# OWL, DL and Rules

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

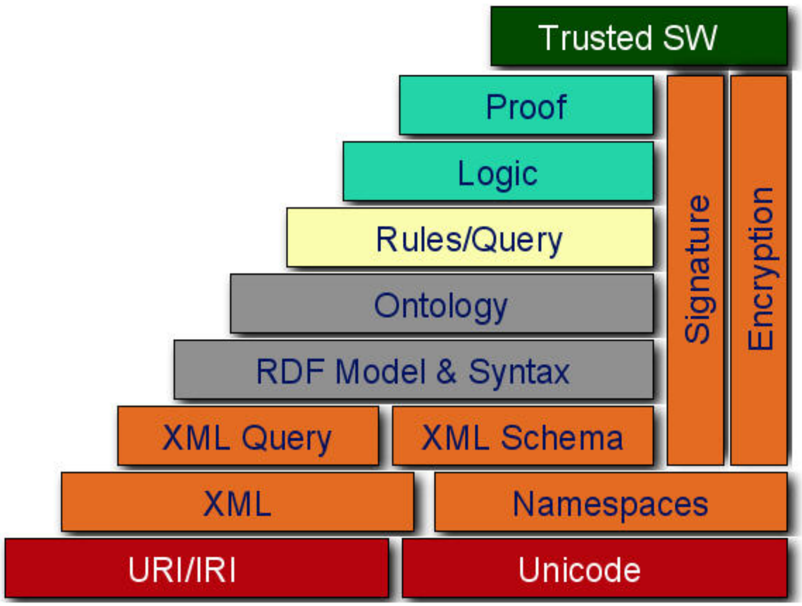
# Semantic Web and Logic

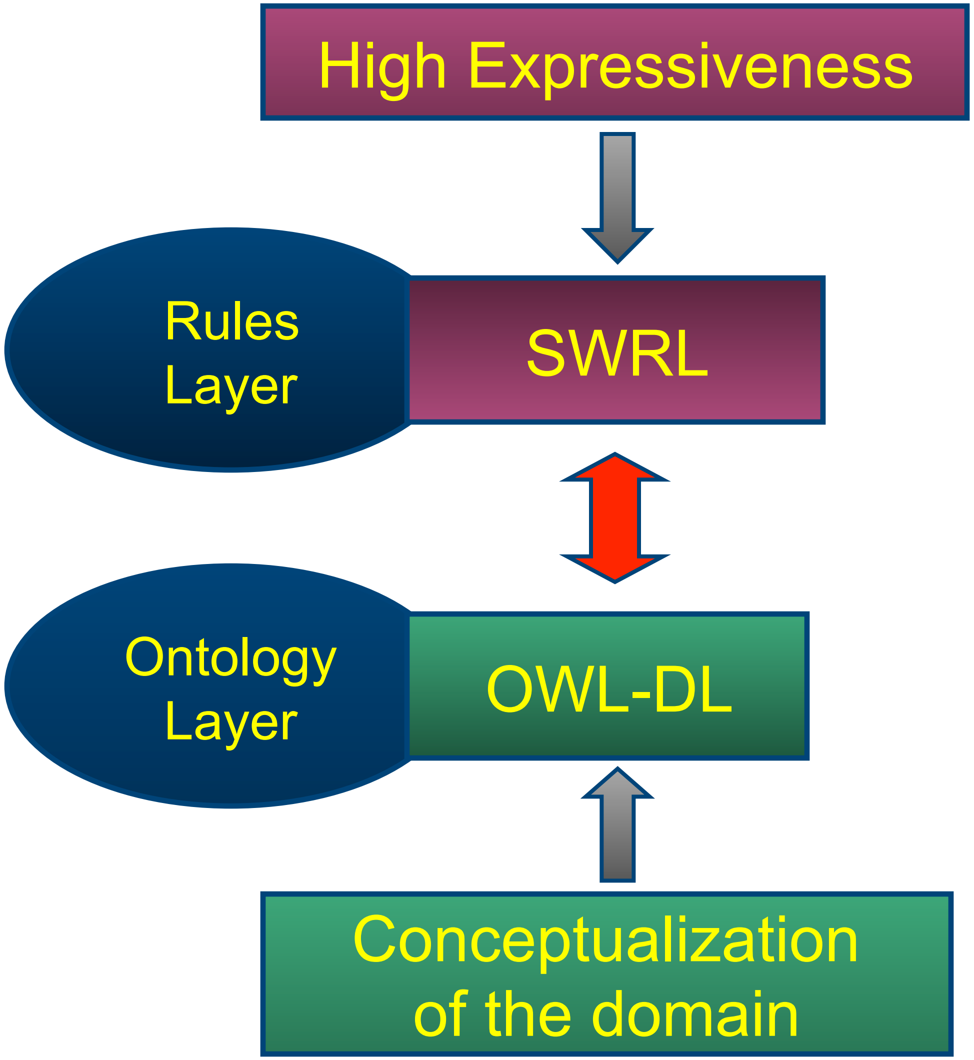
- The Semantic Web is grounded in logic
- But what logic?
  - OWL Full = Classical first order logic (FOL)
  - OWL-DL = Description logic
  - N3 rules ~= logic programming (LP) rules
  - SWRL ~= DL + LP
  - Other choices are possible, e.g., default logic, Markov logic, …
- How do these fit together?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

# We need both structure and rules

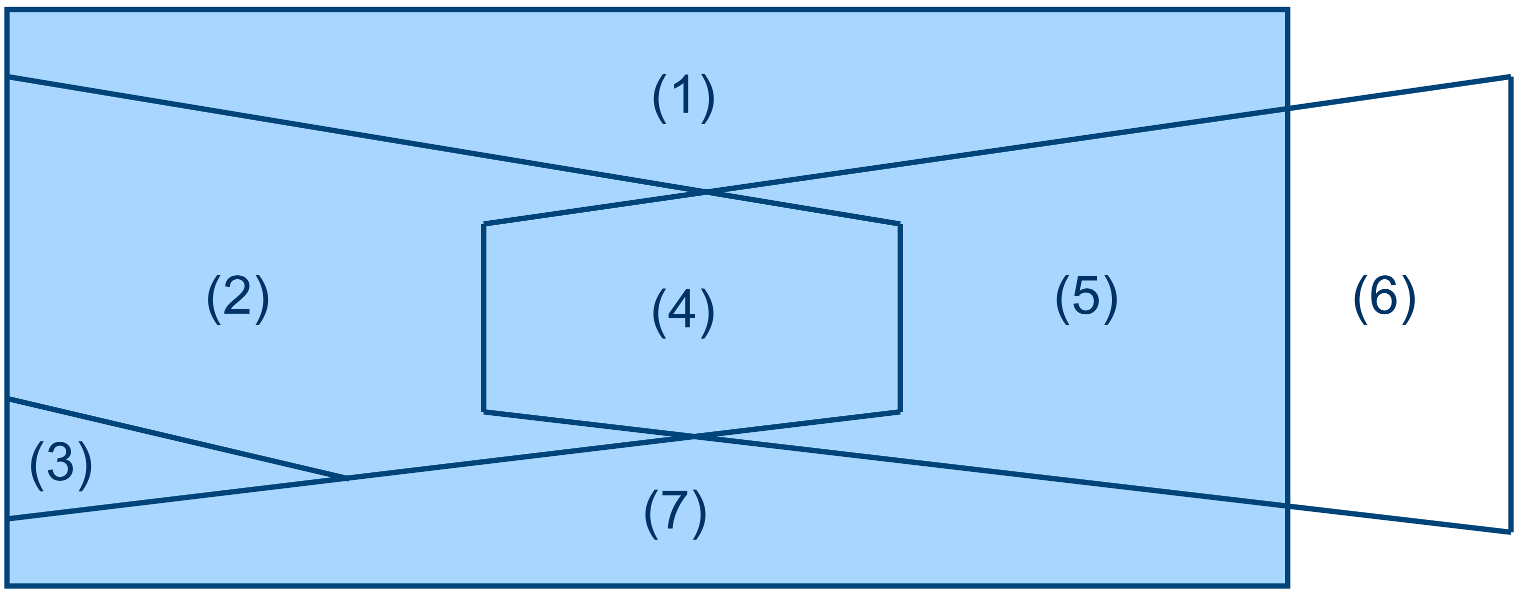
- **OWL's ontologies** are based on Description Logics (and thus in FOL)
  - ✓ The Web is an open environment.
  - ✓ Reusability / interoperability.
  - ✓ An ontology is a model easy to understand.
- Many **rule systems** based on logic programming
  - ✓ For the sake of decidability, ontology languages don't offer the expressiveness we want (e.g. constructor for composite properties?). Rules do it well.
  - ✓ Efficient reasoning support already exists.
  - ✓ Rules are well-known in practice.

# A common approach



# LP and classical logic overlap




| | |
|---|---|
| FOL: (All except (6)), | (2)+(3)+(4): DLs |
| (4): Description Logic Programs (DLP), | (3): Classical Negation |
| (4)+(5): Horn Logic Programs, | (4)+(5)+(6): LP |
| (6): Non-monotonic features (like NAF, etc.) | (7): ^head and, $\vee$body |

# Description Logics vs. Horn Logic

- Neither of them is a subset of the other
- It is impossible to assert that persons who study and live in the same city are “home students” in OWL
  - This can be done easily using rules:

    **studies(X,Y), lives(X,Z), loc(Y,U), loc(Z,U) → homeStudent(X)**
- Rules cannot assert the information that a person is either a man or a woman
  - This information is easily expressed in OWL using disjoint union

# Basic Difficulties

Classical Logic vs. Logic Programming

- Monotonic vs. Non-monotonic Features
  - Open-world vs. Closed-world assumption
  - Negation-as-failure vs. classical negation
- Non-ground entailment
- Strong negation vs. classical negation
- Equality
- Decidability

# What’s Horn clause logic

- Prolog and most ‘logic’-oriented rule languages use horn clause logic
  - Cf. UCLA mathematician Alfred Horn
- Horn clauses are a subset of FOL where every sentence is a disjunction of literals (atoms) where at most one is positive

  ~P V ~Q V ~R V S

  ~P V ~Q V ~R

# An alternate formulation

- 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 ~ “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))

# Where are the quantifiers?

- Quantifiers (forall, exists) are implicit
  - Variables in head are universally quantified
  - Variables only in body are existentially quantified
- Example:
  - isParent(X) $\leftarrow$ hasChild(X,Y)
  - forAll X: isParent(X) if Exisits Y: hasChild(X,Y)

# 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 something are just not allowed:
  - No disjunction in head
  - No negation operator, i.e. NOT

# Facts & rule conclusions are definite

- A fact is just a rule with the trivial true condition
- Consider these true facts:
  - P V Q
  - P → R
  - Q → R
- What can you conclude?
- Can this be expressed in horn logic?

# Facts & rule conclusions are definite

- Consider these true facts:
  - not(P) ➔ Q, not(Q) ➔P
  - P ➔ R
  - Q ➔ R
- A horn clause reasoner (e.g., Prolog) will be unable to prove that either P or Q is necessarily true or false
- And can not show that R must be true

# Open- vs. closed-world assumption

- Logic Programming – CWA
  - If KB $\not\models a$, then KB = KB $\cup \neg a$
- Classical Logic – OWA
  - It keeps the world open.
  - KB:

    Man $\sqsubseteq$ Person, Woman $\sqsubseteq$ Person

    Bob $\in$ Man, Mary $\in$ Woman

  Query: "find all individuals that are not women"

# Non-ground entailment

- The LP-semantics is 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

# Decidability

- The largest obstacle!
  - Tradeoff between expressiveness and decidability.
- Facing decidability issues from 2 different angles
  - In LP: Finiteness of the domain
  - In classical logic (and thus in DL ): Combination of constructs
- Problem:

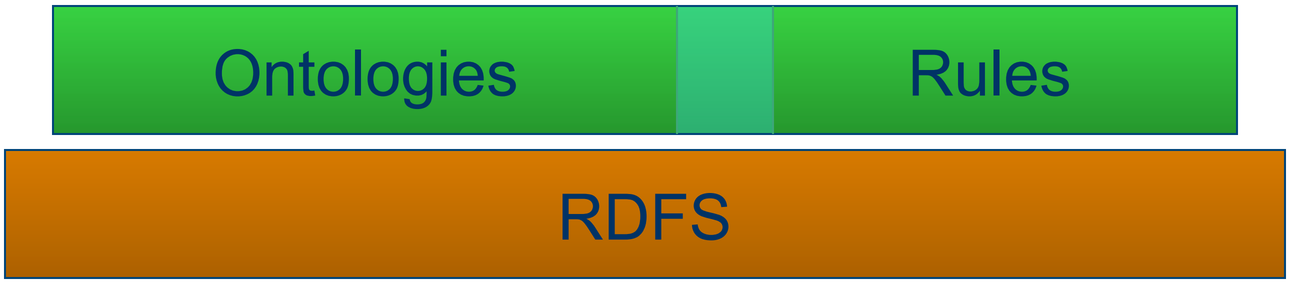
  Combination of “simple” DLs and Horn Logic are undecidable. (Levy & Rousset, 1998)

# Rules + Ontologies

- Still a challenging task!
- A number of different approaches exists: SWRL, DLP (Grosof), dl-programs (Eiter), DL-safe rules, Conceptual Logic Programs (CLP), AL-Log, DL +log
- Two main strategies:
  - Tight Semantic Integration (Homogeneous Approaches)
  - Strict Semantic Separation (Hybrid Approaches)

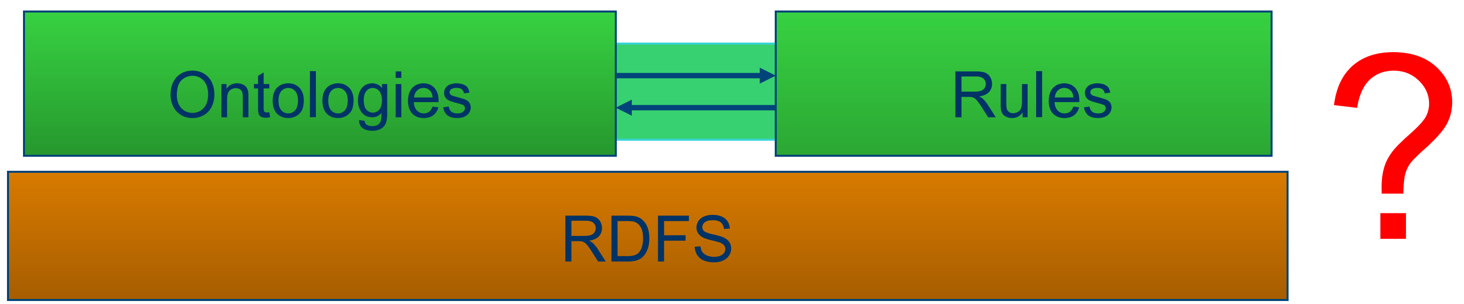
# Homogeneous Approach

- Interaction with tight semantic integration.
- Both ontologies and rules are embedding in a common logical language.
- No distinction between rule predicates and ontology predicates.
- Rules may be used for defining classes and properties of the ontology.
- Example: SWRL, DLP



# Hybrid Approach

- Integration with strict semantic separation between the two layers.
- Ontology is used as a conceptualization of the domain.
- Rules cannot define classes and properties of the ontology, but some application-specific relations.
- Communication via a “safe interface”.
- Example: Answer Set Programming (ASP)



# The Essence of DLP

- Simplest approach for combining DLs with Horn logic: their intersection
  - the Horn-definable part of OWL, or equivalently
  - the OWL-definable part of Horn logic

# Advantages of DLP

- **Modeling**: Freedom to use either OWL or rules (and associated tools and methodologies)
- **Implementation**: use either description logic reasoners or deductive rule systems
  - extra flexibility, interoperability with a variety of tools
- **Expressivity**: existing OWL ontologies frequently use very few constructs outside DLP

# RDFS and Horn Logic

| | |
|---|---|
| Statement(a,P,b) | $P(a,b)$ |
| type(a,C) | $C(a)$ |
| C subClassOf D | $C(X) \rightarrow D(X)$ |
| P subPorpertyOf Q | $P(X,Y) \rightarrow Q(X,Y)$ |
| domain(P,C) | $P(X,Y) \rightarrow C(X)$ |
| range(P,C) | $P(X,Y) \rightarrow C(Y)$ |

# OWL in Horn Logic

| | |
|---|---|
| C sameClassAs D | $C(X) \rightarrow D(X)$ |
| | $D(X) \rightarrow C(X)$ |
| P samePropertyAs Q | $P(X,Y) \rightarrow Q(X,Y)$ |
| | $Q(X,Y) \rightarrow P(X,Y)$ |

# OWL in Horn Logic (2)

| | |
|---|---|
| transitiveProperty(P) | $P(X,Y), P(Y,Z) \rightarrow P(X,Z)$ |
| inverseProperty(P,Q) | $Q(X,Y) \rightarrow P(Y,X)$ |
| | $P(X,Y) \rightarrow Q(Y,X)$ |
| functionalProperty(P) | $P(X,Y), P(X,Z) \rightarrow Y=Z$ |

# OWL in Horn Logic (3)

$(C1 \cap C2)$ subClassOf D

$$C1(X), C2(X) \rightarrow D(X)$$

C subClassOf $(D1 \cap D2)$

$$C(X) \rightarrow D1(X)$$

$$C(X) \rightarrow D2(X)$$

# OWL in Horn Logic (4)

$(C1 \cup C2)$ subClassOf D

$$C1(X) \rightarrow D(X)$$

$$C2(X) \rightarrow D(X)$$

C subClassOf $(D1 \cup D2)$

Translation not possible!

# OWL in Horn Logic (5)

C subClassOf AllValuesFrom(P,D)

$$C(X), P(X,Y) \rightarrow D(Y)$$

AllValuesFrom(P,D) subClassOf C

Translation not possible!

# OWL in Horn Logic (6)

C subClassOf SomeValuesFrom(P,D)

Translation not possible!

SomeValuesFrom(P,D) subClassOf C

$D(X), P(X,Y) \rightarrow C(Y)$

# OWL in Horn Logic (7)

- MinCardinality cannot be translated due to existential quantification
- MaxCardinality 1 may be translated if equality is allowed
- Complement cannot be translated, in general

# The Essence of SWRL

- Combines OWL DL (and thus OWL Lite) with function-free Horn logic.
- Thus it allows Horn-like rules to be combined with OWL DL ontologies.

# Rules in SWRL

**B1, . . . , Bn → A1, . . . , Am**

**A1, . . . , Am, B1, . . . , Bn** have one of the forms:

- C(x)
- P(x,y)
- sameAs(x,y) differentFrom(x,y)

where C is an OWL description, P is an OWL property, and x,y are variables, OWL individuals or OWL data values.

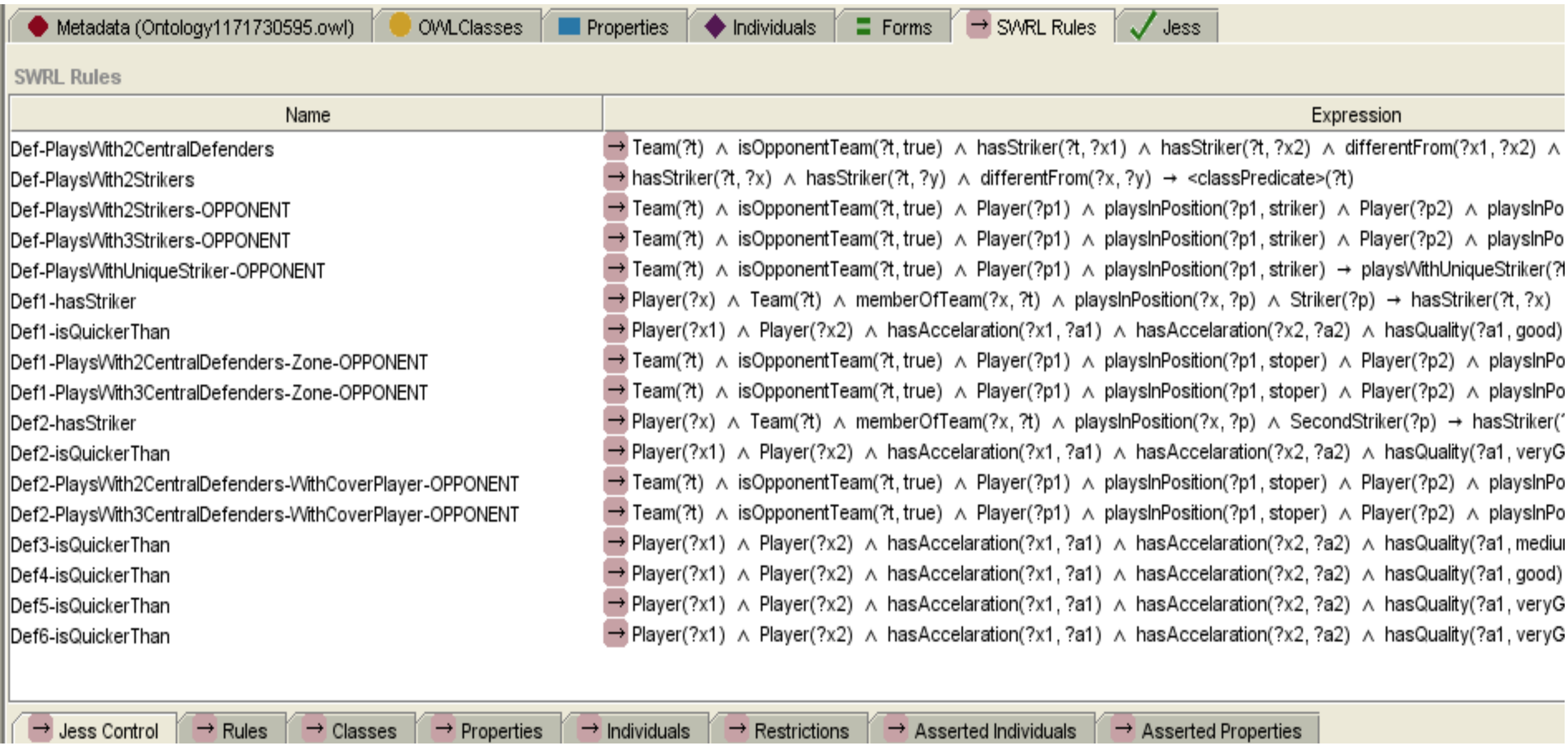
# Drawbacks of SWRL

- Main source of complexity:
  - arbitrary OWL expressions, such as 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

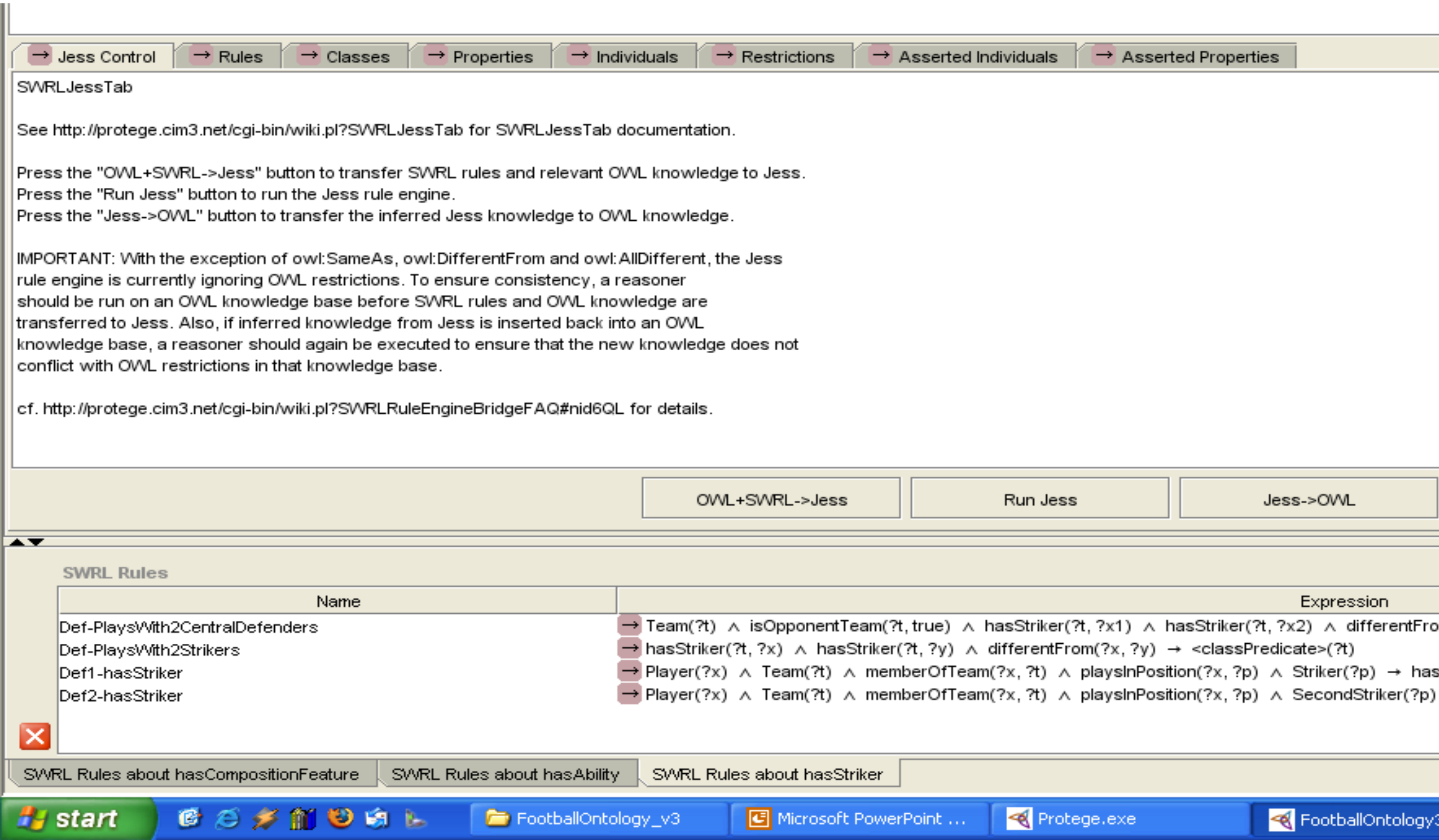
- SWRL adds the expressivity of DLs and function-free rules.
- One challenge: identify sublanguages of SWRL with right balance between expressivity and computational viability.
- A candidate OWL DL + DL-safe rules
  - every variable must appear in a non-description logic atom in the rule body.

# Protégé SWRL-Tab




| Name | Expression |
|---|---|
| Def-PlaysWith2CentralDefenders | Team(?t) ∧ isOpponentTeam(?t, true) ∧ hasStriker(?t, ?x1) ∧ hasStriker(?t, ?x2) ∧ differentFrom(?x1, ?x2) ∧ |
| Def-PlaysWith2Strikers | hasStriker(?t, ?x) ∧ hasStriker(?t, ?y) ∧ differentFrom(?x, ?y) → <classPredicate>(?t) |
| Def-PlaysWith2Strikers-OPPONENT | Team(?t) ∧ isOpponentTeam(?t, true) ∧ Player(?p1) ∧ playsInPosition(?p1, striker) ∧ Player(?p2) ∧ playsInPo |
| Def-PlaysWith3Strikers-OPPONENT | Team(?t) ∧ isOpponentTeam(?t, true) ∧ Player(?p1) ∧ playsInPosition(?p1, striker) ∧ Player(?p2) ∧ playsInPo |
| Def-PlaysWithUniqueStriker-OPPONENT | Team(?t) ∧ isOpponentTeam(?t, true) ∧ Player(?p1) ∧ playsInPosition(?p1, striker) → playsWithUniqueStriker(?t |
| Def1-hasStriker | Player(?x) ∧ Team(?t) ∧ memberOfTeam(?x, ?t) ∧ playsInPosition(?x, ?p) ∧ Striker(?p) → hasStriker(?t, ?x) |
| Def1-isQuickerThan | Player(?x1) ∧ Player(?x2) ∧ hasAccelaration(?x1, ?a1) ∧ hasAccelaration(?x2, ?a2) ∧ hasQuality(?a1, good) |
| Def1-PlaysWith2CentralDefenders-Zone-OPPONENT | Team(?t) ∧ isOpponentTeam(?t, true) ∧ Player(?p1) ∧ playsInPosition(?p1, stoper) ∧ Player(?p2) ∧ playsInPo |
| Def1-PlaysWith3CentralDefenders-Zone-OPPONENT | Team(?t) ∧ isOpponentTeam(?t, true) ∧ Player(?p1) ∧ playsInPosition(?p1, stoper) ∧ Player(?p2) ∧ playsInPo |
| Def2-hasStriker | Player(?x) ∧ Team(?t) ∧ memberOfTeam(?x, ?t) ∧ playsInPosition(?x, ?p) ∧ SecondStriker(?p) → hasStriker( |
| Def2-isQuickerThan | Player(?x1) ∧ Player(?x2) ∧ hasAccelaration(?x1, ?a1) ∧ hasAccelaration(?x2, ?a2) ∧ hasQuality(?a1, veryG |
| Def2-PlaysWith2CentralDefenders-WithCoverPlayer-OPPONENT | Team(?t) ∧ isOpponentTeam(?t, true) ∧ Player(?p1) ∧ playsInPosition(?p1, stoper) ∧ Player(?p2) ∧ playsInPo |
| Def2-PlaysWith3CentralDefenders-WithCoverPlayer-OPPONENT | Team(?t) ∧ isOpponentTeam(?t, true) ∧ Player(?p1) ∧ playsInPosition(?p1, stoper) ∧ Player(?p2) ∧ playsInPo |
| Def3-isQuickerThan | Player(?x1) ∧ Player(?x2) ∧ hasAccelaration(?x1, ?a1) ∧ hasAccelaration(?x2, ?a2) ∧ hasQuality(?a1, mediu |
| Def4-isQuickerThan | Player(?x1) ∧ Player(?x2) ∧ hasAccelaration(?x1, ?a1) ∧ hasAccelaration(?x2, ?a2) ∧ hasQuality(?a1, good) |
| Def5-isQuickerThan | Player(?x1) ∧ Player(?x2) ∧ hasAccelaration(?x1, ?a1) ∧ hasAccelaration(?x2, ?a2) ∧ hasQuality(?a1, veryG |
| Def6-isQuickerThan | Player(?x1) ∧ Player(?x2) ∧ hasAccelaration(?x1, ?a1) ∧ hasAccelaration(?x2, ?a2) ∧ hasQuality(?a1, veryG |

# Protégé SWRL-Tab



# Non-monotonic rules

- Non-monotonic rules exploit an "unprovable" operator
- This can be used to implement default reasoning, e.g.,
  - assume P(X) is true for some X unless you can prove hat it is not
  - Assume that a bird can fly unless you know it can not

# monotonic

```
canFly(X) :- bird (X)
bird(X) :- eagle(X)
bird(X) :- penguin(X)
eagle(sam)
penguin(tux)
```

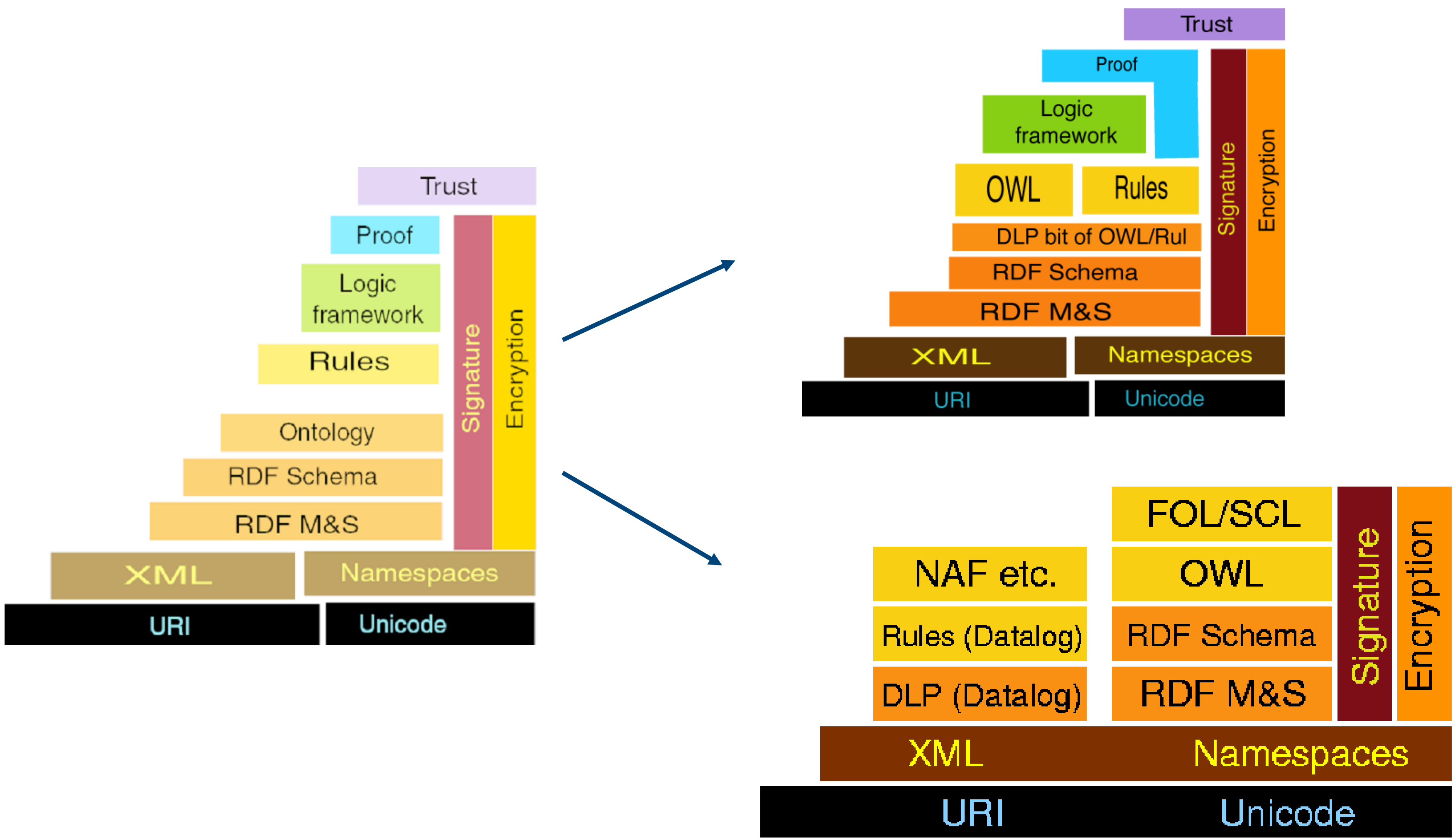
# Non-monotonic

```
canFly(X) :- bird (X), \+ not(canFly(X))
bird(X) :- eagle(X)
bird(X) :- penguin(X)
not(canFly(X)) :- penguin(X)
eagle(sam)
penguin(tux)
```

# Rule priorities

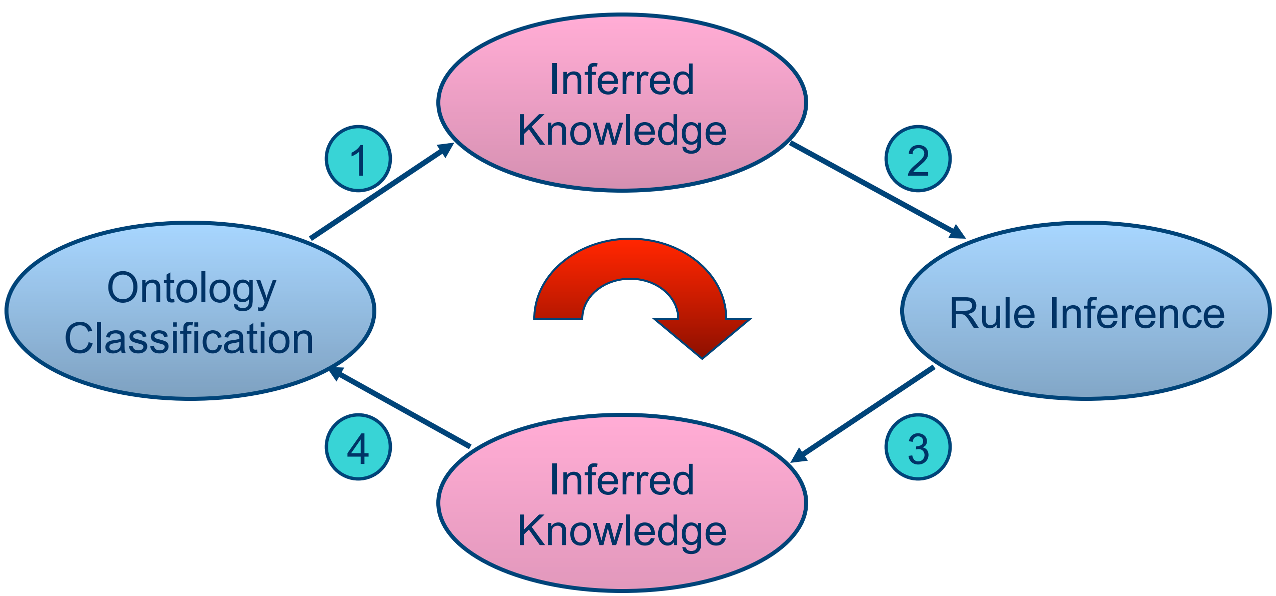
- This approach can be extended to implement systems where rules have priorities
- This seems to be intuitive to people – used in many human systems
  - E.g., University policy overrules Department policy
  - The “Ten Commandments” can not be contravened

# Two Semantic Webs?



# Limitations

- The rule inference support not integrated with OWL classifier.
- New assertions by rules may violate existing restrictions in ontology. New inferred knowledge from classification may in turn produce knowledge useful for rules.



# Limitations

- Existing solution:

  Solve these possible conflicts manually.
- Ideal solution:

  Have a single module for both ontology classification and rule inference.
- What if we want to combine non-monotonic features with classical logic?
  - Partial Solutions:
    - Answer set programming
    - Externally (through the use of appropriate rule engines)

# 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
- Representation XML-like languages is straightforward