Ontologies Rules!

Tomas Lööw

2011

Ontologies and Rules

Tomas Lööw
2011

Rules

- If-then-statements.
- Datalog, a computationally simple subset of Prolog.

  bird(X) ← animal(X), has_wings(X).

  happy(X) ← married(X,Y), loves(X,Y), loves(Y,X).

  happy(X) ← student(X), subject(X, computer-science).
- Many different extensions, negation, disjunction etc.

The layers of the Semantic Web

- RDF: Describes relations between objects

  <*Subject*> <*relation*> <*object*> triples
- RDFS: Extension to RDF.

  Can define simple taxonomies, ranges, etc.
- OWL: Describes relations between *concepts.*
- RIF: Description of deduction rules.

  Work in progress.

Combining Ontologies and rule engines

- The goal:
- Use knowledge from Ontologies inside a rules engine.
- Use deductions from rules to add knowledge to Ontologies.

Sounds easy, right?

- In the ontology we have:

  student = undergrad_student ∪ *grad_student*

  undergrad_student(Linus), grad_student(Linnéa), student(Lukas)

- In the rules engine we have:

  takes_courses(X) ← *grad_student(X).*

  takes_courses(X) ← *undergrad_student(X).*

- What about the query ”takes_courses(Lukas)?”
- A sophisticated integration between Ontology and Rules engine is needed.

Problems with integration.

- Different axiomatic grounds. FOL vs non-monotonic logic.
- Many possible designs. Homogeneous/heterogeneous. Tight/Loose coupling, syntax, etc.
- Tractability concerns.

Heterogeneous integration.

- Rules and ontology facts are handled the same way.
- One computational engine.
- For example, treat rules as FOL statements and add to the DL-base.
- Problems with efficiency, datalog computations is simpler than general DL computations.

Homogeneous integration

- Ontology knowledge and rules are treated explicitly different in the language.
- Can be constructed by coupling two engines that communicate. Might allow reuse of existing engines easier.

Semantics of Rules

- Two main kinds of semantics
- Strong Answer semantics:

  Calculates many different, incompatible, models.
- Well Founded semantics

  Calculates *one* model which may be incomplete.

DL-programs

- Heterogeneous integration
- *(P,L)* where *P* is a logic program and *L* is a description logic base.
- Bidirectional flow between Logic Program and Ontology.

  The Logic Program can query the Ontology but can also add knowledge to a *copy* of it.

Syntax

- predicate(atom).
- predicate(X) ← some(X),
  other(X,Y),predicates(Y).
- predicate(X) ← DL[Class](X), other(X)
- predicate(X) ←
  DL[Rel $\uplus$ p, Rel2 $\cup\!\!-$ p2;Class](X)

Example from the paper, Knowledgebase.

≥ 1 wired $\sqsubseteq$ Node; $\top \sqsubseteq \forall$wired.Node;

wired = wired$^-$;

≥ 4 wired $\sqsubseteq$ HighTrafficNode;

n1 = n2 = n3 = n4 = n5;

Node(n1); Node(n2); Node(n3); Node(n4); Node(n5);

wired(n1, n2); wired(n2, n3); wired(n2, n4);

wired(n2, n5); wired(n3, n4); wired(n3, n5).

Example, figure

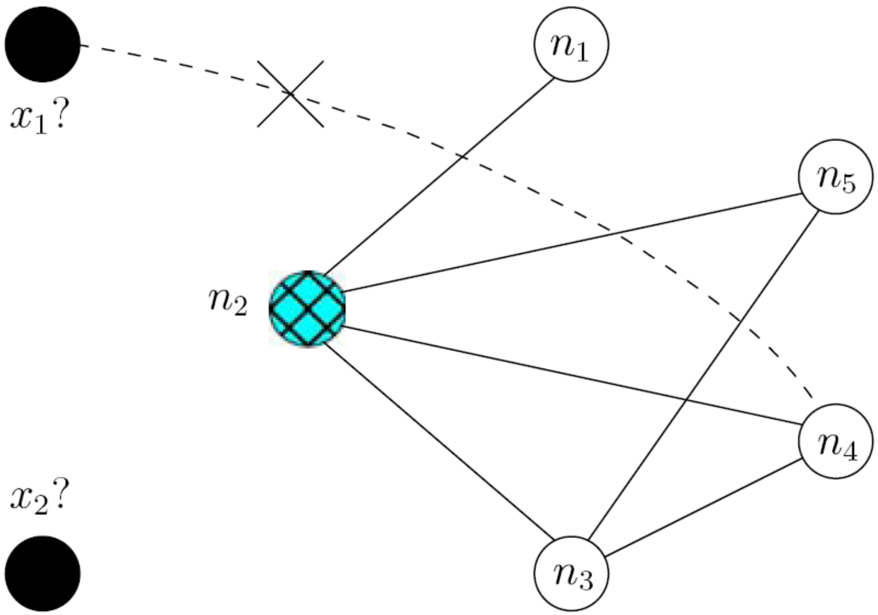

Fig. 2. Hightraffic network

Example, Program

newnode(x1).

newnode(x2).

overloaded(X) ← DL[wired ⊎ connect; HighTrafficNode](X).

connect(X, Y) ← newnode(X),DL[Node](Y),
not overloaded(Y), not excl (X, Y).

excl(X, Y) ← connect (X,Z), DL[Node](Y), Y ≠ Z.

excl(X, Y) ← connect (Z, Y), newnode(Z),
newnode(X), Z ≠ X.

excl(x1, n4).

Semantics

Strong-answer semantics:

- For a rule *r*, $B^+(r)$ is the positive clauses and $B^-(r)$ is the negative clauses.
- $I \models_L a$ if (informally) *L* with the added relations imply a.
- *Gelfond-Lifschitz transform* of a KB given a consistent set of ground literals, I:

  1. Delete all rules *r* such that $I \models_L a$, and $a \in B^-(r)$

  2. Delete all negated literals from rules.

  The resulting program is negation free and has a minimal model.
- *I* is a *strong answer set* of a KB if it is a minimal model of KB^I

Nice properties of DL-programs

- DL-programs without negation and ∩ has a least model semantics, similar to prolog.
- Stratified programs have a *unique* least model
- Good computational complexity.

Other systems

- HEX-Programs:

  Generalization of DL-programs.

  Some higher order.

  Non-monotonic. (If a is true in P then a might be false in $P \cup \{R\}$.

  Allows disjunctive rules:

  $a(X) \vee b(Y) \leftarrow c(X,Y)$.

Summary

- Combining various forms of logic programming with ontologies is a useful and powerful technique.
- There are many different ways to do it, each with different strengths and problems.
- DL-programs is one useful model with many nice properties, but not the only one.
- RIF aims to become a standard for describing rules.

References

- *Hybrid reasoning with Rules and Ontologies. - W Drabent, T Eiter, G Ianni T Krennwallner, T Lukasiewicz J Mauszynski*