

OWL, DL and rules

OWL and Rules

- Rule based systems are an important and useful way to represent and reason with knowledge
- Adding rules to OWL has proved to be fraught with problems
- We'll look at the underlying issues and several approaches
 - SWRL: failed standard that has become widely used
 - RIF: a successful standard that's not yet widely used

Semantic Web and Logic

- The Semantic Web is grounded in logic
- But what logic?
 - OWL Full = Classical first order logic (FOL)
 - OWL-DL = Description logic
 - N3 rules \sim = logic programming (LP) rules
 - SWRL \sim = DL + LP
 - Other choices are possible, e.g., default logic, fuzzy logic, probabilistic logics, ...
- How do these fit together and what are the consequences

We need both structure and rules

- **OWL's ontologies** based on DL (and thus in FOL)
 - The Web is an open environment
 - Reusability / interoperability
 - An ontology is a model easy to understand
- Many **rule systems** based on logic programming
 - To achieve decidability, ontology languages don't offer the expressiveness we want. Rules do it well
 - Efficient reasoning support already exists
 - Rules are well-known and often more intuitive

Description Logics vs. Horn Logic

- Neither is a subset of the other
- Impossible in OWL DL: people who study & live in same city are local students,
- Easily done with a a rule
 $\text{studiesAt}(X,U), \text{loc}(U,L), \text{lives}(X,L) \rightarrow \text{localStud}(X)$
- Impossible in horn rules: every person is either a man or a woman
- Easily done in OWL DL:
`:Person owl:disjointUnionOf (:Man :Woman).`

What's Horn clause logic

- Prolog and most 'logic'-oriented rule languages use horn clause logic
 - Defined by UCLA mathematician Alfred Horn
- Horn clauses: a subset of FOL where every sentence is a disjunction of literals (atoms) where at most one is positive
$$\sim P \vee \sim Q \vee \sim R \vee S$$
$$\sim P \vee \sim Q \vee \sim R$$
- Atoms: propositional variables (isRaining) or predicates (married(alice, ?x))

Alternate formulation as implications

- Horn clauses can be re-written using the implication operator

$$\sim P \vee Q = P \rightarrow Q$$

$$\sim P \vee \sim Q \vee R = P \wedge Q \rightarrow R$$

$$\sim P \vee \sim Q = P \wedge Q \rightarrow$$

- What we end up with is \sim “pure prolog”
 - Single positive atom as the rule conclusion
 - Conjunction of positive atoms as the rule antecedents (conditions)
 - No **not** operator
 - Atoms can be predicates (e.g., mother(X,Y))

Prolog's syntax

- Prolog syntax is a bit different, putting the rule's conclusion first

hasMother(?x, ?m) :- hasParent(?x, ?m), female(?m) .
head = conclusion body = conjunction of conditions

- A fact is a rule w/o a body (i.e., no conditions)

hasParent(john, tom).

hasParent(john, mary).

female(mary).

- Prolog 'proves' queries by matching a fact, or a rule's conclusion and then proving each condition in the rule's body

We can relax this a bit

- Head can contain a conjunction of atoms
 - $P \wedge Q \leftarrow R$ is equivalent to $P \leftarrow R$ and $Q \leftarrow R$
- Body can have disjunctions
 - $P \leftarrow R \vee Q$ is equivalent to $P \leftarrow R$ and $P \leftarrow Q$
- But somethings are just not allowed:
 - No disjunction in head, e.g.,
 $\text{man}(?x) ; \text{woman}(?x) :- \text{person}(x)$
 - No logical negation operator, i.e. NOT
 $\text{man}(?x) :- \text{person}(x), \text{not}(\text{woman}(x))$

Where are the quantifiers?

- Quantifiers (forall, exists) are implicit
 - Variables in *rule head* are universally quantified
 - Variables only *in rule body* existentially quantified
- Example:
 - $\text{IsParent}(?x) \text{ :- hasChild}(?x, ?y).$
 - $\text{isParent}(X) \leftarrow \text{hasChild}(X,Y)$
 - $\text{forAll } X: \text{isParent}(X) \text{ if Exisits } Y: \text{hasChild}(X,Y)$

Facts & rule conclusions are definite

- Definite means *not a disjunction*
- Facts are rule with the trivial true condition
- Consider these true facts:
 - $P \vee Q$ # either P or Q (or both) are true
 - $P \rightarrow R$ # if P is true, then R is true
 - $Q \rightarrow R$ # if Q is true, then R is true
- What can you conclude?
- Can this be expressed in horn logic?

Facts & rule conclusions are definite

- Consider these true facts where *not* is classical negation rather than “negation as failure”

$\text{not}(P) \rightarrow Q, \text{not}(Q) \rightarrow P$ # i.e. $P \vee Q$

$P \rightarrow R, Q \rightarrow R$

- Horn clause reasoners can't prove that either P or Q is necessarily true or false so can't show that R must be true

The Programming in Prolog

- Prolog = PROgramming in LOGic
- Prolog's procedural elements make it very useful, if used in moderation
- One is its unprovable operator, $\backslash+$
- $\backslash+$ P succeeds if and only P cannot be proven
- Often called "*negation as failure*"
- Example: assume a person is unmarried if we don't know they are married
Unmarried(?x) :- person(?x), $\backslash+$ married(?x) .

Non-ground entailment

- The LP-semantics defined in terms of minimal Herbrand model, i.e., sets of ground facts
- Because of this, Horn clause reasoners can not derive rules, so that can not do general subsumption reasoning
 - i.e., It can only reason about atomic facts to infer new facts
 - It can't reason about rules and complex facts to create new rules

Decidability

- The largest obstacle!

 - Tradeoff between expressiveness and decidability

- Facing decidability issues from

 - In **LP**: Finiteness of the domain

 - In **classical logic** (and thus in DL): combination of constructs

- **Problem:**

 - Combination of “simple” DLs and Horn Logic are undecidable. (Levy & Rousset, 1998)

SWRL: Semantic Web Rule Language

- SWRL is the **union** of DL and horn logic + many built-in functions (e.g., for math)
- Submitted to W3C in 2004, but failed to become a recommendation (led to RIF)
- Problem: full SWRL specification leads to undecidability in reasoning
- SWRL is well specified and subsets are widely supported (e.g., in Pellet, Hermit)
- Based on OWL: rules use terms for OWL concepts (classes, properties, individuals, literals...)

SWRL

- OWL classes are unary predicates, properties are binary ones
 - Person(?p) ^ sibling(?p,?s) ^ Man(?s) → brother(?p,?s)
- As in Prolog, builtins can be booleans or do a computation and unify the result to a variable
 - swrlb:greaterThan(?age2, ?age1) # age2>age1
 - swrlb:subtract(?n1,?n2,?diff) # diff=n1-n2
- SWRL predicates for OWL axioms and data tests
 - differentFrom(?x, ?y), sameAs(?x, ?y), xsd:int(?x), [3, 4, 5](?x), ...

SWRL Built-Ins

- SWRL defines a set of built-in predicate that allow for comparisons, math evaluation, string operations and more
- See [here](#) for the complete list
- Examples
 - Person(?p), hasAge(?p, ?age), **swrlb:greaterThan**(?age, 18) -> Adult(?p)
 - Person(?p), bornOnDate(?p, ?date), **xsd:date**(?date), **swrlb:date**(?date, ?year, ?month, ?day, ?timezone) -> bornInYear(?p, ?year)
- Some reasoners (e.g., Pellet) allow you to define new built-ins in Java

Drawbacks of full SWRL

- Main *source of complexity*:
 - arbitrary OWL expressions (e.g. restrictions) can appear in the head or body of a rule
- Adds significant expressive power to OWL, but causes *undecidability*
 - there is no inference engine that draws exactly the same conclusions as the SWRL semantics

SWRL Sublanguages

- Challenge: identify sublanguages of SWRL with right balance between expressivity and computational viability
- A candidate OWL DL + *DL-safe rules*
 - every variable must appear in a non-description logic atom in the rule body

DL-safe rules

- Standard reasoners support only DL-safe rules
 - Rule variables bind only to known individuals (i.e., owl:NamedIndividual)
- Example
 - $:Vehicle(?v) \wedge :Motor(?m) \wedge :hasMotor(?v,?m) \rightarrow :MotorVehicle(?v)$
- Where
 - $:Car = :Vehicle \text{ and some } hasMotor \text{ Motor}$
 - $:x \text{ a } :Car$
- Reasoner won't bind ?m to a motor since it is not a known individual
- Thus the rule cannot conclude $MotorVehicle(:x)$

Protégé 5 had SWRLTab

Add/edit rules and optionally run a separate rules engine

The screenshot shows the Protégé 5 SWRLTab interface. The browser address bar displays the URL `peeps (http://ebiq.org/ontologies/peeps/)`. The main window title is `peeps (http://ebiq.org/ontologies/peeps/)`. The interface includes a search bar and a navigation menu with tabs for `Active Ontology`, `Entities`, `Object Properties`, `Data Properties`, `Individuals by class`, and `SWRLTab`.

Name	Rule
<input checked="" type="checkbox"/> S1	<code>peeps:hasAge(?p1, ?a1) ^ peeps:hasAge(?p2, ?a2) ^ swrlb:lessThan(?a1, ?a2) -> youngerThan(?p1, ?p2)</code>
<input checked="" type="checkbox"/> S2	<code>peeps:Woman(?p2) ^ peeps:hasParent(?p1, ?p2) -> hasMother(?p1, ?p2)</code>

Buttons for `New`, `Edit`, `Clone`, and `Delete` are located below the table.

The `Control` tab is active, showing instructions for using the Drools rule engine:

Using the Drools rule engine.

Press the 'OWL+SWRL->Drools' button to transfer SWRL rules and relevant OWL knowledge to the rule engine.
Press the 'Run Drools' button to run the rule engine.
Press the 'Drools->OWL' button to transfer the inferred rule engine knowledge to OWL knowledge.

The SWRLAPI supports an OWL profile called OWL 2 RL and uses an OWL 2 RL-based reasoner to perform reasoning.
See the 'OWL 2 RL' sub-tab for more information on this reasoner.

Buttons for `OWL+SWRL->D...`, `Run Drools`, and `Drools->OWL` are located at the bottom of the interface.

To use the reasoner click Reasoner > Start reasoner Show Inferences

SWRL limitations

SWRL rules do not support many useful features of some rule-based systems

- Default reasoning
- Rule priorities
- Negation as failure (e.g., for closed-world semantics)
- Data structures
- ...

Limitations led to [RIF](#), Rule Interchange Format

Summary

- Horn logic is a subset of predicate logic that allows efficient reasoning, orthogonal to description logics
- Horn logic is the basis of monotonic rules
- DLP and SWRL are two important ways of combining OWL with Horn rules.
 - DLP is essentially the intersection of OWL and Horn logic
 - SWRL is a much richer language

Summary (2)

- Nonmonotonic rules are useful in situations where the available information is incomplete
- They are rules that may be overridden by contrary evidence
- Priorities are sometimes used to resolve some conflicts between rules