Hybrid KR languages

Combining Rules and DLs

Bad news: these formalisms don’t get along too well!

- The basic properties that ensure their good computational behavior are orthogonal
  - In DLs, we have infinite, regular structures that are somehow tree-shaped
  - In ASP, we have finite, arbitrary shaped structures that are somehow tree-shaped
- If we combine them, both properties are lost and things immediately become undecidable!

Fragments of $\mathcal{ALC}$ combined with positive, normal ASP programs (Datalog) are already undecidable!

Can we do something?

To regain decidability, restrict the interaction between both parts

Possible Solutions

1. Trivial languages around the intersection of both formalisms (of course, they don’t combine the strengths of both)

2. Rules on top of ontologies: ASP rules can look at properties of (the models of) a DL KB, but can not modify them

3. Special safety conditions that restrict the way the rules can propagate information to the (possibly infinitely many) unnamed objects induced by the KB

4. Logic programs that export reasoning tasks to an ontology

The choice of the approach is purpose specific

Combining the strengths of both formalisms is a major challenge

Knowledge Representation Formalisms

- Description Logics
  - Well suited for describing complex structures of knowledge
  - Most prominent formalisms for ontologies and semantic technologies
    - Poor as query languages for data access
    - Their classical monotonic semantics makes them inadequate for common sense reasoning
- Answer set programming, and related rule-based languages
  - Good for declarative problem solving, powerful as query languages
  - Their non-monotonic semantics makes them well suited for common sense reasoning and dealing with incomplete knowledge
  - Have always been at the core of AI, applied in planning, abductive reasoning, etc.
    - Can not (naturally) describe the complex knowledge structures expressible in DLs

Hybrid KR languages

A glimpse beyond: Hybrid KR languages combining DLs and ASP

Magdalena Ortiz

Knowledge Base Systems Group
Institute of Information Systems
ortiz@kr.tuwien.ac.at

15 December 2010
Some Hybrid Languages

Some examples of the above approaches are:

1. **DLP**, which results by intersecting datalog rules (with open world semantics) and **ALC** [Horrocks and Patel-Schneider, 1998]
2. **CARIN** [Levy and Rousset, 98]
3. **ALC-log** and its variations [Rosati, 99]
4. **dl-programs** [Eiter, Lukasiewicz, Schindlauer & Tompits, 04]

**ALC-log**

Combines Datalog with **ALC**
- Only DL concepts are allowed in the bodies of the rules
- **DL safety**: the variables of the DL atoms in the body of a rule must also appear in the non-DL atoms

Query answering algorithm:
- Construct DL constraints for a given query using backward chaining.
- Use an **ALC** tableau reasoner to check that the disjunction of the DL constraints is entailed by the ontology

Many later extensions extended the language by considering more expressive DLs, relaxed the notions of safety, etc.

**CARIN**

A family of languages combining Datalog with any DL contained in **ALC N R**.

- Any concept or role instance query may appear in the body of a rule
- No restriction on the variables of the DL queries
- Undecidable in general, but can be made decidable by imposing different kinds of restrictions
- For example, the rules are limited to be non-recursive (popular CARIN variant)
- For non-recursive CARIN reasoning is similar to query answering

Later extended to DLs around **SHIQ**

**Example**

- **DL component**:

  \[
  \begin{align*}
  \text{european} \sqcap \text{american} & \sqsubseteq \bot \\
  \text{european} \text{Associate} & \equiv \exists \text{associate.european} \\
  \text{american} \text{Associate} & \equiv \exists \text{associate.american} \\
  \text{international} & \equiv \text{european} \text{Associate} \sqcup \text{american} \text{Associate} \\
  \text{noFellowCompany} & \equiv \forall \text{associate.} \neg \text{american international(b)}
  \end{align*}
  \]

- **Rule component**:

  \[
  \begin{align*}
  \text{price}(X, \text{usa}, \text{high}) & :\neg \text{madeBy}(X, Y), \text{noFellowCompany}(Y) \\
  \text{price}(X, \text{usa}, \text{high}) & :\neg \text{madeBy}(X, Y), \text{monopoly}(Y, X, \text{usa}), \\
  & \text{associate}(Y, Z), \text{american}(Z) \\
  \text{madeBy}(a, b) & \text{monopoly}(b, a, \text{usa}) \\
  \text{Query} & \text{price}(a, \text{usa}, \text{high})?
  \end{align*}
  \]
dl-programs

- Normal ASP programs:
  - the head is an atom
  - the body is a conjunction of possibly negated atoms

- but the body may contain *dl-atoms* that are evaluated (and possibly update) a DL KB

- reasoning feasible by interfacing an ASP engine with a DL reasoner