$$

$$\bigwedge_{i \in V, B \in \lambda(i), B \neq A} B(x_i) \wedge \bigwedge_{\langle i,j \rangle \in E, R \in \lambda(i,j)} R(x_i, x_j) \rightarrow G(x_1, \ldots, x_{|V|}) \qquad (r_\mathsf{G})$$

The start and layout rules of a description graph serve to unfold the graph's structure. The function terms $f_1^G(x), \ldots, f_{|V|-1}^G(x)$ correspond to existential restrictions whose existentially quantified variables have been skolemised.

Example 1. The DG of cyclobutane from Figure 2 can be naturally represented by the existential restriction (4). The skolemised version of (4) is the start rule ($s_{\mathsf{G_{cb}}}$).

$$\mathsf{Cyclobutane(x)} \rightarrow \exists \mathsf{y_1, y_2, y_3, y_4} : \mathsf{G_{cb}(x, y_1, y_2, y_3, y_4)} \qquad (4)$$

$$\mathsf{Cyclobutane(x)} \rightarrow \mathsf{G_{cb}(x, f_1^{G_{cb}}(x), f_2^{G_{cb}}(x), f_3^{G_{cb}}(x), f_4^{G_{cb}}(x))} \qquad (s_{\mathsf{G_{cb}}})$$

The layout rule ($\ell_{\mathsf{G_{cb}}}$) encodes the edges and the labelling of the description graph.[3] Finally, the rule $r_{\mathsf{G_{cb}}}$ is responsible for identifying the cyclobutane structure:

$$\mathsf{G_{cb}(x_1, x_2, x_3, x_4, x_5)} \rightarrow \mathsf{Cyclobutane(x_1)} \wedge \bigwedge_{2 \leq i \leq 4} \mathsf{Bond(x_i, x_{i+1})} \wedge \mathsf{Bond(x_5, x_2)} \wedge \bigwedge_{2 \leq i \leq 5} \mathsf{HasAtom(x_1, x_i)} \wedge \bigwedge_{2 \leq i \leq 5} \mathsf{Carbon(x_i)} \qquad (\ell_{\mathsf{G_{cb}}})$$

$$\bigwedge_{2 \leq i \leq 5} \mathsf{HasAtom(x_1, x_i)} \wedge \bigwedge_{2 \leq i \leq 5} \mathsf{Carbon(x_i)} \wedge \bigwedge_{2 \leq i \leq 4} \mathsf{Bond(x_i, x_{i+1})} \wedge \mathsf{Bond(x_5, x_2)} \rightarrow \mathsf{G_{cb}(x_1, x_2, x_3, x_4, x_5)} \qquad (r_{\mathsf{G_{cb}}})$$

[3] In the rest of the paper for simplicity we assume that bonds are unidirectional.

Next, we define $\mathsf{Axioms}(DG)$, which is a logic program that encodes a set of DGs.

Definition 5 **($\mathsf{Axioms}(DG)$).** *For a description graph $G = (V, E, \lambda, A, m)$, the program $\mathsf{Axioms}(G)$ is the set of rules that contains the start rule s_G and the layout rule ℓ_G if $m \in \{\Rightarrow, \Leftrightarrow\}$, and the recognition rule r_G if $m \in \{\Leftarrow, \Leftrightarrow\}$. For a set of description graphs $DG = \{G_i\}_{1 \leq i \leq n}$, let $\mathsf{Axioms}(DG) = \bigcup_{G_i \in DG} \mathsf{Axioms}(G_i)$.*

For each DGLP ontology $\mathcal{O} = \langle DG, \prec, R, F \rangle$, we denote with $\mathsf{LP}(\mathcal{O})$ the program $\mathsf{Axioms}(DG) \cup R \cup F$. To check whether a class C is subsumed by a class D, we can proceed as in standard OWL reasoning: we assert $C(a)$ for a a fresh individual, and we check whether $D(a)$ is entailed.

Definition 6 **(Subsumption).** *Let $\mathcal{O}$ be a DGLP ontology, let C and D be unary predicates occurring in $\mathcal{O}$, and let a be a fresh individual not occurring in $\mathcal{O}$. Then, D* subsumes *C w.r.t. $\mathcal{O}$, written $\mathcal{O} \models C \sqsubseteq D$, if $\mathsf{LP}(\mathcal{O}) \cup \{C(a)\} \models D(a)$ holds.*

Example 2. We now show how a DGLP ontology can be used to obtain the inferences described in Section 3. Rule (r_1) encodes the class of four-membered ring molecules:

$$\mathsf{Molecule(x)} \wedge \bigwedge_{1 \leq \mathsf{i} \leq 4} \mathsf{HasAtom(x, y_i)} \wedge \bigwedge_{1 \leq \mathsf{i} \leq 3} \mathsf{Bond(y_i, y_{i+1})} \wedge \mathsf{Bond(y_4, y_1)}$$
$$\bigwedge_{1 \leq i < j \leq 4} \mathbf{not}\ \mathsf{y_i = y_j} \rightarrow \mathsf{MolWith4MemberedRing(x)} \qquad (r_1)$$

The use of the equality predicate $=$ in the body of r_1 does not require an extension to our syntax: if $=$ occurs only in the body and not in the head of the rules, then negation of equality can be implemented using a built-in predicate. In addition, we represent the class of hydrocarbons with rules (r_2) and (r_3).

$$\mathsf{Molecule(x)} \wedge \mathsf{HasAtom(x, y)} \wedge \mathbf{not}\mathsf{Carbon(y)} \wedge \mathbf{not}\mathsf{Hydrogen(y)} \rightarrow \mathsf{NHC(x)} \qquad (r_2)$$
$$\mathsf{Molecule(x)} \wedge \mathbf{not}\ \mathsf{NHC(x)} \rightarrow \mathsf{HydroCarbon(x)} \qquad (r_3)$$
$$\mathsf{Cyclobutane(x)} \rightarrow \mathsf{Molecule(x)} \qquad (r_4)$$

Finally, we state that cyclobutane is a molecule using (r_4) that corresponds to the OWL 2 RL axiom $\mathsf{Cyclobutane} \sqsubseteq \mathsf{Molecule}$. Let $DG = \{\mathsf{G_{cb}}\}$, let $\prec = \emptyset$, let $R = \{r_i\}_{i=1}^{4}$, let $F = \{\mathsf{Cyclobutane(a)}\}$, and let $\mathcal{O} = \langle DG, \prec, R, F \rangle$. Figure 2(b) shows the only stable model of $\mathsf{LP}(\mathcal{O})$ by inspection of which we see that $\mathsf{LP}(\mathcal{O}) \models \mathsf{Hydrocarbon(a)}$ and $\mathsf{LP}(\mathcal{O}) \models \mathsf{MolWith4MemberedRing(a)}$, as expected.

5 Semantic Acyclicity

Deciding whether a logic program with function symbols entails a given fact is known to be undecidable in general [4]. This problem is closely related to the problem of reasoning with datalog programs with existentially quantified rule heads (known as tuple-generating dependencies or tgds) [5]. For such programs, conditions such as weak acyclicity [6] or super-weak acyclicity [16] ensure the termination of bottom-up reasoning algorithms. Roughly speaking, these conditions examine the syntactic structure of the program's rules and check whether values created by a rule's head can be propagated so as to eventually satisfy the premise of the same rule. Due to the similarity between tgds and our formalism, such conditions can also be applied to DGLP ontologies. These conditions, however, may overestimate the propagation of values introduced by existential quantification and thus rule out unproblematical programs that generate only finite structures. As we show in Example 4, this turns out to be the case for programs that naturally arise from DGLP representations of molecular structures.

To mitigate this problem, we propose a new *semantic* acyclicity condition. The idea is to detect repetitive construction of DG instances by checking the entailment of a special propositional symbol Cycle. To avoid introducing an algorithm-specific procedural definition, our notion is declarative. The graph ordering $\prec$ of a DGLP ontology $\mathcal{O}$ is used to extend $\mathsf{LP}(\mathcal{O})$ with rules that derive Cycle whenever an instance of a DG G_1 implies existence of an instance of a DG G_2 but $G_1 \not\prec G_2$.

Definition 7 **($\mathsf{Check}(\mathcal{O})$).** *Let $G_i = (V_i, E_i, \lambda_i, A_i, m_i)$, $i \in \{1, 2\}$ be two description graphs. We define* $\mathsf{ChkPair}(G_1, G_2)$ *and* $\mathsf{ChkSelf}(G_i)$ *as follows:*

$$\mathsf{ChkPair}(G_1, G_2) = \{G_1(x_1, \ldots, x_{|V_1|}) \wedge A_2(x_k) \rightarrow \mathsf{Cycle} \mid 1 \leq k \leq |V_1|\} \quad (5)$$

$$\mathsf{ChkSelf}(G_i) = \{G_i(x_1, \ldots, x_{|V_i|}) \wedge A_i(x_k) \rightarrow \mathsf{Cycle} \mid 1 < k \leq |V_i|\} \quad (6)$$

Let $DG = \{G_i\}_{1 \leq i \leq n}$ be a set of description graphs and let $\prec$ be a graph ordering on DG. We define $\mathsf{Check}(DG, \prec)$ *as follows:*

$$\mathsf{Check}(DG, \prec) = \bigcup_{i,j \in \{1,\ldots,n\},\ i \neq j,\ G_i \not\prec G_j} \mathsf{ChkPair}(G_i, G_j) \cup \bigcup_{1 \leq i \leq n} \mathsf{ChkSelf}(G_i)$$

For a DGLP ontology $\mathcal{O} = \langle DG, \prec, R, F\rangle$, we define $\mathsf{Check}(\mathcal{O}) = \mathsf{Check}(DG, \prec)$.

Example 3. Figure 3(a) shows the structure of acetic acid molecules and the parts they consist of. In this example, however, we focus on the description graphs for acetic acid ($\mathsf{G_{AA}}$) and carboxyl ($\mathsf{G_{cxl}}$), which are shown in Figures (3)(b) and (3)(c), respectively. Since an instance of acetic acid implies the existence of an instance of a carboxyl, but not vice versa, we define our ordering as $\mathsf{G_{AA}} \prec \mathsf{G_{cxl}}$. Thus, for $DG = \{\mathsf{G_{AA}}, \mathsf{G_{cxl}}\}$ and $\prec = \{(\mathsf{G_{AA}}, \mathsf{G_{cxl}})\}$, set $\mathsf{Check}(DG, \prec)$ contains the following rules:

$$\mathsf{G_{cxl}}(\mathsf{x_1}, \mathsf{x_2}, \mathsf{x_3}) \wedge \mathsf{AceticAcid}(\mathsf{x_i}) \rightarrow \mathsf{Cycle} \qquad \text{for } 1 \leq \mathsf{i} \leq 3$$

$$\mathsf{G_{AA}}(\mathsf{x_1}, \mathsf{x_2}, \mathsf{x_3}) \wedge \mathsf{AceticAcid}(\mathsf{x_i}) \rightarrow \mathsf{Cycle} \qquad \text{for } 2 \leq \mathsf{i} \leq 3$$

$$\mathsf{G_{cxl}}(\mathsf{x_1}, \mathsf{x_2}, \mathsf{x_3}) \wedge \mathsf{Carboxyl}(\mathsf{x_i}) \rightarrow \mathsf{Cycle} \qquad \text{for } 2 \leq \mathsf{i} \leq 3$$

We next define when a DGLP ontology is semantically acyclic. Intuitively, this condition will ensure that the evaluation of $\mathsf{LP}(\mathcal{O})$ does not generate a chain of description graph instances violating the DG ordering.

Definition 8. *A DGLP ontology* $\mathcal{O}$ *is said to be* semantically acyclic *if and only if* $\mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O}) \not\models \mathsf{Cycle}$.

Example 4. Let $DG = \{\mathsf{G}_{\mathsf{AA}}, \mathsf{G}_{\mathsf{cxl}}\}$ with $m_{\mathsf{AA}} = m_{\mathsf{cxl}} = \Leftrightarrow$, let $\prec = \{(\mathsf{G}_{\mathsf{AA}}, \mathsf{G}_{\mathsf{cxl}})\}$, let $F = \{\mathsf{AceticAcid(a)}\}$, and let $\mathcal{O} = \langle DG, \prec, \emptyset, F\rangle$. By Definition 5, logic program $\mathsf{LP}(\mathcal{O})$ contains F and the following rules (HP abbreviates HasPart):

$$\mathsf{AceticAcid(x)} \to \mathsf{G}_{\mathsf{AA}}(\mathsf{x}, \mathsf{f}_1(\mathsf{x}), \mathsf{f}_2(\mathsf{x}))$$
$$\mathsf{G}_{\mathsf{AA}}(\mathsf{x},\mathsf{y},\mathsf{z}) \to \mathsf{AceticAcid(x)} \wedge \mathsf{Methyl(y)} \wedge \mathsf{Carboxyl(z)} \wedge \mathsf{HP(x,y)} \wedge \mathsf{HP(x,z)}$$
$$\mathsf{Methyl(y)} \wedge \mathsf{Carboxyl(z)} \wedge \mathsf{HP(x,y)} \wedge \mathsf{HP(x,z)} \to \mathsf{G}_{\mathsf{AA}}(\mathsf{x},\mathsf{y},\mathsf{z})$$
$$\mathsf{Carboxyl(x)} \to \mathsf{G}_{\mathsf{cxl}}(\mathsf{x}, \mathsf{g}_1(\mathsf{x}), \mathsf{g}_2(\mathsf{x}))$$
$$\mathsf{G}_{\mathsf{cxl}}(\mathsf{x},\mathsf{y},\mathsf{z}) \to \mathsf{Carboxyl(x)} \wedge \mathsf{Carbonyl(y)} \wedge \mathsf{Hydroxyl(z)} \wedge \mathsf{HP(x,y)} \wedge \mathsf{HP(x,z)}$$
$$\mathsf{Carbonyl(y)} \wedge \mathsf{Hydroxyl(z)} \wedge \mathsf{HP(x,y)} \wedge \mathsf{HP(x,z)} \to \mathsf{G}_{\mathsf{cxl}}(\mathsf{x},\mathsf{y},\mathsf{z})$$

Let also $\mathsf{Check}(\mathcal{O}) = \mathsf{Check}(DG, \prec)$ as defined in Example 3. The stable model of $P = \mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O})$ can be computed using the T_P operator:

$$T_P^{\infty} = \{\mathsf{AceticAcid(a)}, \mathsf{G}_{\mathsf{AA}}(\mathsf{a}, \mathsf{f}_1(\mathsf{a}), \mathsf{f}_2(\mathsf{a})), \mathsf{HP}(\mathsf{a}, \mathsf{f}_1(\mathsf{a})), \mathsf{HP}(\mathsf{a}, \mathsf{f}_2(\mathsf{a})),$$
$$\mathsf{Methyl}(\mathsf{f}_1(\mathsf{a})), \mathsf{Carboxyl}(\mathsf{f}_2(\mathsf{a})), \mathsf{G}_{\mathsf{cxl}}(\mathsf{f}_2(\mathsf{a}), \mathsf{g}_1(\mathsf{f}_2(\mathsf{a})), \mathsf{g}_2(\mathsf{f}_2(\mathsf{a}))), \mathsf{Carbonyl}(\mathsf{g}_1(\mathsf{f}_2(\mathsf{a}))),$$
$$\mathsf{Hydroxyl}(\mathsf{g}_2(\mathsf{f}_2(\mathsf{a}))), \mathsf{HP}(\mathsf{f}_2(\mathsf{a}), \mathsf{g}_1(\mathsf{f}_2(\mathsf{a}))), \mathsf{HP}(\mathsf{f}_2(\mathsf{a}), \mathsf{g}_2(\mathsf{f}_2(\mathsf{a})))\}$$

Since Cycle is not in the (only) stable model of P, we have $P \not\models \mathsf{Cycle}$ and $\mathcal{O}$ is semantically acyclic. However, P is neither weakly [6] nor super-weakly acyclic [16]. This, we believe, justifies the importance of semantic acyclicity for our applications.

Example 5 shows how functions may trigger infinite generation of DG instances.

Example 5. Let $\mathcal{O} = \langle\{\mathsf{G}\}, \emptyset, \{\mathsf{B(x)} \to \mathsf{A(x)}\}, \{\mathsf{A(a)}\}\rangle$ be a DGLP ontology where G is such that Axioms(G) is as follows:

$$\mathsf{Axioms(G)} = \{\mathsf{A(x)} \to \mathsf{G(x, f(x))},\quad \mathsf{G}(\mathsf{x}_1, \mathsf{x}_2) \to \mathsf{A}(\mathsf{x}_1) \wedge \mathsf{B}(\mathsf{x}_2) \wedge \mathsf{R}(\mathsf{x}_1, \mathsf{x}_2)\}$$

For $\mathsf{Check}(\mathcal{O}) = \{\mathsf{G}(\mathsf{x}_1, \mathsf{x}_2) \wedge \mathsf{A}(\mathsf{x}_2) \to \mathsf{Cycle}\}$ and $P = \mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O})$, we have $T_P^{\infty} = \{\mathsf{A(a)}, \mathsf{G(a, f(a))}, \mathsf{R(a, f(a))}, \mathsf{B(f(a))}, \mathsf{A(f(a))}, \mathsf{Cycle}, \ldots\}$. Now $\mathcal{O}$ is not semantically acyclic because $\mathsf{Cycle} \in T_P^{\infty}$, which indicates that T_P can be applied to F in a repetitive way without terminating.

Semantic acyclicity is a sufficient, but not a necessary termination condition: bottom-up evaluation of $\mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O})$ can terminate even if $\mathcal{O}$ is not semantically acyclic.

Example 6. Let $\mathcal{O} = \langle\{\mathsf{G}\}, \emptyset, \{\mathsf{R}(\mathsf{x}_1, \mathsf{x}_2) \wedge \mathsf{C}(\mathsf{x}_1) \to \mathsf{A}(\mathsf{x}_2)\}, \{\mathsf{A(a)}, \mathsf{C(a)}\}\rangle$ be a DGLP ontology where G, $\mathsf{Check}(\mathcal{O})$, and P are defined as in Example 5. One can see that $\{\mathsf{A(a)}, \mathsf{C(a)}, \mathsf{G(a, f(a))}, \mathsf{R(a, f(a))}, \mathsf{B(f(a))}, \mathsf{A(f(a))}, \mathsf{Cycle}, \mathsf{G(f(a), f(f(a)))}, \mathsf{B(f(f(a)))}, \mathsf{R(f(a), f(f(a)))}\}$ is the stable model of P computable by finitely many applications of the T_P operator; however, $\mathcal{O}$ is not semantically acyclic since $\mathsf{Cycle} \in T_P^{\infty}$.

6 Reasoning with Negation-Free DGLP ontologies

In the present section, we consider the problem of reasoning with a DGLP ontology $\mathcal{O} = \langle DG, \prec, R, F\rangle$ where R is negation-free. Intuitively, one can simply apply the T_P operator to $P = \mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O})$ and compute $T_P^1, T_P^2, \ldots, T_P^i$ and so on. By Theorem 7, for some i we will either reach a fixpoint or derive Cycle. In the former case, we have the stable model of $\mathcal{O}$ (if $\bot \notin T_P^i$), which we can use to decide the relevant reasoning problems; in the latter case, we know that $\mathcal{O}$ is not acyclic.

Theorem 7. *Let $\mathcal{O} = \langle DG, \prec, R, F\rangle$ be a DGLP ontology with R negation-free, and let $P = \mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O})$. Then,* $\mathsf{Cycle} \in T_P^i$ *or $T_P^{i+1} = T_P^i$ for some $i \geq 1$.*

By Theorem 7, checking the semantic acyclicity of $\mathcal{O}$ is thus decidable. If the stable model of $\mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O})$ is infinite, then Cycle is derived; however, the inverse does not hold as shown in Example 6. Furthermore, a stable model of $\mathsf{LP}(\mathcal{O})$, if it exists, is clearly contained in the stable model of $\mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O})$, and the only possible difference between the two stable models is for the latter to contain Cycle.

7 Reasoning with Stratified Negation-as-Failure

We now extend the reasoning algorithm from Section 6 to the case of DGLP ontologies $\mathcal{O} = \langle DG, \prec, R, F\rangle$ where R contains stratified negation-as-failure. We start by recapitulating several definitions. Let P be a logic program. A *stratification* of P is a mapping $\sigma : P \to \mathbb{N}$ such that for each rule $r \in P$ the following conditions hold:

- if $B \in body_P^+(r)$, then $\sigma(r') \leq \sigma(r)$ for each $r' \in P$ with $B \in head_P(r')$; and
- if $B \in body_P^-(r)$, then $\sigma(r') < \sigma(r)$ for each $r' \in P$ with $B \in head_P(r')$.

A logic program P is *stratifiable* if there exists a stratification of P. Moreover, a partition $P_1, \ldots, P_n$ of P is a *stratification partition* of P w.r.t. σ if, for each $r \in P$, we have $r \in P_{\sigma(r)}$. The sets $P_1, \ldots, P_n$ are called the *strata* of P. Let $U_{P_0}^\infty = T_{P_1}^1$, $U_{P_j}^i = T_{P_j}^i(U_{P_{j-1}}^\infty)$ for $1 \leq j \leq n$ and $i \geq 1$ and $U_{P_j}^\infty = T_{P_j}^\infty(U_{P_{j-1}}^\infty)$. The stable model of P is given by $U_{P_n}^\infty$.

Example 8. Take $\mathcal{O}$ from Example 2 and let $P = \mathsf{LP}(\mathcal{O})$. The mapping $\sigma : P \to \mathbb{N}$ such that we have $\sigma(\mathsf{Cyclobutane(a)}) = \sigma(s_{\mathsf{G_{cb}}}) = \sigma(\ell_{\mathsf{G_{cb}}}) = \sigma(r_{\mathsf{G_{cb}}}) = \sigma(r_4) = 1$, $\sigma(r_1) = \sigma(r_2) = 2$, and $\sigma(r_3) = 3$, is a stratification of P. A stratification partition of P w.r.t. σ is $\{\mathsf{Cyclobutane(a)}, s_{\mathsf{G_{cb}}}, \ell_{\mathsf{G_{cb}}}, r_{\mathsf{G_{cb}}}, r_4\}$, $\{r_1, r_2\}$, $\{r_3\}$.

Next we introduce the notion of a DG-stratification, which ensures that the cycle detection rules are assigned to the strata containing the relevant start rules of DGs.

Definition 9 (DG-stratification). *Let $\mathcal{O} = \langle DG, \prec, R, F\rangle$ be a DGLP ontology, let $P = \mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O})$ and let $P_1, \ldots, P_n$ be a stratification partition of P w.r.t. some stratification σ of P. Then, σ is a* DG-stratification *if*

- *for each $G_1, G_2 \in DG$ such that $G_1 \neq G_2$, $G_1 \not\prec G_2$, and $\{s_{G_1}, s_{G_2}\} \subseteq P_i$, we have* $\mathsf{ChkPair}(G_1, G_2) \subseteq P_i$*, and*

– *for each $G \in DG$ such that $s_G \in P_i$, we have* $\mathsf{ChkSelf}(G) \subseteq P_i$.

Example 9. Let P be the program and let σ be the stratification defined in Example 8. Then $\mathsf{ChkSelf}(\mathsf{G_{cb}}) = \{\mathsf{G_{cb}}(\mathsf{x_1}, \mathsf{x_2}, \mathsf{x_3}, \mathsf{x_4}, \mathsf{x_5}) \wedge \mathsf{Cyclobutane}(\mathsf{x_i}) \rightarrow \mathsf{Cycle}\}_{2 \leq \mathsf{i} \leq 5}$; also let $P' = P \cup \mathsf{ChkSelf}(\mathsf{G_{cb}})$. The stratification σ' where $\sigma'(r) = \sigma(r)$ for $r \in P$ and $\sigma'(r) = 1$ for $r \in \mathsf{ChkSelf}(\mathsf{G_{cb}})$ is a DG-stratification of P'.

The following result shows that, as long as $\mathsf{LP}(\mathcal{O})$ is stratified, one can always assign the cycle checking rules in $\mathsf{Check}(\mathcal{O})$ to the appropriate strata and thus obtain a DG-stratification of $\mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O})$. In other words, for the reasoning algorithm presented in this section to be applicable, it is sufficient to find a stratification of $\mathsf{LP}(\mathcal{O})$.

Lemma 1. *Let $\mathcal{O} = \langle DG, \prec, R, F \rangle$ be a DGLP ontology. If σ is a stratification of* $\mathsf{LP}(\mathcal{O})$*, then σ can be extended to a DG-stratification σ' of* $\mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O})$.

The following theorem implies that, given a stratifiable DGLP ontology, we can decide whether the ontology is semantically acyclic, and if so, we can compute its stable model and thus solve all relevant reasoning problems.

Theorem 10. *Let $\mathcal{O}$ be a DGLP ontology and* $P = \mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O})$. *If $P_1, \ldots, P_n$ is a stratification partition of P w.r.t. a DG-stratification of P, then, for each j with $1 \leq j \leq n$, there exists $i \geq 1$ such that* $\mathsf{Cycle} \in U^i_{P_j}$, *or $U^{i+1}_{P_j} = U^i_{P_j}$ and $U^i_{P_j}$ is finite.*

8 Implementation Results and Discussion

In order to test the applicability of our approach in practice, we have developed a prototypical implementation based on the XSB system.[4] We used the XSB engine because it supports function symbols, which are needed in the transformation of DGLP ontologies to logic programs. XSB does not compute the stable model(s) of a logic program, but implements query answering as the main reasoning task; by Definitions 8 and 6, this is sufficient for checking acyclicity and subsumption.

To obtain test data, we extracted from the ChEBI ontology seven DGLP ontologies $\mathcal{O}_i = \langle DG_i, \prec, R, F_i \rangle$, $1 \leq i \leq 7$. Each set of description graphs DG_i contained $i \cdot 10$ description graphs, each describing the structure of a particular molecule; the start predicate of each DG is the molecule's name; and the mode of the DG is $\Rightarrow$. ChEBI describes about 24,000 molecules in total;[5] however, due to the prototypical nature of our implementation we can currently handle only a small subset of ChEBI. Also, since DG_i does not model DGs that imply existence of other DGs, for each $\mathcal{O}_i$ we set $\prec = \emptyset$. Each $\mathcal{O}_i$ contains the same set of rules R that models general chemical knowledge, such as 'cyclobutane is a molecule' and 'a single bond is a bond'; furthermore, R contains rules that classify molecules into five classes $\mathsf{T_1}$–$\mathsf{T_5}$ shown in Figure 4. Finally, each F_i contains a fact of the form $\mathsf{M_j}(\mathsf{m_j})$ for each molecule predicate $\mathsf{M_j}$ and fresh individual $\mathsf{m_j}$. Thus, F_i 'instantiates' all description graphs in DG_i, so we can check $\mathcal{O}_i \models \mathsf{M_j} \sqsubseteq \mathsf{T_k}$ by equivalently checking $\mathsf{LP}(\mathcal{O}_i) \models \mathsf{T_k}(\mathsf{m_j})$. All test ontologies are available online.[6]

[4] http://xsb.sourceforge.net/
[5] http://www.ebi.ac.uk/chebi/statisticsForward.do
[6] http://www.cs.ox.ac.uk/isg/people/despoina.magka/tools/ChEBIClassifier.tar.gz

No mol.	No rules	Loading time	T_1	T_2	T_3	T_4	T_5	Total time
10	1417	2.08	< 0.01	< 0.01	< 0.01	0.36	0.02	2.47
20	5584	8.35	< 0.01	< 0.01	0.02	2.07	0.21	10.66
30	8994	11.35	0.01	< 0.01	0.03	2.23	0.23	13.85
40	14146	16.14	0.01	< 0.01	0.04	2.58	0.29	19.06
50	21842	23.11	0.01	0.01	0.06	3.55	0.41	27.15
60	55602	168.71	0.04	0.02	0.51	109.88	21.68	300.84
70	77932	239.06	0.06	0.03	0.75	172.14	35.08	447.12

T_1: hydrocarbons, T_2: inorganic molecules, T_3: molecules with exactly two carbons, T_4: molecules with a four-membered ring, T_5:molecules with a benzene ring

Fig. 4. Evaluation results

We conducted the following tests for each $\mathcal{O}_i$. First, we loaded $\mathsf{LP}(\mathcal{O}_i)$ into XSB. Second, we checked whether $\mathcal{O}_i$ is acyclic. Third, for each class of molecules $\mathsf{T_k}$, $1 \leq k \leq 5$ shown in Table 4, we measured the time needed to test $\mathcal{O}_i \models \mathsf{M_j} \sqsubseteq \mathsf{T_k}$ for each molecule $\mathsf{M_j}$ in $\mathcal{O}_i$; each test of the latter form was performed by checking $\mathsf{LP}(\mathcal{O}_i) \models \mathsf{T_k}(\mathsf{m_j})$. Figure 4 summarises the loading and classification times; the times needed to check acyclicity are not shown since they were under a second in all cases. The experiments were performed on a desktop computer with 3.7 GB of RAM and Intel Core$^{\mathrm{TM}}$ 2 Quad Processors running at 2.5 GHz and 64 bit Linux.

Our tests have shown all of the ontologies to be acyclic. Furthermore, all tests have correctly classified the relevant molecules into appropriate molecule classes; for example, we were able to conclude that acetylene has exactly two carbons, that cyclobutane has a four-membered ring, and that dinitrogen is inorganic. Please note that none of these inferences can be derived using the approach from [20] due to the lack of negation-as-failure, or using OWL only due to its tree-model property.

All tests were accomplished in a reasonable amount of time: no test required more than a few minutes. Given the prototypical character of our application, we consider these results to be encouraging and we take them as evidence of the practical feasibility of our approach. The most time-intensive test was T_4, which identified molecules containing a four-membered ring. We do not consider this surprising, given that the rule for recognising T_4 contains many atoms in the body and thus requires evaluating a complex join. We noticed, however, that reordering the atoms in the body of the rule significantly reduces reasoning time. Thus, trying to determine an appropriate ordering of rule atoms via join-ordering optimisations, such as the one used for query optimisation in relational databases, might be a useful technique in an optimised DGLP implementation.

9 Conclusions and Future Work

In this paper, we presented an expressive and decidable formalism for the representation of objects with complex structure; additionally, our preliminary evaluation provides evidence that our formalism is practically feasible. In our future work, we shall modify our approach in order to avoid the explicit definition of graph ordering on behalf of the user; furthermore, we plan to investigate whether semantic acyclicity can be combined with other conditions (such as [2]) in order to obtain a more general acyclicity check. Finally, we shall optimise our prototypical implementation in order to obtain a fully-scalable chemical classification system.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, and Applications. CUP (2007)
2. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: On rules with existential variables: Walking the decidability line. Artif. Intell. 175(9-10), 1620–1654 (2011)
3. Baral, C., Gelfond, M.: Logic Programming and Knowledge Representation. Journal of Logic Programming 19, 73–148 (1994)
4. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: Proc. ICALP. pp. 73–85 (1981)
5. Cali, A., Gottlob, G., Lukasiewicz, T., Marnette, B., Pieris, A.: Datalog+/-: A family of logical knowledge representation and query languages for new applications. In: LICS (2010)
6. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and Query Answering. In: ICDT. pp. 207–224 (2003)
7. Grau, B.C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: The next step for OWL. J. Web Sem. 6(4), 309–322 (2008)
8. Graves, H.: Representing Product Designs Using a Description Graph Extension to OWL 2. In: Proc. of the 5th OWLED Workshop (2009)
9. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description Logic Programs: Combining Logic Programs with Description Logic. In: WWW (2003)
10. Hastings, J., Dumontier, M., Hull, D., Horridge, M., Steinbeck, C., Sattler, U., Stevens, R., Hörne, T., Britz, K.: Representing Chemicals using OWL, Description Graphs and Rules. In: OWLED (2010)
11. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission (21 May 2004), http://www.w3.org/Submission/SWRL/
12. Krötzsch, M., Rudolph, S., Hitzler, P.: ELP: Tractable Rules for OWL 2. In: Proc. of the 7th Int. Semantic Web Conference (ISWC 2008). pp. 649–664 (2008)
13. Krötzsch, M., Maier, F., Krisnadhi, A., Hitzler, P.: A better uncle for owl: nominal schemas for integrating rules and ontologies. In: Srinivasan, S., Ramamritham, K., Kumar, A., Ravindra, M.P., Bertino, E., Kumar, R. (eds.) WWW. pp. 645–654. ACM (2011)
14. Levy, A.Y., Rousset, M.C.: Combining Horn Rules and Description Logics in CARIN. Artificial Intelligence 104(1–2), 165–209 (1998)
15. Lutz, C., Areces, C., Horrocks, I., Sattler, U.: Keys, nominals, and concrete domains. J. of Artificial Intelligence Research 23, 667–726 (2004)
16. Marnette, B.: Generalized Schema-Mappings: from Termination to Tractability. In: PODS (2009)
17. de Matos, P., Alcántara, R., Dekker, A., Ennis, M., Hastings, J., Haug, K., Spiteri, I., Turner, S., Steinbeck, C.: Chemical Entities of Biological Interest: an update. Nucleic Acids Research 38(Database-Issue), 249–254 (2010)
18. McCarthy, J.: Circumscription - a form of non-monotonic reasoning. Artif. Intell. (1980)
19. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language: Profiles, W3C Recommendation (October 27 2009)
20. Motik, B., Grau, B.C., Horrocks, I., Sattler, U.: Representing Ontologies Using Description Logics, Description Graphs, and Rules. Artif. Int. 173, 1275–1309 (2009)

21. Motik, B., Sattler, U., Studer, R.: Query Answering for OWL-DL with Rules. J. Web Sem. 3(1), 41–60 (2005)
22. Rector, A.L., Nowlan, W.A., Glowinski, A.: Goals for concept representation in the GALEN project. In: SCAMC '93. pp. 414–418 (1993)
23. Vardi, M.Y.: Why is modal logic so robustly decidable? In: DIMACS Series in Discrete Mathematics and Theoretical Computer Science. pp. 149–184 (1996)

A Appendix

A.1 Program in Example 4 is not (super)-weakly acyclic

We first reproduce the formal definition of weak acyclicity [6].

Definition 10. *A tgd is a rule of the form $\phi(\vec{x}) \rightarrow \exists\vec{y}\psi(\vec{x},\vec{y})$. For a set Σ of tgds, the* dependency graph *is constructed as follows.*

1. *Add a node for each pair (R, i) (called a* position*), where R is a predicate that occurs in Σ and $1 \leq i \leq ar(R)$.*
2. *For each tgd $\phi(\vec{x}) \rightarrow \exists\vec{y}\psi(\vec{x},\vec{y})$ in Σ, each variable $x \in \vec{x}$, and each occurrence of x in ϕ at position (R, i), add the following edges:*
   - *(a) for each occurrence of x in ψ at position (S, j), introduce a* regular *edge $(R, i) \rightarrow (S, j)$, and*
   - *(b) for each variable $y \in \vec{y}$ and each occurrence of y in ψ at position (T, k), introduce a* special *edge $(R, i) \xrightarrow{\star} (T, k)$.*

The set Σ is weakly acyclic *if its dependency graph does not contain a cycle going through a special edge.*

Intuitively, the simple edges of the graph simulate the propagation of existing values and the special edges the creation of fresh values. Let P be the program from Example 4. Note that the function terms in P correspond to existentially quantified variables. As one can see from (a part of) the dependency graph of P shown in Figure 5, program P is not weakly acyclic.

$$\begin{tikzcd}
\langle \mathsf{AceticAcid}, 1\rangle \arrow[r, "\star"] & \langle \mathsf{G_{AA}}, 3\rangle \arrow[r] & \langle \mathsf{Carboxyl}, 1\rangle \arrow[d] \\
\langle \mathsf{G_{AA}}, 1\rangle \arrow[u] & \langle \mathsf{HP}, 1\rangle \arrow[l] & \langle \mathsf{G_{cxl}}, 1\rangle \arrow[l]
\end{tikzcd}$$

Fig. 5. Part of the Dependency Graph of P

In order to examine whether P is super-weakly acyclic, we need to check whether the *trigger* relation $\rightsquigarrow_\Sigma \subseteq \Sigma \times \Sigma$—as defined in [16]—is acyclic. Roughly speaking, for each $r_1, r_2 \in \Sigma$, we have $r_1 \rightsquigarrow_\Sigma r_2$ if and only if an application of r_1 can produce a fact that can satisfy the premise of r_2.

We next show that, for program P from Example 4, we have $s_{\mathsf{G_{AA}}} \rightsquigarrow_P s_{\mathsf{G_{AA}}}$. First, we compute $\mathsf{In}(s_{\mathsf{G_{AA}}}, \mathsf{x})$ and $\mathsf{Out}(s_{\mathsf{G_{AA}}}, \mathsf{f_2(x)})$:

$$\mathsf{In}(s_{\mathsf{G_{AA}}}, \mathsf{x}) = \{(\mathsf{AceticAcid(x)}, 1)\}$$
$$\mathsf{Out}(s_{\mathsf{G_{AA}}}, \mathsf{f_2(x)}) = \{(\mathsf{G_{AA}(x, f_1(x), f_2(x))}, 3)\}$$

Next, we compute the set $\mathsf{Move}(P, \mathsf{Out}(s_{\mathsf{G_{AA}}}, \mathsf{f_2(x)}))$:

$$\mathsf{Move}(P, \mathsf{Out}(s_{\mathsf{G_{AA}}}, \mathsf{f_2(x)})) = \{(\mathsf{G_{AA}(x, f_1(x), f_2(x))}, 3), (\mathsf{HP(x_1, x_3)}, 2),$$

$$\begin{aligned}&(\mathsf{Carboxyl}(\mathsf{x}_3), 1), (\mathsf{G}_{\mathsf{cxl}}(\mathsf{x}, \mathsf{g}_1(\mathsf{x}), \mathsf{g}_2(\mathsf{x})), 1),\\ &(\mathsf{HP}(\mathsf{x}_1, \mathsf{x}_3), 1), (\mathsf{HP}(\mathsf{x}_1, \mathsf{x}_2), 1),\\ &(\mathsf{G}_{\mathsf{AA}}(\mathsf{x}_1, \mathsf{x}_2, \mathsf{x}_3), 1), (\mathsf{Carboxyl}(\mathsf{x}_1), 1),\\ &(\mathsf{AceticAcid}(\mathsf{x}_1), 1)\}\end{aligned}$$

Finally, since $(\mathsf{AceticAcid}(\mathsf{x}), 1)) \sim (\mathsf{AceticAcid}(\mathsf{x}_1), 1))$, the following holds:

$$\mathsf{In}(s_{\mathsf{G}_{\mathsf{AA}}}, \mathsf{x}) \subseteq \mathsf{Move}(P, \mathsf{Out}(s_{\mathsf{G}_{\mathsf{AA}}}, \mathsf{f}_2(\mathsf{x})))$$

By definition of the trigger relation, we then have $s_{\mathsf{G}_{\mathsf{AA}}} \rightsquigarrow_P s_{\mathsf{G}_{\mathsf{AA}}}$.

A.2 Proof of Theorem 7

Theorem 7 follows immediately from the following lemma, since the logic program $\mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O})$ satisfies the lemma's preconditions.

Lemma 2. *Let DG be a set of description graphs and let $\prec$ be a graph ordering on DG. Furthermore, let P be a negation-free logic program in which each rule containing a function symbol corresponds to the start rule of some $G \in DG$, and that satisfies the following:*

(a) *for each $G_1, G_2 \in DG$ such that $G_1 \neq G_2$, $G_1 \prec G_2$, and $\{s_{G_1}, s_{G_2}\} \subseteq P$, we have $\mathsf{ChkPair}(G_1, G_2) \subseteq P$;*
(b) *for each $G \in DG$ with start rule $s_G \in P$, we have $\mathsf{ChkSelf}(G) \subseteq P$.*

Then, integer $i \geq 1$ exists such that $\mathsf{Cycle} \in T_P^i$ or $T_P^{i+1} = T_P^i$.

Proof. We say that a term t *occurs* in a set of facts if t is one of the arguments of some fact contained in the set.

In order to prove this lemma, we first show the following claim.

Claim 1 *For each $i \geq 0$ and each ground term $f(t)$ that occurs in T_P^i, program P contains a rule of the form $A(x) \to G(x, \ldots, f(x), \ldots) \in P$ such that* (**i**) *and* (**ii**) *hold:*

(i) $G(t, \ldots, f(t), \ldots) \in T_P^i$
(ii) $A(t) \in T_P^i$

Proof. We prove the claim by induction on i.

Induction base: $T_P^0 = \emptyset$, so the claim holds vacuously.

Induction step: We assume that Claim 1 holds for i and we prove that it holds for $i+1$. By the induction hypothesis, for each $f(t)$ that occurs in T_P^i such that t is a ground term, there exists a rule $A(x) \to G(x, \ldots, f(x), \ldots) \in P$ such that **(i)** and **(ii)** hold. We assume that $f(t)$ occurs in T_P^{i+1}. There are two cases: either $f(t)$ occurs in T_P^i or it does not. If $f(t)$ occurs in T_P^i, then there exists a rule $A(x) \to G(x, \ldots, f(x), \ldots) \in P$ such that $G(t, \ldots, f(t), \ldots) \in T_P^i$ and $A(t) \in T_P^i$ by induction hypothesis. If $f(t)$ does not occur in T_P^i, then $f(t)$ has been introduced by the $(i+1)$-th application of T_P. Since the function symbol f appears in P only in the rule $A(x) \to G(x, \ldots, f(x), \ldots)$, the only way to derive $f(t)$ is via the aforementioned rule. Therefore, the precondition of the rule must be satisfied and, so, we have $A(t) \in T_P^i$, which implies $A(t) \in T_P^{i+1}$. □

We now prove the main claim of this theorem by showing that, if no $i \geq 1$ exists such that $T_P^{i+1} = T_P^i$, then i exists such that $\mathsf{Cycle} \in T_P^i$. Since $T_P^i \subset T_P^{i+1}$ for each $i \geq 1$, there are infinitely many different atoms in T_P^∞. Furthermore, the number of predicates that occur in P is finite and every predicate that appears in P has a finite arity so, since T_P^∞ contains infinitely many different atoms, there are infinitely many different terms occurring in T_P^∞. By the definition of T_P, the range of θ is $HU(P)$, so no new constants are introduced; but then, since there are finitely many function in P but there are infinitely many different terms occurring in T_P^∞, at least one of the function symbols is used twice resulting in a term of the form $f_1(f_k(\ldots(f_1(t))\ldots))$ occurring in T_P^∞, where t is a ground term. For $1 \leq j \leq k+1$, we use the following abbreviations:

$$t_j = f_j(\ldots(f_1(t))\ldots), \text{ for } 1 \leq j \leq k$$
$$t_j = f_1(f_k \ldots (f_1(t))\ldots), \text{ for } j = k+1$$

Since t_{k+1} occurs in T_P^∞, there exists an atom $Q(\ldots, t_{k+1}, \ldots) \in T_P^\infty$. The set T_P^∞ is defined as an (infinite) union of sets, so $Q(\ldots, t_{k+1}, \ldots)$ is contained in one of these sets, and there exists $i \geq 1$ such that t_{k+1} occurs in T_P^i. By Claim 1, a rule $A_1(x) \rightarrow G_1(x, \ldots, f(x), \ldots)$ exists such that $G_1(t_k, \ldots, t_{k+1}, \ldots) \in T_P^i$ and $A_1(t_k) \in T_P^i$; but then, t_k occurs in T_P^i. Next, we show the following claim:

Claim 2 *For each j with $k \geq j > 1$, if t_j occurs in T_P^i, then P contains a rule $A_j(x) \rightarrow G_j(x, \ldots, f_j(x), \ldots)$ such that*

(i) *$G_j(t_{j-1}, \ldots, t_j, \ldots) \in T_P^i$, and*
(ii) *$A_j(t_{j-1}) \in T_P^i$.*

Proof. We prove the claim by induction on j.

Induction base: For $j = k$, we prove that there exists in P a rule of the form $A_k(x) \rightarrow G_k(x, \ldots, f_k(x), \ldots)$ such that $G_k(t_{k-1}, \ldots, t_k, \ldots) \in T_P^i$ and $A_k(t_{k-1}) \in T_P^i$. Since t_k occurs in T_P^i, by Claim 1 the statement holds.

Induction step: We assume that Claim 2 holds for j and we prove that it holds for $j-1$. By induction hypothesis, we have $A_j(x) \rightarrow G_j(x, \ldots, f_j(x), \ldots) \in P$, such that $G_j(t_{j-1}, \ldots, t_j, \ldots) \in T_P^i$ and $A_j(t_{j-1}) \in T_P^i$. Thus, t_{j-1} occurs in T_P^i, so by Claim 1 we have $A_{j-1}(x) \rightarrow G_{j-1}(x, \ldots, f_{j-1}(x), \ldots) \in P$ and, as a consequence, $G_{j-1}(t_{j-2}, \ldots, t_{j-1}, \ldots) \in T_P^i$ and $A_{j-1}(t_{j-2}) \in T_P^i$. □

Since $\prec$ is a transitive and irreflexive relation, $G_1 \prec \ldots \prec G_k \prec G_1$ does not hold, so an integer ℓ with $1 \leq \ell \leq k$ exists such that $G_\ell \not\prec G_{(\ell \bmod k)+1}$. Due to the latter, the following rule is contained in $\mathsf{Check}(DG, \prec)$ and is thus also in P:

$$G_\ell(x, \ldots, y, \ldots) \wedge A_{(\ell \bmod k)+1}(y) \rightarrow \mathsf{Cycle} \qquad (7)$$

We distinguish the following possibilities for ℓ.

$\underline{\ell = k}$: By Claim 2(**i**) for $j = k$, by $A_1(t_k) \in T_P^i$, and by (7), we have that $\mathsf{Cycle} \in T_P^{i+1}$.

$\underline{1 \leq \ell < k}$: By Claim 2(**i**) for $j = \ell$, by Claim 2(**ii**) for $j = \ell + 1$, and by (7), we have that $\mathsf{Cycle} \in T_P^{i+1}$. □

A.3 Proof of Lemma 1

Lemma 1. Let $\mathcal{O} = \langle DG, \prec, R, F \rangle$ be a DGLP ontology. If σ is a stratification of $\mathsf{LP}(\mathcal{O})$, then σ can be extended to a DG-stratification σ' of $\mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O})$.

Proof. Let $P = \mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O})$. We define the extension σ' of σ as follows.

1. If $r \in \mathsf{LP}(\mathcal{O})$, then $\sigma'(r) = \sigma(r)$.
2. If $r \in \mathsf{Check}(\mathcal{O})$ and r is of the form $G(\vec{y}) \wedge A(y_i) \rightarrow \mathsf{Cycle}$ (where $1 \leq i \leq ar(G)$ and $\vec{y}$ is $y_1, \ldots, y_{ar(G)}$), then

$$\sigma'(r) = \max\{\sigma(r') \mid r' \in \mathsf{LP}(\mathcal{O}),\ \text{either } A \in head_P(r') \text{ or } G \in head_P(r')\}.$$

Since the cycle detection rules have Cycle in the head, and Cycle does not occur in the body of a rule in $\mathsf{LP}(\mathcal{O})$, mapping σ' is a stratification of P. We show that σ' is also a DG-stratification of P. We prove condition $(\mathbf{i})$ of Definition 9; condition $(\mathbf{ii})$ can be shown in a similar way. Let $P_1, \ldots, P_n$ be a stratification partition of P w.r.t. σ'. Consider an arbitrary i with $1 \leq i \leq n$ and arbitrary $G_1, G_2 \in DG$ such that $G_1 \neq G_2$, $G_1 \not\prec G_2$ and $\{s_{G_1}, s_{G_2}\} \subseteq P_i$. Each rule $r \in \mathsf{ChkPair}(G_1, G_2)$ is of the form $G_1(x_1, \ldots, x_{|V_1|}) \wedge A_2(x_j) \rightarrow \mathsf{Cycle}$, where $1 \leq j \leq |V_1|$. We prove that $\sigma(r) = i$. The only rule r' such that $G_1 \in head(r')$ is the start or the recognition rule of G_1—that is, s_{G_1} or r_{G_1}. Now $s_{G_1} \in P_i$ implies that $\sigma(s_{G_1}) = i$, and $l_{G_1} \in P_i$ implies that $\sigma(r_{G_1}) = i$. Thus, we only need to prove that $\max\{\sigma(r') \mid A_2 \in head_P(r')\} \leq i$. Now $\sigma(r') > i = \sigma(r)$ cannot hold because this would imply that $A_2 \in head_P(r')$, $A_2 \in body^+(r)$ and $\sigma(r') \not\leq \sigma(r)$, so the property holds. □

A.4 Proof of Theorem 10

Theorem 10. Let $\mathcal{O}$ be a DGLP ontology and $P = \mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O})$. If $P_1, \ldots, P_n$ is a stratification partition of P w.r.t. a DG-stratification of P, then, for each j with $1 \leq j \leq n$, there exists $i \geq 1$ such that $\mathsf{Cycle} \in U^i_{P_j}$, or $U^{i+1}_{P_j} = U^i_{P_j}$ and $U^i_{P_j}$ is finite.

Proof. Properties (8) and (9) hold by the definition of $U^i_{P_j}$ and $U^\infty_{P_j}$.

$$U^i_{P_j} \subseteq U^{i+1}_{P_j} \qquad \text{for each } i \geq 1 \text{ and each } 1 \leq j \leq n \tag{8}$$

$$U^\infty_{P_j} \subseteq U^\infty_{P_{j+1}} \qquad \text{for each } 1 \leq j < n \tag{9}$$

Evaluating a stratifiable logic program can be reduced to evaluating a negation-free program for each stratum. The claim of this theorem then holds by a straightforward induction on the strata.

Induction base: By Definition 9, P_1 contains the necessary cycle detection rules, so the conditions $(\mathbf{a})$ and $(\mathbf{b})$ of Lemma 2 are satisfied. Hence, $i \geq 1$ exists such that $\mathsf{Cycle} \in U^i_{P_1}$ or $U^{i+1}_{P_1} = U^i_{P_1}$ and $U^i_{P_1}$ is finite.

Induction step: By the induction hypothesis, there exists integer $i \geq 1$ such that $\mathsf{Cycle} \in U^i_{P_j}$ or $U^{i+1}_{P_j} = U^i_{P_j}$ and $U^i_{P_j}$ is finite. If $\mathsf{Cycle} \in U^i_{P_j}$, then by (9) the claim trivially holds. Otherwise, by Definition 9 and the fact that $U^i_{P_j}$ is finite, the conditions of Lemma 2 are satisfied, so $i \geq 1$ exists such that $\mathsf{Cycle} \in U^i_{P_{j+1}}$ or $U^{i+1}_{P_{j+1}} = U^i_{P_{j+1}}$ and $U^i_{P_{j+1}}$ is finite. □

A.5 The representation of chemical classes by XSB rules

The following are the XSB rules used for modelling chemical classes in the experiments described in Section 8.

```
hasBenzeneRing(X) :- molecule(X), hasAtom(X,Y1), c(Y1),
                     hasAtom(X,Y2), c(Y2), single(Y1,Y2),
                     single(Y2,Y1), hasAtom(X,Y3), c(Y3),
                     double(Y2,Y3), double(Y3,Y2),
                     hasAtom(X,Y4), c(Y4), single(Y3,Y4),
                     single(Y4,Y3), hasAtom(X,Y5),
                     double(Y4,Y5), double(Y5,Y4), c(Y5),
                     hasAtom(X,Y6), c(Y6), single(Y5,Y6),
                     single(Y6,Y5), double(Y6,Y1),
                     double(Y1,Y6).

hasFourMemberedRing(X) :- molecule(X), hasAtom(X,Y1),
                          hasAtom(X,Y2), (Y1) \== (Y2),
                          bond(Y1,Y2), bond(Y2,Y1),
                          hasAtom(X,Y3), (Y1) \== (Y3),
                          (Y2) \== (Y3), bond(Y2,Y3),
                          bond(Y3,Y2), hasAtom(X,Y4),
                          (Y1) \== (Y4), (Y2) \== (Y4),
                          (Y3) \== (Y4),
                          bond(Y3,Y4), bond(Y4,Y3),
                          bond(Y4,Y1), bond(Y1,Y4).

notHydroCarbon(X) :- hasAtom(X,Y),
                     tnot(c(Y)), tnot(h(Y)).
hydroCarbon(X) :- molecule(X), tnot(notHydroCarbon(X)).

molWithCarbon(X) :- hasAtom(X, Y), c(Y).
inorganic(X) :- molecule(X), tnot(molWithCarbon(X)).

atLeast2Carbons(X) :- molecule(X), hasAtom(X,Y1), c(Y1),
                      hasAtom(X,Y2), c(Y2), Y1 \== Y2.
atLeast3Carbons(X) :- molecule(X), hasAtom(X,Y1), c(Y1),
                      hasAtom(X,Y2), c(Y2), Y1 \== Y2,
                      hasAtom(X,Y3), c(Y3),
                      Y1 \== Y3, Y2 \== Y3.
exactly2Carbon(X) :- atLeast2Carbons(X),
                     tnot(atLeast3Carbons(X)).
```