# 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

## 1 Introduction

OWL 2[1] is commonly used to represent objects with complex structure, such as complex assemblies in engineering applications [5], human anatomy [13], or the structure of chemical molecules [7]. 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[2] ontology—a public dictionary of *molecular entities* used to enhance interoperability of applications supporting tasks such as drug discovery. In order to automate the classification of molecular entities, ChEBI descriptions were translated into OWL and then classified using automated reasoning. However, this approach was hindered by the fundamental inability of OWL to precisely represent the structure of complex molecular entities, as the tree-model property of OWL prevents one from describing non-tree-like relationships using schema axioms. For example, OWL 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 fully described in OWL. 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;[3] however, this either results in undecidability [9] or requires restrictions in the shape of the rules [8], which typically prevent the rules from axiomatising the required structures. An alternative approach suggests a combination of OWL 2, rules, and *description graphs* (DGs) [12]—a graphical notation for describing non-tree-like structures. Decidability of reasoning is ensured 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 [7], DGs solved

* Work supported by EU FP7 SEALS and EPSRC projects ConDOR, ExODA, and LogMap.

[1] http://www.w3.org/TR/owl2-overview/

[2] http://www.ebi.ac.uk/chebi/

[3] http://www.w3.org/Submission/SWRL/

only some of the problems related to the representation of structured objects, and our subsequent discussions with EBI researchers revealed the following drawbacks.

First, the DG approach 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 [12] 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.[4] In order to overcome the first problem, we give semantics to our formalism via a translation into logic programs 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 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 precedence relation describing allowed implications between DGs is specified by the modeller; 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 negation-free ontologies and ontologies with stratified negation. 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. In Section 7 we briefly discuss a preliminary evaluation of our formalism which indicates that reasoning with DGLP ontologies is practically feasible. Proofs of the technical results can be found online.[5]

## 2 Preliminaries

We assume the reader to be familiar with OWL and description logics; for brevity, we write OWL axioms using the DL notation. Let $\Sigma = (\Sigma_C, \Sigma_F, \Sigma_P)$ be a *first-order logic signature*, where $\Sigma_C$, $\Sigma_F$, and $\Sigma_P$ are countably infinite sets of constant, function, and

[4] http://www.w3.org/TR/owl-profiles/

[5] http://www.cs.ox.ac.uk/isg/people/despoina.magka/pubs/reports/DGLPTechnicalReport.pdf

predicate symbols, respectively, and where $\Sigma_P$ 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

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

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 $head(r) \cup body^-(r)$ 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 $P$, $HU(P)$ (Herbrand Universe) is the set of all terms that can be formed using constants and functions 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 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 $body^+(r) \subseteq I$ and $body^-(r) \cap I = \emptyset$ imply $head(r) \subseteq I$. 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 $r \in ground(P)$. 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 $ground(P)$ by removing each rule $r$ such that $body^-(r) \cap I \neq \emptyset$, 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 $\theta$ 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:

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

Let $T_P^0 = \emptyset$, let $T_P^i = T_P(T_P^{i-1})$ for $i \geq 1$, and let $T_P^\infty = \bigcup_{i=1}^\infty T_P^i$. Such $P$ has at most one stable model, and $T_P^\infty$ 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. The inability of OWL to describe cyclic structures with sufficient precision causes problems when modelling chemical compounds, as molecules are highly cyclic. For example, the cyclobutane molecule contains four carbon atoms connected in a ring, 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 adding axioms due to the *tree-model property* of OWL, according to which 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 [12] 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 complex object by means of a directed labeled graph. To avoid misunderstandings, we refer to the formalism from [12] 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 2(a). Nevertheless, the interpretation $I''$ shown in Figure 1(d) also 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.

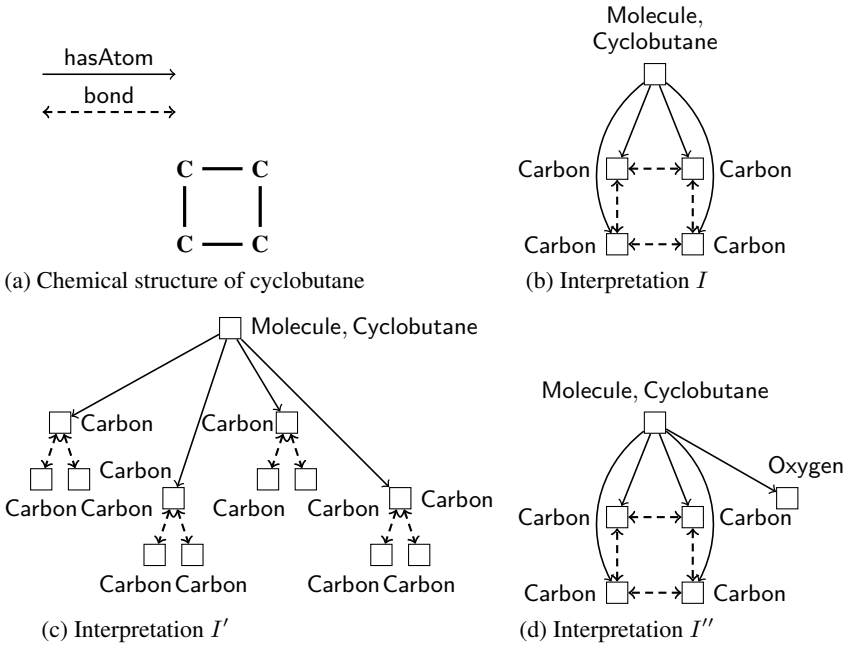
One might describe the class of hydrocarbon molecules (i.e., molecules consisting exclusively of hydrogens and carbons) using axiom (2). One would expect the definition of cyclobutane (as given in a DGDL ontology) and (2) to imply subsumption (3).

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

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

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 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.

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 [6]. We give semantics to our formalism by translating DGLP ontologies into logic programs with function symbols. As is common in logic programming, the




(a) Chemical structure of cyclobutane

(b) Interpretation $I$

(c) Interpretation $I'$

(d) Interpretation $I''$

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

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.

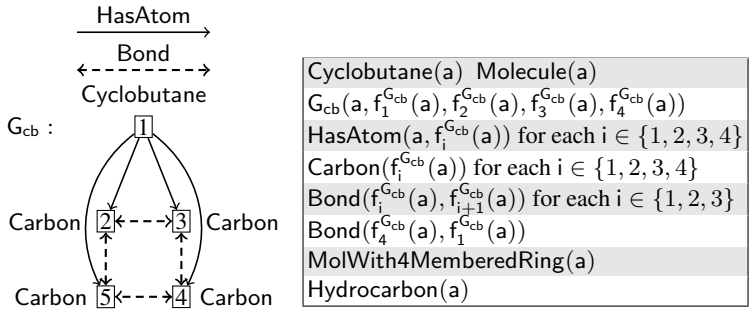
Logic programs with function symbols can axiomatise infinite non-tree-like structures, so reasoning with DGLP ontologies is trivially undecidable [2]. Our goal, however, is not to model arbitrarily large structures, but to describe complex objects up to a certain level of granularity. For example, acetic 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 new acyclicity condition that ensures decidability and allows for the modelling of naturally-arising molecular structures, such as acetic acid, that would be ruled out by existing syntactic acyclicity conditions [4, 11].

## 4 Description Graph Logic Programs

We now present the DGLP formalism in detail; we first define description graphs.

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

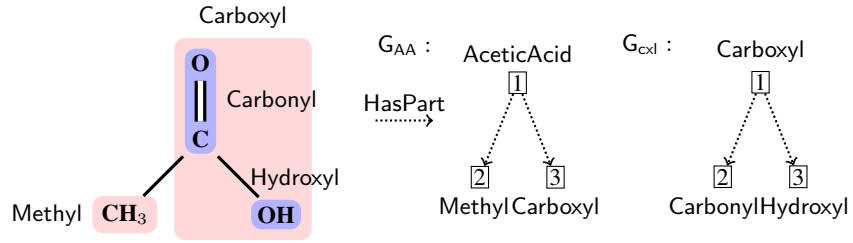
- $V = \{1, \ldots, n\}$ *is a nonempty set of vertices,*
- $E \subseteq V \times V$ *is a set of edges,*
- $\lambda$ *assigns a set of unary predicates* $\lambda(v) \subseteq \Sigma_P$ *to each vertex* $v \in V$ *and a set of binary predicates* $\lambda(v_1, v_2) \subseteq \Sigma_P$ *to each edge* $(v_1, v_2) \in E$*,*




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

(a) Cyclobutane DG (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 $G_{AA}$ and the $G_{cxl}$ DGs

- $A \in \Sigma_P$ *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 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 $G_{AA}$ be a graph that represents acetic acid, and

let $\mathsf{G_{cxl}}$ be a graph that represents the carboxyl group (Fig. 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 [6] and included in $R$, and datatypes can be handled as in [10]. Similarly, we could think of $F$ as an OWL 2 ABox, as ABox assertions correspond directly to facts [6]. 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 $\ell_G$*, and the* recognition rule $r_G$ of $G$ *are defined as follows ($G$ is also used as a predicate of arity $|V|$):*

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

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

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

The start and layout rules of a description graph serve to unfold the graph's structure, whereas the recognition rule identifies instances of the start predicate. 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 start rule and layout rule that correspond to the DG of cyclobutane from Figure 2 (assuming mode $\Rightarrow$ as specified in Definition 5) are the following.

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

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

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* $\ell_G$ *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 $\mathsf{Axioms}(DG) \cup R \cup F$. We check whether $D$ subsumes $C$ 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 constant 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 and rules $(r_2)$ and $(r_3)$ represent the class of hydrocarbons.

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

$$\mathsf{Molecule}(\mathsf{x}) \wedge \mathsf{HasAtom}(\mathsf{x}, \mathsf{y}) \wedge \mathbf{not}\mathsf{Carbon}(\mathsf{y}) \wedge \mathbf{not}\mathsf{Hydrogen}(\mathsf{y}) \rightarrow \mathsf{NHC}(\mathsf{x}) \quad (r_2)$$

$$\mathsf{Molecule}(\mathsf{x}) \wedge \mathbf{not}\ \mathsf{NHC}(\mathsf{x}) \rightarrow \mathsf{HydroCarbon}(\mathsf{x}) \quad (r_3)$$

$$\mathsf{Cyclobutane}(\mathsf{x}) \rightarrow \mathsf{Molecule}(\mathsf{x}) \quad (r_4)$$

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. We also 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 $F = \{\mathsf{Cyclobutane}(\mathsf{a})\}$, 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 $\mathsf{LP}(\mathcal{O}) \models \mathsf{Hydrocarbon}(\mathsf{a})$ and $\mathsf{LP}(\mathcal{O}) \models \mathsf{MolWith4MemberedRing}(\mathsf{a})$, as expected.

## 5 Semantic Acyclicity

Reasoning about logic programs with function symbols is undecidable in general [2]. This problem is similar to reasoning about datalog programs with existentially quantified rule heads (known as tuple-generating dependencies or tgds) [3]. For such programs, conditions such as weak acyclicity [4] or super-weak acyclicity [11] ensure the termination of bottom-up reasoning algorithms: these conditions examine the syntactic structure of the 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. Such conditions can also be applied to DGLP ontologies; however, they may overestimate the propagation of values introduced by existential quantification and thus rule out unproblematical programs that generate only finite structures. As shown in Example 4, this is 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. In particular, 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 $G_1 \not\leftarrow G_2$.

**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:*

$$\mathsf{ChkPair}(G_1, G_2) = \{G_1(x_1, \ldots, x_{|V_1|}) \wedge A_2(x_k) \rightarrow \mathsf{Cycle} \mid 1 \leq k \leq |V_1|\} \quad (4)$$

$$\mathsf{ChkSelf}(G_i) = \{G_i(x_1, \ldots, x_{|V_i|}) \wedge A_i(x_k) \rightarrow \mathsf{Cycle} \mid 1 < k \leq |V_i|\} \quad (5)$$

*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:

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

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

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

**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 $DG = \{\mathsf{G_{AA}}, \mathsf{G_{cxl}}\}$ with $m_{\mathsf{AA}} = m_{\mathsf{cxl}} = \ \Leftrightarrow$, let $\prec = \{(\mathsf{G_{AA}}, \mathsf{G_{cxl}})\}$, let $F = \{\mathsf{AceticAcid(a)}\}$, and let $\mathcal{O} = \langle DG, \prec, \emptyset, F \rangle$. The logic program $\mathsf{LP}(\mathcal{O})$ contains $F$ and the following rules (HP abbreviates HasPart):

$$\mathsf{AceticAcid}(x) \rightarrow \mathsf{G_{AA}}(x, f_1(x), f_2(x))$$

$$\mathsf{G_{AA}}(x, y, z) \rightarrow \mathsf{AceticAcid}(x) \wedge \mathsf{Methyl}(y) \wedge \mathsf{Carboxyl}(z) \wedge \mathsf{HP}(x, y) \wedge \mathsf{HP}(x, z)$$

$$\mathsf{Methyl}(y) \wedge \mathsf{Carboxyl}(z) \wedge \mathsf{HP}(x, y) \wedge \mathsf{HP}(x, z) \rightarrow \mathsf{G_{AA}}(x, y, z)$$

$$\mathsf{Carboxyl}(x) \rightarrow \mathsf{G_{cxl}}(x, g_1(x), g_2(x))$$

$$\mathsf{G_{cxl}}(x, y, z) \rightarrow \mathsf{Carboxyl}(x) \wedge \mathsf{Carbonyl}(y) \wedge \mathsf{Hydroxyl}(z) \wedge \mathsf{HP}(x, y) \wedge \mathsf{HP}(x, z)$$

$$\mathsf{Carbonyl}(y) \wedge \mathsf{Hydroxyl}(z) \wedge \mathsf{HP}(x, y) \wedge \mathsf{HP}(x, z) \rightarrow \mathsf{G_{cxl}}(x, y, z)$$

Let also $\mathsf{Check}(\mathcal{O}) = \mathsf{Check}(DG, \prec)$ as defined in Example 3. We can easily compute the stable model of $P = \mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O})$ using the $T_P$ operator and check that Cycle is not in the (only) stable model of $P$; so $P \not\models \mathsf{Cycle}$ and $\mathcal{O}$ is semantically acyclic. However, $P$ is neither weakly [4] nor super-weakly acyclic [11]. This, we believe, justifies the importance of semantic acyclicity for our applications.

## 6 Reasoning with DGLP Ontologies

Initially, we consider the problem of reasoning with a negation-free DGLP ontology $\mathcal{O} = \langle DG, \prec, R, F \rangle$. Intuitively, one can apply the $T_P$ operator to $P = \mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O})$ and compute $T_P^1, \ldots, T_P^i$ and so on. By Theorem 5, 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 5.** *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 $i \geq 1$.*

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; note that the inverse does not hold as semantic acyclicity is a sufficient but not necessary termination condition.

Next, we consider the case of DGLP ontologies with stratified negation-as-failure. We start by recapitulating several definitions. For each program $P$, a *stratification* of $P$ is a mapping $\sigma : P \to \mathbb{N}$ such that for each rule $r \in P$ we have (i) if $B \in body_P^+(r)$, then $\sigma(r') \leq \sigma(r)$ for each $r' \in P$ with $B \in head_P(r')$ and (ii) 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. $\sigma$ 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 $U_{P_n}^\infty$. 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 $\sigma$ of $P$. Then, $\sigma$ 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 $\mathsf{ChkSelf}(G) \subseteq P_i$.*

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 $\mathsf{LP}(\mathcal{O}) \cup \mathsf{Check}(\mathcal{O})$.

**Lemma 1.** *Let $\mathcal{O} = \langle DG, \prec, R, F \rangle$ be a DGLP ontology. If $\sigma$ is a stratification of $\mathsf{LP}(\mathcal{O})$, then $\sigma$ can be extended to a DG-stratification $\sigma'$ 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; please note that we show the decidability of semantic acyclicity only for ontologies with stratified negation.

**Theorem 6.** *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 $\mathsf{Cycle} \in U_{P_j}^i$, or $U_{P_j}^{i+1} = U_{P_j}^i$ and $U_{P_j}^i$ is finite.*

## 7 Implementation Results and Discussion

In order to test the applicability of our approach in practice, we developed a prototypical implementation based on the XSB system.[6] Using data extracted from ChEBI, we built a number of DGLP ontologies with stratified NAF and we checked each ontology for acyclicity and for entailed subsumptions: all ontologies were found acyclic and all molecules were classified as expected; additionally, testing subsumptions for an ontology representing 70 molecules did not require more than a few minutes on a standard desktop computer. Given the prototypical nature of our application, we consider the results as evidence of the practical feasibility of our approach.

In this paper we have laid the theoretical foundations of a novel, expressive, and OWL 2 RL-compatible ontology language that is well suited to modelling objects with complex structure. In the future, we plan to modify our approach in order to avoid the explicit definition of graph ordering by the modeller; furthermore, we shall investigate whether semantic acyclicity can be combined with other conditions (such as those defined in [1]) in order to obtain a more general acyclicity check. Finally, we will optimise our prototype in order to obtain a fully-scalable chemical classification system.

## References


1. 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)
2. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: Proc. ICALP. pp. 73–85 (1981)
3. 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)
4. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. In: ICDT. pp. 207–224 (2003)
5. Graves, H.: Representing product designs using a description graph extension to OWL 2. In: Proc. of the 5th OWLED Workshop (2009)
6. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: WWW (2003)
7. 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)
8. 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)
9. Levy, A.Y., Rousset, M.C.: Combining horn rules and description logics in CARIN. Artificial Intelligence 104(1–2), 165–209 (1998)
10. Lutz, C., Areces, C., Horrocks, I., Sattler, U.: Keys, nominals, and concrete domains. J. of Artificial Intelligence Research 23, 667–726 (2004)
11. Marnette, B.: Generalized schema-mappings: from termination to tractability. In: PODS (2009)
12. Motik, B., Grau, B.C., Horrocks, I., Sattler, U.: Representing ontologies using description logics, description graphs, and rsules. Artif. Int. 173, 1275–1309 (2009)
13. Rector, A.L., Nowlan, W.A., Glowinski, A.: Goals for concept representation in the GALEN project. In: SCAMC '93. pp. 414–418 (1993)


[6] http://xsb.sourceforge.net/