Modelling Structured Domains With Logic

Despoina Magka, Boris Motik and Ian Horrocks

Department of Computer Science, University of Oxford

February 9, 2012
Outline

1 Motivation

2 Introducing DGLPs

3 Prototype

4 Conclusion
WHAT IS OWL?

Family of logic-based knowledge representation languages

Web Ontology Language: a W3C standard, widely used in ontology-based applications, e.g. formal biomedical vocabularies

Formal foundations of OWL provided by Description Logics

What are Description Logics?
WHAT IS OWL?

- Family of logic-based knowledge representation languages

Web Ontology Language: a W3C standard, widely used in ontology-based applications, e.g. formal biomedical vocabularies

Formal foundations of OWL provided by Description Logics

What are Description Logics?
WHAT IS OWL?

- Family of **logic-based** knowledge representation languages
- **Web Ontology Language**: a W3C standard, widely used in ontology-based applications, e.g. formal biomedical vocabularies
What is OWL?

- Family of logic-based knowledge representation languages
- Web Ontology Language: a W3C standard, widely used in ontology-based applications, e.g. formal biomedical vocabularies
What is OWL?

- Family of logic-based knowledge representation languages
- Web Ontology Language: a W3C standard, widely used in ontology-based applications, e.g. formal biomedical vocabularies
- Formal foundations of OWL provided by Description Logics
What is OWL?

- Family of logic-based knowledge representation languages
- Web Ontology Language: a W3C standard, widely used in ontology-based applications, e.g. formal biomedical vocabularies

Formal foundations of OWL provided by Description Logics

What are Description Logics?
What is OWL?

- Family of logic-based knowledge representation languages
- Web Ontology Language: a W3C standard, widely used in ontology-based applications, e.g. formal biomedical vocabularies
- Formal foundations of OWL provided by Description Logics
- What are Description Logics?
  → Decidable fragments of first-order logic with well-understood computational properties
Modelling structured domains with OWL

- OWL used for the representation of complex structures:
Modelling structured domains with OWL

- OWL used for the representation of complex structures:
  - Aerospace
Modelling structured domains with OWL

- OWL used for the representation of complex structures:
  - Aerospace
  - Cellular biology
Modelling structured domains with OWL

- OWL used for the representation of complex structures:
  - Aerospace
  - Cellular biology
  - Human anatomy
Modelling structured domains with OWL

- OWL used for the representation of **complex** structures:
  - Aerospace
  - Cellular biology
  - Human anatomy
  - Molecules
THE ChEBI Ontology

- OWL ontology Chemical Entities of Biological Interest
The ChEBI Ontology

- OWL ontology Chemical Entities of Biological Interest
  - Freely accessible dictionary of molecular entities
THE CHEBI ONTOLOGY

- OWL ontology Chemical Entities of Biological Interest
  - Freely accessible dictionary of molecular entities
  - High quality annotation and taxonomy of chemical compounds

Drug discovery and elucidation of metabolic pathways
THE CHEBI ONTOLOGY

- OWL ontology Chemical Entities of Biological Interest
  - Freely accessible dictionary of molecular entities
  - High quality annotation and taxonomy of chemical compounds
  - Interoperability between researchers
THE ChEBI Ontology

- OWL ontology Chemical Entities of Biological Interest
  - Freely accessible dictionary of molecular entities
  - High quality annotation and taxonomy of chemical compounds
  - Interoperability between researchers
  - Drug discovery and elucidation of metabolic pathways
AUTOMATE CHEMICAL CLASSIFICATION

- ChEBI is manually incremented
**AUTOMATE CHEMICAL CLASSIFICATION**

- ChEBI is *manually* incremented

- Currently contains approx. **27,000** fully annotated entries
AUTOMATE CHEMICAL CLASSIFICATION

- ChEBI is **manually** incremented
- Currently contains approx. **27,000** fully annotated entries
- Grows at a rate of **1,500 entries per curator per year**
AUTOMATE CHEMICAL CLASSIFICATION

- ChEBI is manually incremented
- Currently contains approx. 27,000 fully annotated entries
- Grows at a rate of 1,500 entries per curator per year
- Biologically interesting entities possibly > 1,000,000
AUTOMATE CHEMICAL CLASSIFICATION

- ChEBI is manually incremented
- Currently contains approx. 27,000 fully annotated entries
- Grows at a rate of 1,500 entries per curator per year
- Biologically interesting entities possibly > 1,000,000
- Each new molecule is subsumed by several chemical classes
AUTOMATE CHEMICAL CLASSIFICATION

- ChEBI is **manually** incremented
- Currently contains approx. **27,000** fully annotated entries
- Grows at a rate of **1,500 entries per curator per year**
- Biologically interesting entities possibly possibly **> 1,000,000**
- Each new **molecule** is subsumed by several chemical **classes**
  - Is dinitrogen **inorganic**?
AUTOMATE CHEMICAL CLASSIFICATION

- ChEBI is *manually* incremented

- Currently contains approx. 27,000 fully annotated entries

- Grows at a rate of 1,500 entries per curator per year

- Biologically interesting entities possibly > 1,000,000

- Each new *molecule* is subsumed by several chemical classes
  - Is dinitrogen *inorganic*?
  - Does cyclobutane contain a *four-membered ring*?
AUTOMATE CHEMICAL CLASSIFICATION

- ChEBI is manually incremented
- Currently contains approx. 27,000 fully annotated entries
- Grows at a rate of 1,500 entries per curator per year
- Biologically interesting entities possibly > 1,000,000
- Each new molecule is subsumed by several chemical classes
  - Is dinitrogen inorganic?
  - Does cyclobutane contain a four-membered ring?
  - Is acetylene a hydrocarbon?

Speed up curating tasks with automated reasoning tools
AUTOMATE CHEMICAL CLASSIFICATION

- ChEBI is manually incremented
- Currently contains approx. 27,000 fully annotated entries
- Grows at a rate of 1,500 entries per curator per year
- Biologically interesting entities possibly > 1,000,000
- Each new molecule is subsumed by several chemical classes
  - Is dinitrogen inorganic?
  - Does cyclobutane contain a four-membered ring?
  - Is acetylene a hydrocarbon?
  - Does benzaldehyde contain a benzene ring?

Speed up curating tasks with automated reasoning tools
AUTOMATE CHEMICAL CLASSIFICATION

- ChEBI is manually incremented
- Currently contains approx. 27,000 fully annotated entries
- Grows at a rate of 1,500 entries per curator per year
- Biologically interesting entities possibly > 1,000,000
- Each new molecule is subsumed by several chemical classes
  - Is dinitrogen inorganic?
  - Does cyclobutane contain a four-membered ring?
  - Is acetylene a hydrocarbon?
  - Does benzaldehyde contain a benzene ring?
- Speed up curating tasks with automated reasoning tools
AUTOMATE CHEMICAL CLASSIFICATION

- ChEBI is manually incremented

- Currently contains approx. 27,000 fully annotated entries

- Grows at a rate of 1,500 entries per curator per year

- Biologically interesting entities possibly > 1,000,000

- Each new molecule is subsumed by several chemical classes
  - Is dinitrogen inorganic? ▶️ Yes
  - Does cyclobutane contain a four-membered ring? ▶️ Yes
  - Is acetylene a hydrocarbon? ▶️ Yes
  - Does benzaldehyde contain a benzene ring? ▶️ Yes

- Speed up curating tasks with automated reasoning tools
(Mis)representing rings with OWL

- Chemical compounds with rings are highly frequent
(Mis)representing rings with OWL

- Chemical compounds with rings are highly frequent
- Fundamental inability of OWL to represent cycles
(Mis)representing rings with OWL

- Chemical compounds with rings are highly frequent
- Fundamental inability of OWL to represent cycles
- OWL syntax does not offer access to variables
(Mis)representing rings with OWL

- Chemical compounds with rings are highly frequent
- Fundamental inability of OWL to represent cycles
- OWL syntax does not offer access to variables

**Example**

```
Uncle ⊑ Male ⊓ ∃hasSibling.∃hasChild.(Human)
```
(Mis)representing rings with OWL

- Chemical compounds with rings are highly frequent
- Fundamental inability of OWL to represent cycles
- OWL has the tree-model property: each consistent OWL ontology has at least one tree-shaped model
(Mis)representing rings with OWL

- Chemical compounds with rings are highly frequent
- Fundamental inability of OWL to represent cycles
- OWL has the tree-model property: each consistent OWL ontology has at least one tree-shaped model

**Example**

Cyclobutane \( \subseteq \exists^{(=4)} \text{hasAtom} . (\text{Carbon} \sqcap \exists^{(=2)} \text{hasBond}.\text{Carbon}) \)

```
C --- C
|    |
C --- C
```
(Mis)representing rings with OWL

- Chemical compounds with rings are highly frequent
- Fundamental inability of OWL to represent cycles
- OWL has the tree-model property: each consistent OWL ontology has at least one tree-shaped model

**Example**

\[
\text{Cyclobutane} \sqsubseteq \exists(=4)\text{hasAtom.}(\text{Carbon} \sqcap \exists(=2)\text{hasBond.}\text{Carbon})
\]

```
  C ─── C
     |   |
     C ─── C
```

OWL-based reasoning support

- Does cyclobutane contain a four-membered ring?
- Does benzaldehyde contain a benzene ring?
Chemical compounds with rings are highly frequent

Fundamental inability of OWL to represent cycles

OWL has the tree-model property: each consistent OWL ontology has at least one tree-shaped model

**Example**

Cyclobutane ⊑ ∃(=4) hasAtom.(Carbon ∩ ∃(=2) hasBond.Carbon)

![Diagram of cyclobutane and benzene rings with OWL-based reasoning support]
(Mis)representing rings with OWL

- Chemical compounds with rings are highly frequent
- Fundamental inability of OWL to represent cycles
- OWL has the tree-model property: each consistent OWL ontology has at least one tree-shaped model

**Example**

Cyclobutane ⊑ ∃(=4) hasAtom.(Carbon ⊓ ∃(=2) hasBond.Carbon)

- OWL-based reasoning support
Chemical compounds with rings are highly frequent.

Fundamental inability of OWL to represent cycles.

OWL has the tree-model property: each consistent OWL ontology has at least one tree-shaped model.

**Example**

\[
\text{Cyclobutane} \subseteq [\exists (=4) \text{hasAtom.}(\text{Carbon} \cap \exists (=2) \text{hasBond.} \text{Carbon})]
\]

- OWL-based reasoning support
  - Does cyclobutane contain a four-membered ring? ×
  - Does benzaldehyde contain a benzene ring? ×
Limitation of OWL to represent cycles (partially) remedied by extension of OWL with Description Graphs and rules [Motik et al., 2009]
OWL EXTENSIONS

- Limitation of OWL to represent cycles (partially) remedied by extension of OWL with Description Graphs and rules [Motik et al., 2009]
- A Description Graph represents structures by means of a directed labeled graph
Limitation of OWL to represent cycles (partially) remedied by extension of OWL with Description Graphs and rules [Motik et al., 2009]

A Description Graph represents structures by means of a directed labeled graph

Example

Cyclobutadiene

C \equiv C

|\hspace{1cm}\hspace{1cm}|\hspace{1cm}\hspace{1cm}|

C \equiv C

\forall \text{hasAtom} . (\text{Carbon} \sqcup \text{Hydrogen}) \sqsubseteq \text{Hydrocarbon}

Is cyclobutadiene a hydrocarbon?
Limitation of OWL to represent cycles (partially) remedied by extension of OWL with Description Graphs and rules [Motik et al., 2009]

A Description Graph represents structures by means of a directed labeled graph

Example

Is cyclobutadiene a hydrocarbon?
OWL Extensions

- Limitation of OWL to represent cycles (partially) remedied by extension of OWL with Description Graphs and rules [Motik et al., 2009]
- A Description Graph represents structures by means of a directed labeled graph

Example

Cyclobutadiene

Does cyclobutadiene have a conjugated four-membered ring?
OWL Extensions

- Limitation of OWL to represent cycles (partially) remedied by extension of OWL with Description Graphs and rules [Motik et al., 2009]
- A Description Graph represents structures by means of a directed labeled graph

**Example**

Cyclobutadiene

Does cyclobutadiene have a conjugated four-membered ring? √
OWL Extensions

- Limitation of OWL to represent cycles (partially) remedied by extension of OWL with Description Graphs and rules [Motik et al., 2009]
- A Description Graph represents structures by means of a directed labeled graph

**Example**

Cyclobutadiene

\[ \forall \text{hasAtom. (Carbon } \sqcup \text{Hydrogen) } \sqsubseteq \text{Hydrocarbon} \]

Is cyclobutadiene a hydrocarbon?
OWL Extensions

- Limitation of OWL to represent cycles (partially) remedied by extension of OWL with Description Graphs and rules [Motik et al., 2009]
- A Description Graph represents structures by means of a directed labeled graph

Example

∀ hasAtom. (Carbon △ Hydrogen) ⊆ Hydrocarbon

Cyclobutadiene

\[ \begin{align*}
\text{C} & \quad \text{C} \\
\text{C} & \quad \text{C} \\
\text{Carbon} & \quad \text{Carbon} \\
\text{Carbon} & \quad \text{Carbon} \\
\end{align*} \]

Oxygen
OWL EXTENSIONS

- Limitation of OWL to represent cycles (partially) remedied by extension of OWL with Description Graphs and rules [Motik et al., 2009]

- A Description Graph represents structures by means of a directed labeled graph

EXAMPLE

Cyclobutadiene

\[
\forall \text{hasAtom.} (\text{Carbon} \sqcup \text{Hydrogen}) \sqsubseteq \text{Hydrocarbon}
\]

Is cyclobutadiene a hydrocarbon?
OWL Extensions

- Limitation of OWL to represent cycles (partially) remedied by extension of OWL with Description Graphs and rules [Motik et al., 2009]

- A Description Graph represents structures by means of a directed labeled graph

**Example**

Cyclobutadiene

\[ \forall \text{hasAtom}.(\text{Carbon} \sqcup \text{Hydrogen}) \sqsubseteq \text{Hydrocarbon} \]

- Is cyclobutadiene a hydrocarbon? ✗
SWITCH TO LOGIC PROGRAMMING

- Shift from first-order logic semantics to logic programming semantics

- Replace classical negation with negation-as-failure to derive non-monotonic inferences.

- Double benefit:
  - Encode chemical classes based on the absence of information (e.g. hydrocarbons, inorganic molecules, saturated compounds, ...)
  - Represent rings with adequate precision (no tree-model property)
Switch to Logic Programming

- Shift from first-order logic semantics to logic programming semantics
- Replace classical negation with negation-as-failure to derive non-monotonic inferences.
Switch to Logic Programming

- Shift from first-order logic semantics to logic programming semantics
- Replace classical negation with negation-as-failure to derive non-monotonic inferences. Two different negations?
Switch to Logic Programming

- Shift from first-order logic semantics to logic programming semantics
- Replace classical negation with negation-as-failure to derive non-monotonic inferences. Two different negations?

Classical negation $\leftrightarrow$ Open-world assumption $\leftrightarrow$ Missing information treated as not known (OWL has it)
Switch to Logic Programming

- Shift from first-order logic semantics to logic programming semantics
- Replace classical negation with negation-as-failure to derive non-monotonic inferences. Two different negations?
  Classical negation \( \leftrightarrow \) Open-world assumption \( \leftrightarrow \) Missing information treated as *not known* (OWL has it)
  Negation as failure \( \leftrightarrow \) Closed-world assumption \( \leftrightarrow \) Missing information treated as *false* (LP has it)
Switch to Logic Programming

- Shift from first-order logic semantics to logic programming semantics
- Replace classical negation with negation-as-failure to derive non-monotonic inferences. Two different negations?
  - Classical negation ↔ Open-world assumption ↔ Missing information treated as *not known* (OWL has it)
  - Negation as failure ↔ Closed-world assumption ↔ Missing information treated as *false* (LP has it)

**Example**

\[
\text{Icecream}(x) \land \neg \text{Bananalscream}(x) \rightarrow \text{Likes}(alice, x)
\]

\[
\text{Icecream}(x) \land \textbf{not} \, \text{Bananalscream}(x) \rightarrow \text{Likes}(alice, x)
\]
Switch to Logic Programming

- Shift from first-order logic semantics to logic programming semantics
- Replace classical negation with negation-as-failure to derive non-monotonic inferences. Two different negations?

Classical negation $\leftrightarrow$ Open-world assumption $\leftrightarrow$ Missing information treated as not known (OWL has it)
Negation as failure $\leftrightarrow$ Closed-world assumption $\leftrightarrow$ Missing information treated as false (LP has it)

**Example**

\[
\text{Icecream}(x) \land \neg \text{Banananscream}(x) \rightarrow \text{Likes}(\text{alice}, x) \\
\mathcal{K} = \{ \text{Icecream}(\text{gnds}) \} \quad \mathcal{K} \not\models \text{Likes}(\text{alice}, \text{gnds}) \\
\text{Icecream}(x) \land \textbf{not} \text{Banananscream}(x) \rightarrow \text{Likes}(\text{alice}, x)
\]
Switch to Logic Programming

- Shift from first-order logic semantics to logic programming semantics
- Replace classical negation with negation-as-failure to derive non-monotonic inferences. Two different negations?
  
  Classical negation ↔ Open-world assumption ↔ Missing information treated as *not known* (*OWL* has it)
  
  Negation as failure ↔ Closed-world assumption ↔ Missing information treated as *false* (*LP* has it)

**Example**

\[
\text{Icecream}(x) \land \neg \text{BananaIscream}(x) \quad \rightarrow \quad \text{Likes}(\text{alice}, x)
\]

\[
\mathcal{K} = \{ \text{Icecream}(\text{gnds}) \}
\]

\[
\text{Icecream}(x) \land \textbf{not} \ \text{BananaIscream}(x)
\]

\[
\mathcal{K} = \{ \text{Icecream}(\text{gnds}) \}
\]

\[
\mathcal{K} \nmodels \text{Likes}(\text{alice}, gnds)
\]

Double benefit:

- Encode chemical classes based on the absence of information (e.g. hydrocarbons, inorganic molecules, saturated compounds, ...)
- Represent rings with adequate precision (no tree-model property)
SWITCH TO LOGIC PROGRAMMING

- Shift from first-order logic semantics to logic programming semantics
- Replace classical negation with negation-as-failure to derive non-monotonic inferences. Two different negations?
  - Classical negation ↔ Open-world assumption ↔ Missing information treated as not known (OWL has it)
  - Negation as failure ↔ Closed-world assumption ↔ Missing information treated as false (LP has it)
- Double benefit:
### Switch to Logic Programming

- Shift from first-order logic semantics to **logic programming semantics**
- Replace classical negation with **negation-as-failure** to derive non-monotonic inferences. Two different negations?
  - Classical negation $\leftrightarrow$ Open-world assumption $\leftrightarrow$ Missing information treated as *not known* (**OWL** has it)
  - Negation as failure $\leftrightarrow$ Closed-world assumption $\leftrightarrow$ Missing information treated as *false* (**LP** has it)
- Double benefit:
  - Encode chemical classes based on the **absence of information** (e.g. hydrocarbons, inorganic molecules, saturated compounds,...)
**Switch to Logic Programming**

- Shift from first-order logic semantics to **logic programming semantics**

- Replace classical negation with **negation-as-failure** to derive non-monotonic inferences. Two different negations?
  
  Classical negation ↔ Open-world assumption ↔ Missing information treated as *not known* (OWL has it)
  
  Negation as failure ↔ Closed-world assumption ↔ Missing information treated as *false* (LP has it)

- Double benefit:
  
  - Encode chemical classes based on the **absence of information** (e.g. hydrocarbons, inorganic molecules, saturated compounds, ...)
  
  - Represent **rings** with adequate precision (no tree-model property)
RESULTS OVERVIEW

- Design of an expressive logic-based formalism for modelling structured entities that we call Description Graphs Logic Programs (DGLPs)
Results overview

- Design of an expressive logic-based formalism for modelling structured entities that we call Description Graphs Logic Programs (DGLPs)

- Ensure decidability of reasoning tasks by formulating sufficient restrictive conditions
RESULTS OVERVIEW

- Design of an expressive logic-based formalism for modelling structured entities that we call **Description Graphs Logic Programs (DGLPs)**

- Ensure **decidability** of reasoning tasks by formulating sufficient restrictive conditions

- Develop a prototypical **implementation**
RESULTS OVERVIEW

- Design of an expressive logic-based formalism for modelling structured entities that we call Description Graphs Logic Programs (DGLPs)

- Ensure decidability of reasoning tasks by formulating sufficient restrictive conditions

- Develop a prototypical implementation

- Encouraging results of a preliminary evaluation
1. Motivation

2. Introducing DGLPs

3. Prototype

4. Conclusion
What is a DGLP ontology?

- The syntactic objects of a DGLP ontology:
WHAT IS A DGLP ONTOLOGY?

- The syntactic objects of a DGLP ontology:
  - Description graphs
WHAT IS A DGLP ONTOLOGY?

- The syntactic objects of a DGLP ontology:
  - Description graphs

EXAMPLE

Cyclobutane

\[
\begin{align*}
C & \quad C \\
\text{Carbon} & \quad \text{Carbon} \\
\text{Carbon} & \quad \text{Carbon} \\
\text{Oxygen} & \quad \text{Oxygen}
\end{align*}
\]

\[
\text{Cyclobutane} \quad \text{C} \quad \text{C}
\]

\[
\text{Carbon} \quad \text{Carbon} \quad \text{Carbon} \quad \text{Oxygen} \quad \text{Oxygen}
\]

\[
\text{C} \quad \text{C}
\]

\[
\text{O} \quad \text{O}
\]
WHAT IS A DGLP ONTOLOGY?

The syntactic objects of a DGLP ontology:

- Description graphs
- Function-free FOL Horn rules
What is a DGLP ontology?

- The syntactic objects of a DGLP ontology:
  - Description graphs
  - Function-free FOL Horn rules

**Example**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bond(x, y)</td>
<td>Bond(y, x)</td>
</tr>
<tr>
<td>SingleBond(x, y)</td>
<td>Bond(x, y)</td>
</tr>
</tbody>
</table>
What is a DGLP ontology?

- The syntactic objects of a DGLP ontology:
  - Description graphs
  - Function-free FOL Horn rules

**Example**

<table>
<thead>
<tr>
<th>Description</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bond(x, y)</td>
<td>→ Bond(y, x)</td>
</tr>
<tr>
<td>SingleBond(x, y)</td>
<td>→ Bond(x, y)</td>
</tr>
</tbody>
</table>

- Rules with negation-as-failure
WHAT IS A DGLP ONTOLOGY?

- The syntactic objects of a DGLP ontology:
  - Description graphs
  - Function-free FOL Horn rules

**Example**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bond(x, y) → Bond(y, x)</td>
<td></td>
</tr>
<tr>
<td>SingleBond(x, y) → Bond(x, y)</td>
<td></td>
</tr>
</tbody>
</table>

- Rules with negation-as-failure

**Example**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasAtom(x, y) ∧ Carbon(y) → HasCarbon(x)</td>
<td></td>
</tr>
<tr>
<td>Molecule(x) ∨ not HasCarbon(x) → Inorganic(x)</td>
<td></td>
</tr>
</tbody>
</table>
What is a DGLP ontology?

- The syntactic objects of a DGLP ontology:
  - Description graphs
  - Function-free FOL Horn rules

**Example**

| Bond(x, y)   | ➔ | Bond(y, x) |
| SingleBond(x, y) | ➔ | Bond(x, y) |

- Rules with negation-as-failure

**Example**

| HasAtom(x, y) ∧ Carbon(y) | ➔ | HasCarbon(x) |
| Molecule(x) ∧ not HasCarbon(x) | ➔ | Inorganic(x) |

- Facts
What is a DGLP ontology?

- The syntactic objects of a DGLP ontology:
  - Description graphs
  - Function-free FOL Horn rules

**Example**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bond((x, y))</td>
<td>(\rightarrow) Bond((y, x))</td>
</tr>
<tr>
<td>SingleBond((x, y))</td>
<td>(\rightarrow) Bond((x, y))</td>
</tr>
</tbody>
</table>

- Rules with negation-as-failure

**Example**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>HasAtom((x, y)) (\wedge) Carbon((y))</td>
<td>(\rightarrow) HasCarbon((x))</td>
</tr>
<tr>
<td>Molecule((x)) (\wedge) not HasCarbon((x))</td>
<td>(\rightarrow) Inorganic((x))</td>
</tr>
</tbody>
</table>

- Facts

**Example**

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyclobutane((c_1)), Dinitrogen((c_2)), ...</td>
</tr>
</tbody>
</table>
SEMANTICS OF DGs

- Translate DGs into logic programs with function symbols
Translate DGs into logic programs with function symbols
**Semantics of DGs**

- Translate DGs into logic programs with function symbols

**Example**

Let's translate the DG for Cyclobutane into a logic program.

\[
\text{Cyclobutane}(x) \rightarrow G_{cb}(x, f_1(x), f_2(x), f_3(x), f_4(x))
\]

\[
G_{cb}(x, y_1, y_2, y_3, y_4) \rightarrow \text{Cyclobutane}(x) \land \\
\text{Carbon}(y_1) \land \text{Carbon}(y_2) \land \\
\text{Carbon}(y_3) \land \text{Carbon}(y_4) \land \\
\text{HasAtom}(x, y_1) \land \text{Bond}(y_1, y_2) \land \\
\text{HasAtom}(x, y_2) \land \text{Bond}(y_2, y_3) \land \\
\text{HasAtom}(x, y_3) \land \text{Bond}(y_3, y_4) \land \\
\text{HasAtom}(x, y_4) \land \text{Bond}(y_4, y_1)
\]
SEMANITCS OF DGs

- Translate DGs into logic programs with function symbols

**Example**

\[
\text{Cyclobutane}(x) \rightarrow \text{G}_{\text{cb}}(x, f_1(x), f_2(x), f_3(x), f_4(x))
\]

\[
\text{G}_{\text{cb}}(x, y_1, y_2, y_3, y_4) \rightarrow \text{Cyclobutane}(x) \land \\
\text{Carbon}(y_1) \land \text{Carbon}(y_2) \land \\
\text{Carbon}(y_3) \land \text{Carbon}(y_4) \land \\
\text{HasAtom}(x, y_1) \land \text{Bond}(y_1, y_2) \land \\
\text{HasAtom}(x, y_2) \land \text{Bond}(y_2, y_3) \land \\
\text{HasAtom}(x, y_3) \land \text{Bond}(y_3, y_4) \land \\
\text{HasAtom}(x, y_4) \land \text{Bond}(y_4, y_1)
\]

- Function symbols allow for schema-level reasoning
CLASSIFYING OBJECTS

EXAMPLE

\[ \text{Molecule } \sqsubseteq \forall \text{hasAtom.}(\text{Carbon } \sqcup \text{Hydrogen}) \sqsubseteq \text{Hydrocarbon} \]
CLASSIFYING OBJECTS

EXAMPLE

Molecule ⊓ ∀ hasAtom.(Carbon ⊔ Hydrogen) ⊑ Hydrocarbon

⇓

Is cyclobutane a hydrocarbon?
**Example**

Molecule ⊆ ∀ hasAtom. (Carbon ⊔ Hydrogen) ⊑ Hydrocarbon

⇒

Molecule ⊆ ¬∃ hasAtom. (¬Carbon) ⊔ (¬Hydrogen)) ⊑ Hydrocarbon

Is cyclobutane a hydrocarbon?
**Example**

Molecule $\forall \text{hasAtom.}(\text{Carbon} \sqcup \text{Hydrogen}) \sqsubseteq \text{Hydrocarbon}$

$\Downarrow$

Molecule $\forall \neg \exists \text{hasAtom.}(\neg \text{Carbon} \sqcap \neg \text{Hydrogen}) \sqsubseteq \text{Hydrocarbon}$

$\Downarrow$

Is cyclobutane a hydrocarbon?
Classifying objects

Example

\[
\text{Molecule} \sqcap \forall \text{hasAtom.}(\text{Carbon} \sqcup \text{Hydrogen}) \sqsubseteq \text{Hydrocarbon}
\]

\[
\Downarrow
\]

\[
\text{Molecule} \sqcap \neg \exists \text{hasAtom.}((\neg \text{Carbon}) \sqcap (\neg \text{Hydrogen})) \sqsubseteq \text{Hydrocarbon}
\]

\[
\Downarrow
\]

\[
\text{Molecule}(x) \land \text{HasAtom}(x, y) \land \text{not Carbon}(y) \land \text{not Hydrogen}(y) 
\rightarrow \text{NotHydroCarbon}(x)
\]

\[
\text{Molecule}(x) \land \text{not NotHydroCarbon}(x) \rightarrow \text{HydroCarbon}(x)
\]
**Classifying objects**

**Example**

\[ Molecule \sqcap \forall \text{hasAtom.}(\text{Carbon} \sqcup \text{Hydrogen}) \sqsubseteq \text{Hydrocarbon} \]

\[ \Downarrow \]

\[ Molecule \sqcap \neg \exists \text{hasAtom.}((\neg \text{Carbon}) \sqcap (\neg \text{Hydrogen})) \sqsubseteq \text{Hydrocarbon} \]

\[ \Downarrow \]

\[ Molecule(x) \land \text{HasAtom}(x, y) \land \text{not Carbon}(y) \land \text{not Hydrogen}(y) \rightarrow \text{NotHydroCarbon}(x) \]

\[ Molecule(x) \land \text{not NotHydroCarbon}(x) \rightarrow \text{HydroCarbon}(x) \]
EXAMPLE

\[
\begin{align*}
\text{Molecule} & \sqcap \forall \text{hasAtom}. (\text{Carbon} \sqcup \text{Hydrogen}) \sqsubseteq \text{Hydrocarbon} \\
\quad \Downarrow \\
\text{Molecule} & \sqcap \neg \exists \text{hasAtom}. ((\neg \text{Carbon}) \sqcap (\neg \text{Hydrogen})) \sqsubseteq \text{Hydrocarbon} \\
\quad \Downarrow \\
\text{Molecule}(x) & \land \text{HasAtom}(x, y) \land \text{not Carbon}(y) \land \text{not Hydrogen}(y) \\
& \rightarrow \text{NotHydroCarbon}(x) \\
\text{Molecule}(x) & \land \text{not NotHydroCarbon}(x) \rightarrow \text{HydroCarbon}(x)
\end{align*}
\]

Is cyclobutane a hydrocarbon?
**EXAMPLE**

Molecule \( \sqsubseteq \forall \text{hasAtom}.(\text{Carbon} \sqcup \text{Hydrogen}) \sqsubseteq \text{Hydrocarbon} \)

\[ \downarrow \]

Molecule \( \sqsubseteq \neg \exists \text{hasAtom}.((\neg \text{Carbon}) \sqcap (\neg \text{Hydrogen})) \sqsubseteq \text{Hydrocarbon} \)

\[ \downarrow \]

Molecule(\( x \)) \( \land \) \( \text{HasAtom}(x, y) \land \text{not Carbon}(y) \land \text{not Hydrogen}(y) \rightarrow \text{NotHydroCarbon}(x) \)

Molecule(\( x \)) \( \land \) \( \text{not NotHydroCarbon}(x) \rightarrow \text{HydroCarbon}(x) \)

**Is cyclobutane a hydrocarbon?** ✔
Classifying objects

**Example**

\[
\text{Molecule}(x) \land \bigwedge_{1 \leq i \leq 4} \text{HasAtom}(x, y_i) \land \bigwedge_{1 \leq i \leq 3} \text{Bond}(y_i, y_{i+1}) \land \\
\text{Bond}(y_4, y_1) \land \bigwedge_{1 \leq i < j \leq 4} \text{not } y_i = y_j \\
\rightarrow \text{MoleculeWith4MemberedRing}(x)
\]
Example

\[
\text{Molecule}(x) \land \bigwedge_{1 \leq i \leq 4} \text{HasAtom}(x, y_i) \land \bigwedge_{1 \leq i \leq 3} \text{Bond}(y_i, y_{i+1}) \land \\
\text{Bond}(y_4, y_1) \land \bigwedge_{1 \leq i < j \leq 4} \neg y_i = y_j
\]

\[\rightarrow \text{MoleculeWith4MemberedRing}(x)\]
Classifying objects

Example

\[
\text{Molecule}(x) \land \bigwedge_{1 \leq i \leq 4} \text{HasAtom}(x, y_i) \land \bigwedge_{1 \leq i \leq 3} \text{Bond}(y_i, y_{i+1}) \land \\
\text{Bond}(y_4, y_1) \land \bigwedge_{1 \leq i < j \leq 4} \text{not } y_i = y_j \\
\rightarrow \text{MoleculeWith4MemberedRing}(x)
\]

Does cyclobutane contain a four-membered ring? ✓
(UN)DECIDABILITY

- Logic programs with function symbols can axiomatise infinite structures
(Un)decidability

- Logic programs with function symbols can axiomatise infinite structures
- Reasoning with DGLP ontologies is trivially undecidable
(UN)DECIDABILITY

- Logic programs with function symbols can axiomatise infinite structures
- Reasoning with DGLP ontologies is \textit{trivially undecidable}

\textbf{Example}

\begin{align*}
A(x) & \rightarrow G(x, f_1(x), f_2(x)) \\
G(x, y_1, y_2) & \rightarrow A(y_1) \land A(y_2)
\end{align*}

\{A(a), G(a, f_1(a), f_2(a)), A(f_1(a)), A(f_2(a)), \ldots\}
(Un)decidability

- Logic programs with function symbols can axiomatise infinite structures.
- Reasoning with DGLP ontologies is trivially undecidable.
- We are only interested in bounded structures.
(UN)DECIDABILITY

- Logic programs with function symbols can axiomatise infinite structures
- Reasoning with DGLP ontologies is trivially undecidable
- We are only interested in bounded structures

EXAMPLE

[Diagram showing molecular structures with labels for Methyl, Carboxyl, Carbonyl, Hydroxyl, and Acetic Acid with numbered connections.]
SYNTACTIC ACYCLICITY CONDITIONS

- Problem extensively studied, e.g. in theory of databases
SYNTACTIC ACYCLICITY CONDITIONS

- Problem extensively studied, e.g. in theory of databases
- Various syntax-based acyclicity conditions
SYNTACTIC ACYCLICITY CONDITIONS

- Problem extensively studied, e.g. in theory of databases
- Various syntax-based acyclicity conditions
  - Weak acyclicity [Fagin et al., 2002]
  - Super-weak acyclicity [Marnette, 2009]
  - Joint acyclicity [Krötzsch et al., 2011]
SYNTACTIC ACYCLICITY CONDITIONS

- Problem extensively studied, e.g. in theory of databases
- Various syntax-based acyclicity conditions
  - Weak acyclicity [Fagin et al., 2002]
  - Super-weak acyclicity [Marnette, 2009]
  - Joint acyclicity [Krötzsch et al., 2011]

EXAMPLE

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

G_{AA}(x_1, x_2, x_3) \rightarrow AceticAcid(x_1) \land Methyl(x_2) \land Carboxyl(x_3) \land HasPart(x_1, x_2) \land HasPart(x_1, x_3)

Methyl(x_2) \land Carboxyl(x_3) \land HasPart(x_1, x_2) \land HasPart(x_1, x_3)
\rightarrow G_{AA}(x_1, x_2, x_3)

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

G_{cxl}(x_1, x_2, x_3) \rightarrow Carboxyl(x_1) \land Carbonyl(x_2) \land Hydroxyl(x_3) \land HasPart(x_1, x_2) \land HasPart(x_1, x_3)
SYNTACTIC ACYCLICITY CONDITIONS

- Problem extensively studied, e.g. in theory of databases
- Various **syntax-based** acyclicity conditions
  - Weak acyclicity [Fagin et al., 2002]
  - Super-weak acyclicity [Marnette, 2009]
  - Joint acyclicity [Krötzsch et al., 2011]

**Example**

\[
\begin{align*}
\text{AceticAcid}(x) & \rightarrow G_{AA}(x, f_1(x), f_2(x)) \\
G_{AA}(x_1, x_2, x_3) & \rightarrow \text{AceticAcid}(x_1) \land \text{Methyl}(x_2) \land \text{Carboxyl}(x_3) \land \\
& \quad \text{HasPart}(x_1, x_2) \land \text{HasPart}(x_1, x_3) \\
\text{Methyl}(x_2) \land \text{Carboxyl}(x_3) \land \text{HasPart}(x_1, x_2) \land \text{HasPart}(x_1, x_3) & \rightarrow G_{AA}(x_1, x_2, x_3) \\
\text{Carboxyl}(x) & \rightarrow G_{cxl}(x, g_1(x), g_2(x)) \\
G_{cxl}(x_1, x_2, x_3) & \rightarrow \text{Carboxyl}(x_1) \land \text{Carbonyl}(x_2) \land \text{Hydroxyl}(x_3) \land \\
& \quad \text{HasPart}(x_1, x_2) \land \text{HasPart}(x_1, x_3)
\end{align*}
\]
SYNTACTICACYCLICITYCONDITIONS

- Problem extensively studied, e.g. in theory of databases
- Various syntax-based acyclicity conditions
  - Weak acyclicity [Fagin et al., 2002]
  - Super-weak acyclicity [Marnette, 2009]
  - Joint acyclicity [Krötzsch et al., 2011]

**Example**

\[
\begin{align*}
\text{AceticAcid}(x) & \rightarrow G_{\text{AA}}(x, f_1(x), f_2(x)) \\
G_{\text{AA}}(x_1, x_2, x_3) & \rightarrow \text{AceticAcid}(x_1) \land \text{Methyl}(x_2) \land \text{Carboxyl}(x_3) \land \\
& \quad \text{HasPart}(x_1, x_2) \land \text{HasPart}(x_1, x_3) \\
\text{Methyl}(x_2) \land \text{Carboxyl}(x_3) \land \text{HasPart}(x_1, x_2) \land \text{HasPart}(x_1, x_3) & \rightarrow G_{\text{AA}}(x_1, x_2, x_3) \\
\text{Carboxyl}(x) & \rightarrow G_{\text{cxl}}(x, g_1(x), g_2(x)) \\
G_{\text{cxl}}(x_1, x_2, x_3) & \rightarrow \text{Carboxyl}(x_1) \land \text{Carbonyl}(x_2) \land \text{Hydroxyl}(x_3) \land \\
& \quad \text{HasPart}(x_1, x_2) \land \text{HasPart}(x_1, x_3)
\end{align*}
\]
Semantic acyclicity

- Define a transitive and irreflexive graph ordering which specifies which graph instances may imply the existence of other graph instances.
SEMANTIC ACYCLICITY

- Define a transitive and irreflexive graph ordering which specifies which graph instances may imply the existence of other graph instances

EXAMPLE

AceticAcid

Methyl Carboxyl

Carboxyl

Carbonyl Hydroxyl

AceticAcid ≺ Carboxyl
SEMANTIC ACYCLICITY

- Define a transitive and irreflexive graph ordering which specifies which graph instances may imply the existence of other graph instances.

- Extented the logic program with rules that are triggered by violation of the graph ordering.
Semantic Acyclicity

- Define a transitive and irreflexive graph ordering which specifies which graph instances may imply the existence of other graph instances.

- Extended the logic program with rules that are triggered by violation of the graph ordering.

- If a repetitive construction of graph instances is detected during reasoning, then derive Cycle.
**Semantic Acyclicity**

- Define a transitive and irreflexive graph ordering which specifies which graph instances may imply the existence of other graph instances.

- Extend the logic program with rules that are triggered by violation of the graph ordering.

- If a repetitive construction of graph instances is detected during reasoning, then derive **Cycle**

**Example**

\[ G_{AA}(x_1, x_2, x_3) \land \text{AceticAcid}(x_2) \rightarrow \text{Cycle} \]
**Semantic Acyclicity**

- Define a transitive and irreflexive graph ordering which specifies which graph instances may imply the existence of other graph instances.

- Extended the logic program with rules that are triggered by violation of the graph ordering.

- If a repetitive construction of graph instances is detected during reasoning, then derive *Cycle*.

- A DGLP ontology is *semantically acyclic* if it does not entail *Cycle*. 
**Semantic Acyclicity**

- Define a transitive and irreflexive graph ordering which specifies which graph instances may imply the existence of other graph instances.

- Extend the logic program with rules that are triggered by violation of the graph ordering.

- If a repetitive construction of graph instances is detected during reasoning, then derive *Cycle*.

- A DGLP ontology is **semantically acyclic** if it does not entail *Cycle*.

- DGLP ontology with acetic acid is semantically acyclic **✓**
TECHNICAL RESULTS

- Decidability for negation-free DGLP ontologies

- Decidability for DGLP ontologies with stratified negation

DGLP ontologies with stratified negation capture a wide range of chemical classes:

- Is dinitrogen inorganic?
- Does cyclobutane contain a four-membered ring?
- Is acetylene a hydrocarbon?
- Does benzaldehyde contain a benzene ring?
**Technical results**

- Decidability for *negation-free* DGLP ontologies

- Decidability for DGLP ontologies with *stratified negation*
Decidability for negation-free DGLP ontologies

Decidability for DGLP ontologies with stratified negation

DGLP ontologies with stratified negation capture a wide range of chemical classes:
TECHNICAL RESULTS

- Decidability for **negation-free** DGLP ontologies

- Decidability for DGLP ontologies with **stratified negation**

- DGLP ontologies with stratified negation capture a wide range of chemical classes:
  - Is dinitrogen **inorganic**?
  - Does cyclobutane contain a **four-membered ring**?
  - Is acetylene a **hydrocarbon**?
  - Does benzaldehyde contain a **benzene ring**?
TECHNICAL RESULTS

- Decidability for **negation-free** DGLP ontologies

- Decidability for DGLP ontologies with **stratified negation**

DGLP ontologies with stratified negation capture a wide range of chemical classes:

- Is dinitrogen **inorganic**? ✓

- Does cyclobutane contain a **four-membered ring**? ✓

- Is acetylene a **hydrocarbon**? ✓

- Does benzaldehyde contain a **benzene ring**? ✓
Outline

1 Motivation

2 Introducing DGLPs

3 Prototype

4 Conclusion
Experimental setting

- Data extracted from ChEBI
EXPERIMENTAL SETTING

- Data extracted from ChEBI
- XSB logic programming engine

Chemical classes that were modelled:
- Hydrocarbons
- Inorganic molecules
- Molecules with exactly two carbons
- Molecules with a four-membered ring
- Molecules with a benzene

Preliminary evaluation: size of the ontologies ranging from 10 to 70 molecules
EXPERIMENTAL SETTING

- Data extracted from ChEBI
- XSB logic programming engine
- Chemical classes that were modelled:
  - Hydrocarbons
  - Inorganic molecules
  - Molecules with exactly two carbons
  - Molecules with a four-membered ring
  - Molecules with a benzene
EXPERIMENTAL SETTING

- Data extracted from ChEBI
- XSB logic programming engine

Chemical classes that were modelled:
- Hydrocarbons
- Inorganic molecules
- Molecules with exactly two carbons
- Molecules with a four-membered ring
- Molecules with a benzene

Preliminary evaluation: size of the ontologies ranging from 10 to 70 molecules
IMPLEMENTATION RESULTS

- All DGLP ontologies were found acyclic
IMPLEMENTATION RESULTS

- All DGLP ontologies were found acyclic
- Molecules classified as expected
IMPLEMENTATION RESULTS

- All DGLP ontologies were found **acyclic**
- Molecules classified as expected

<table>
<thead>
<tr>
<th>No mol.</th>
<th>T₁</th>
<th>T₂</th>
<th>T₃</th>
<th>T₄</th>
<th>T₅</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>0.36</td>
<td>0.02</td>
<td>2.47</td>
</tr>
<tr>
<td>20</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>0.02</td>
<td>2.07</td>
<td>0.21</td>
<td>10.66</td>
</tr>
<tr>
<td>30</td>
<td>0.01</td>
<td>&lt; 0.01</td>
<td>0.03</td>
<td>2.23</td>
<td>0.23</td>
<td>13.85</td>
</tr>
<tr>
<td>40</td>
<td>0.01</td>
<td>&lt; 0.01</td>
<td>0.04</td>
<td>2.58</td>
<td>0.29</td>
<td>19.06</td>
</tr>
<tr>
<td>50</td>
<td>0.01</td>
<td>0.01</td>
<td>0.06</td>
<td>3.55</td>
<td>0.41</td>
<td>27.15</td>
</tr>
<tr>
<td>60</td>
<td>0.04</td>
<td>0.02</td>
<td>0.51</td>
<td>109.88</td>
<td>21.68</td>
<td>300.84</td>
</tr>
<tr>
<td>70</td>
<td>0.06</td>
<td>0.03</td>
<td>0.75</td>
<td>172.14</td>
<td>35.08</td>
<td>447.12</td>
</tr>
</tbody>
</table>

- **T₁**: hydrocarbons, **T₂**: inorganic molecules
- **T₃**: molecules with exactly two carbons
- **T₄**: molecules with a four-membered ring
- **T₅**: molecules with a benzene
IMPLEMENTATION RESULTS

- All DGLP ontologies were found acyclic
- Molecules classified as expected

<table>
<thead>
<tr>
<th>No mol.</th>
<th>T₁</th>
<th>T₂</th>
<th>T₃</th>
<th>T₄</th>
<th>T₅</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>0.36</td>
<td>0.02</td>
<td>2.47</td>
</tr>
<tr>
<td>20</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>0.02</td>
<td>2.07</td>
<td>0.21</td>
<td>10.66</td>
</tr>
<tr>
<td>30</td>
<td>0.01</td>
<td>&lt; 0.01</td>
<td>0.03</td>
<td>2.23</td>
<td>0.23</td>
<td>13.85</td>
</tr>
<tr>
<td>40</td>
<td>0.01</td>
<td>&lt; 0.01</td>
<td>0.04</td>
<td>2.58</td>
<td>0.29</td>
<td>19.06</td>
</tr>
<tr>
<td>50</td>
<td>0.01</td>
<td>0.01</td>
<td>0.06</td>
<td>3.55</td>
<td>0.41</td>
<td>27.15</td>
</tr>
<tr>
<td>60</td>
<td>0.04</td>
<td>0.02</td>
<td>0.51</td>
<td>109.88</td>
<td>21.68</td>
<td>300.84</td>
</tr>
<tr>
<td>70</td>
<td>0.06</td>
<td>0.03</td>
<td>0.75</td>
<td>172.14</td>
<td>35.08</td>
<td>447.12</td>
</tr>
</tbody>
</table>

- T₁: hydrocarbons, T₂: inorganic molecules
- T₃: molecules with exactly two carbons
- T₄: molecules with a four-membered ring
- T₅: molecules with a benzene
1 Motivation

2 Introducing DGLPs

3 Prototype

4 Conclusion
RESULTS OVERVIEW

- Expressive and decidable formalism for modelling structured objects

RESULTS OVERVIEW

- Expressive and decidable formalism for modelling structured objects

- Novel acyclicity condition for logic programs with restricted use of function symbols
Results Overview

- Expressive and decidable formalism for modelling structured objects
- Novel acyclicity condition for logic programs with restricted use of function symbols
- Development of a prototype for the automatic classification of chemical molecules

Encouraging results of a preliminary evaluation
RESULTS OVERVIEW

- Expressive and decidable formalism for modelling structured objects

- Novel acyclicity condition for logic programs with restricted use of function symbols

- Development of a prototype for the automatic classification of chemical molecules

- Encouraging results of a preliminary evaluation
FUTURE DIRECTIONS

- Extend suggested DGLP formalism:
FUTURE DIRECTIONS

- Extend suggested DGLP formalism:
  - Relax stratifiability criteria for negation
  - Disjunctive rule heads
  - Integrate logic programming rules with use of numerical values

Thank you for listening. Questions?
**Future Directions**

- Extend suggested DGLP formalism:
  - Relax stratifiability criteria for negation
  - Disjunctive rule heads
  - Integrate logic programming rules with use of numerical values

- Optimise our prototype towards a fully-scalable classification system for structured objects
FUTURE DIRECTIONS

- Extend suggested DGLP formalism:
  - Relax stratifiability criteria for negation
  - Disjunctive rule heads
  - Integrate logic programming rules with use of numerical values

- Optimise our prototype towards a fully-scalable classification system for structured objects

- Thank you for listening. Questions?