OWL, DL and rules

Based on slides from Grigoris Antoniou, Frank van Harmele and Vassilis Papataxiarhis

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 ~= logic programming (LP) rules
  - SWRL ~= 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 on 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 and live in same city are local students
- Easily done with a a rule

  studiesAt(X,U), loc(U,L), lives(X,L) $\rightarrow$ 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: subset of FOL where every sentence is a disjunction of atoms where at most one is positive

  ~P V ~Q V R

  ~P V ~Q

  R
- Atoms: propositional variables (isMarried) or predicates (e.g., person(alice), mother(alice, ?x)) that can have variables

Alternate formulation as implications

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

  ~P V ~Q V R ⇔ P ∧ Q ➔R (R true of both P and Q true)

  ~P V ~Q ⇔ P ∧ Q ➔⊥ (contradiction if both P and Q true))

  R ⇔ ➔ R (R is true)

- What we end up with is ~ "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., person(X), mother(X,Y), between(City, newYork, baltimore)

Prolog's synax

- 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 $\land$ Q $\leftarrow$ R is equivalent to P$\leftarrow$R and Q$\leftarrow$R
- Body can have disjunctions
  - P$\leftarrow$R$\lor$Q is equivalent to P$\leftarrow$R and P$\leftarrow$Q
- But somethings are just not allowed:
  - No disjunction in head, e.g.,

    man(?x) ; woman(?x) :- person(x)
  - No logical negation operator, i.e. NOT

    man(?x) :- person(x), not(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:
  - IsParent(?x) :- hasChild(?x, ?y).
  - isParent(X) ← hasChild(X,Y)
  - forAll X: isParent(X) if Exisits Y: hasChild(X,Y)
    ∀X isParent(X) → ∃Y 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:

  $P \lor Q$ # 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"

  not(P) ➔ Q, not(Q) ➔P # i.e. P⋁Q

  P ➔ R, Q ➔ 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, when used in moderation
- One element is it's unprovable operator, \+
- \+ 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), \+ married(?x) .

Non-ground entailment (1)

- The LP-semantics defined in terms of minimal Herbrand model, i.e., sets of ground facts
- Because of this, LP 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

Non-ground entailment (2)

- A horn-clause reasoner can't do the following
- Given

  animal(?X) ∧ disease(?D) ∧ has(?X,?D) → sickAnimal(?x)

  dog(?X) → animal(?X)

  disease(rabies)

- Derive a new rule

  dog(?X), has(?X, rabies) → sickAnimal(?X)

- Even though it follows from the underlying logic

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

  sibling(?p,?s) ^ Man(?s) → brother(?p,?s)

- As in Prolog, bulitins 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., owl2 owl:NamedIndividual)

- Example

  :Vehicle(?v) ^ :Motor(?m) ^ :hasMotor(?v,?m) -> :MotorVehicle(?v)

- Where

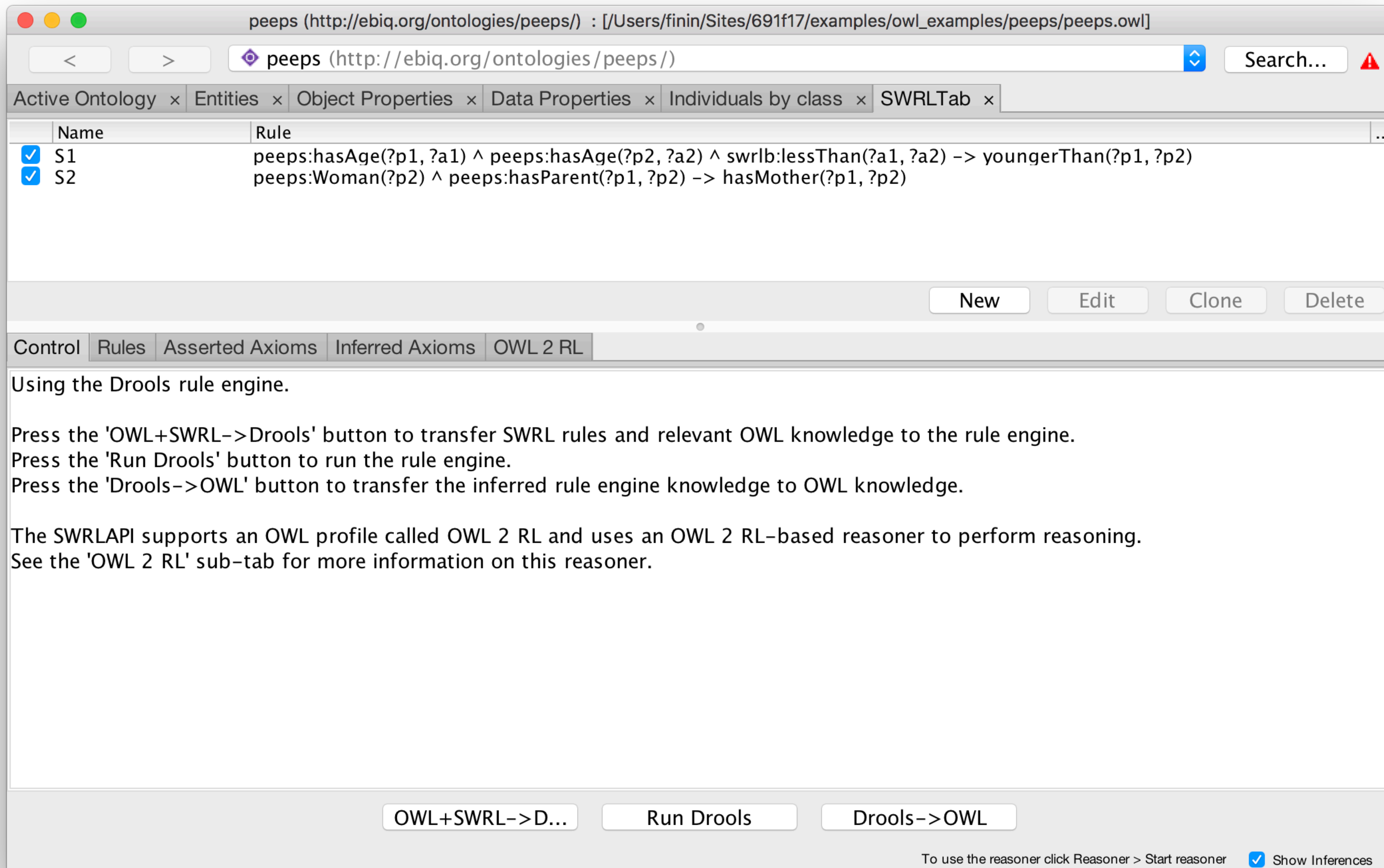
  :Car = :Vehicle and some :hasMotor Motor

  :x 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



SWRL limitations

SWRL rules do not support many useful features of 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