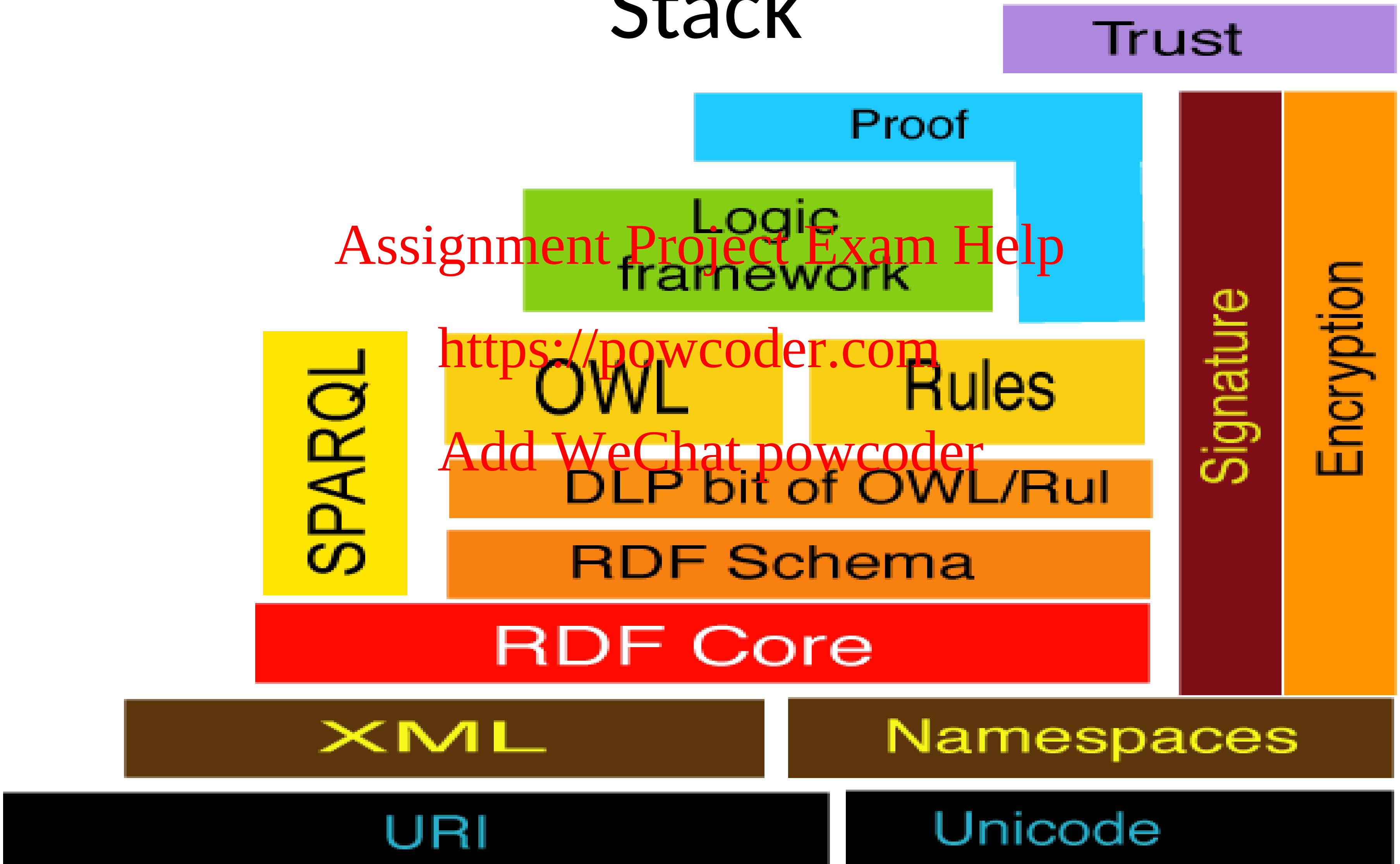
Lecture 11

Logic and Inference: Rules

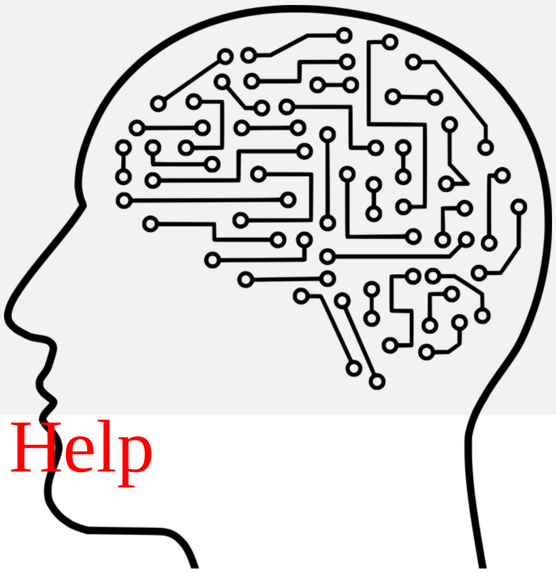
The contents are taken from "A Semantic Web Primer – MIT press"

The slides are prepared by Dr. Davoud Mougouei

A specific Semantic Web Layer Stack

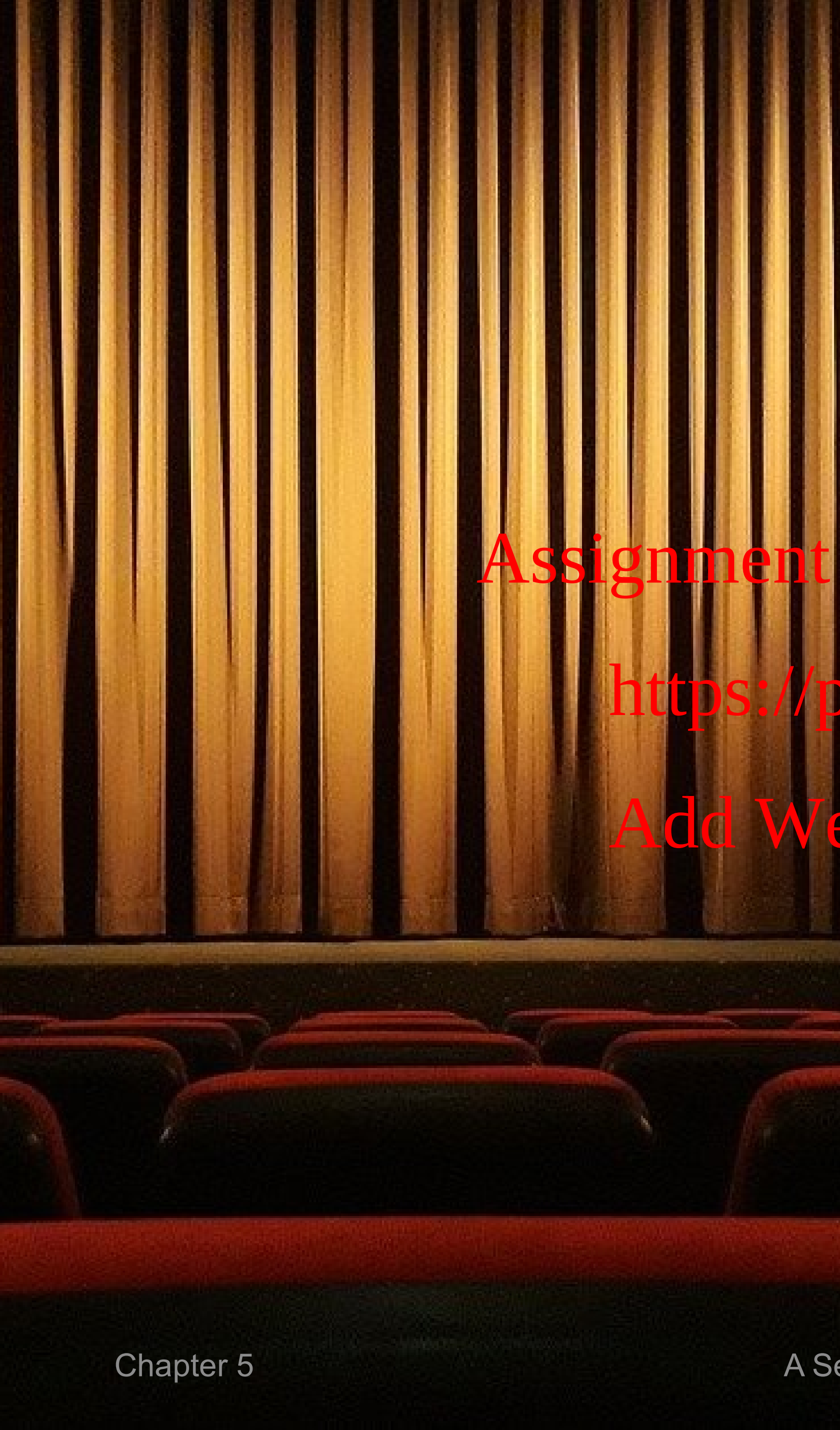


Logic and Rules



- Knowledge representation had been studied long before the emergence of the World Wide Web, in the area of artificial intelligence and, before that, in philosophy.

- The primary original motivation of logic was the study of objective laws of logical consequence.



Logical consequence

https://www.youtube.com/watch?v=SQ0yyhU9G8Y&ab_channel=CursosOn-LineMasivos

Logic and Rules

Popularity and importance of logic

- It provides a high-level language in which knowledge can be expressed in a transparent way.

- It has a high expressive power.

- It has a well-understood formal semantics, which assigns an unambiguous meaning to logical statements.

- There is a precise notion of logical consequence, which determines whether a statement follows semantically from a set of other statements (premises).

Logic and Rules

Popularity and importance of logic

- There exist proof systems that can automatically derive statements syntactically from a set of premises.

- Predicate logic is unique in the sense that sound and complete proof systems do exist. More expressive logics (higher-order logics) do not have such proof systems.

- Because of the existence of proof systems, it is possible to trace the proof that leads to a logical consequence. In this sense, the logic can provide explanations for answers.

Logic and Rules

- The languages of RDF and OWL2 profiles (other than OWL2 Full) can be viewed as specializations of predicate logic.

- One justification for the existence of such specialized languages is that they provide a syntax that fits well with the intended use (web languages based on tags).

- Major justification: they define reasonable subsets of logic. There is a trade-off between the expressive power and the computational complexity of logics: the more expressive the language, the less efficient the corresponding proof systems.

Logic and Rules

- Most OWL variants correspond to a description logic, a subset of predicate logic for which efficient proof systems exist.

- Another subset of predicate logic with efficient proof systems comprises the so-called rule systems (also known as Horn logic or definite logic programs).

Logic and Rules

- A rule has the form A1,...An →B where Ai and B are atomic formulas.

  - As a deductive rule: If A1 , . . . , An are known to be true, then B is also true.

  - As a reactive rule: If the conditions A1, . . . , An are true, then carry out the action B.

  - Both views have important applications. However, we take the deductive approach.

Logic and Rules

- Description logics and Horn logic are orthogonal; neither of them is a subset of the other.

- It is impossible to define the class of happy spouses as those who are married to their best friend in description logics. But this piece of knowledge can easily be represented using rules:

married(X, Y), bestF riend(X, Y) → happySpouse(X)

Logic and Rules

- Rules cannot (in the general case) assert
  - (a) negation/complement of classes;
  - (b) disjunctive/union information (for instance, that a person is either a man or a woman);
  - (c) existential quantification (for instance, that all persons have a father).

- In contrast, OWL is able to express the complement and union of classes and certain forms of existential quantification.

Logic and Rules

Monotonic vs Nonmonotonic Rules

- R1 : If birthday, then special discount.
  R2 : If not birthday, then not special discount.
- R1 : If birthday, then special discount.
  R2′ : If birthday is not known, then not special discount.
- The premise of rule R2′ is not within the expressive power of predicate logic.

Rules on the Semantic Web

Rule Interchange Format (RIF)

- A W3C working group has developed the Rule Interchange Format (RIF) standard.

- While RDF and OWL are meant for directly represent knowledge, RIF was designed primarily for the exchange of rules across different applications.

- Due to the underlying aim of serving as an interchange format among different rule systems, RIF combines many of their features, and is quite complex.

Rules on the Semantic Web

Alternatives to RIF

- Rules over RDF can be expressed in an elegant way using SPARQL constructs; e.g., SPIN

- Those wishing to use rules in the presence of rich semantic structures can use SWRL, which couples OWL DL functionalities with certain types of rules.

- Those who wish to model in terms of OWL but use rule technology for implementation purposes may use OWL2 RL.

Example of Monotonic Rules

Family Relationships

$mother(X,Y)$	X is the mother of Y
$father(X,Y)$	X is the father of Y
$male(X)$	X is male
$female(X)$	X is female

Example of Monotonic Rules

Family Relationships

$$mother(X, Y) \rightarrow parent(X, Y)$$

$$father(X, Y) \rightarrow parent(X, Y)$$

$$male(X), parent(P, X), parent(P, Y), notSame(X, Y) \rightarrow$$

$$brother(X, Y)$$

$$female(X), parent(P, X), parent(P, Y), notSame(X, Y) \rightarrow$$

$$sister(X, Y)$$

Is there any logical problem with the rules in previous slide?

Example of Monotonic Rules

Family Relationships

$$brother(X,P), parent(P,Y) \rightarrow uncle(X,Y)$$

$$mother(X,P), parent(P,Y) \rightarrow grandmother(X,Y)$$

$$parent(X,Y) \rightarrow ancestor(X,Y)$$

$$ancestor(X,P), parent(P,Y) \rightarrow ancestor(X,Y)$$

Monotonic Rules

Syntax: Rules

- loyal customers with ages over 60 are entitled to a special discount:

$$loyalCustomer(X), age(X) > 60 \rightarrow discount(X)$$

- *variables*, which are placeholders for values: X
- *constants*, which denote fixed values: 60
- *predicates*, which relate objects: loyalCustomer, >
- *function symbols*, which denote a value, when applied to certain arguments: age

Monotonic Rules

Syntax: Rules

$$loyalCustomer(X), age(X) > 60 \rightarrow discount(X)$$

- This rule is applied for any customer: if a customer happens to be loyal and over 60, then she gets the discount.

- In other words, the variable X is implicitly universally quantified (using ∀X).

- In general, all variables occurring in a rule are implicitly universally quantified.

Monotonic Rules

Syntax: Rules

$$B_1, \ldots, B_n \to A$$

$$\forall X_1 \ldots \forall X_k((B_1 \wedge \ldots \wedge B_n) \to A)$$

$$\forall X_1 \ldots \forall X_k(A \vee \neg B_1 \vee \ldots \vee \neg B_n)$$

Monotonic Rules

Syntax: Facts

- A fact is an atomic formula, such as loyalCustomer(a345678), which says that the customer with ID a345678 is loyal.

- The variables of a fact are implicitly universally quantified.

Monotonic Rules

Syntax: Logic Programs

- A logic program P is a finite set of facts and rules. Its predicate logic translation pl (P) is the set of all predicate logic interpretations of rules and facts in P .

Monotonic Rules

Syntax: Goals

- A goal denotes a query G asked to a logic program. It has the form B1,…,Bn →

- If n = 0 we have the empty goal □.

- Interpret goals in predicate logic:

$$\forall X_1 \ldots \forall X_k (\neg B_1 \vee \ldots \vee \neg B_n)$$

$$\neg \exists X_1 \ldots \exists X_k (B_1 \wedge \ldots \wedge B_n)$$

Monotonic Rules

Syntax: Goals

- In logic programming we prove that a goal can be answered positively by negating the goal and proving that we get a contradiction using the logic program.

- For example, given the logic program p(*a*) and the goal ¬∃X p(X) we get a logical contradiction: the second formula says that no element has the property *p*, but the first formula says that the value of *a* does have the property *p*. Thus ∃Xp(X) follows from p(a).

Monotonic Rules

Semantics: Predicate Logic Semantics

- One way of answering a query is to use the predicate logic interpretation of rules, facts, and queries, and to make use of the well-known semantics of predicate logic.

Monotonic Rules

Semantics: Predicate Logic Semantics

- Given a logic program P and a query B1,...,Bn → with the variables X1 , . . . , Xk , we answer positively if, and only if,

$$pl(P) \models \exists X_1 \ldots \exists X_k (B_1 \wedge \ldots \wedge B_n) \qquad (1)$$

$$pl(P) \cup \{\neg \exists X_1 \ldots \exists X_k (B_1 \wedge \ldots \wedge B_n)\} \text{ is unsatisfiable} \qquad (2)$$

Monotonic Rules
Semantics: Predicate Logic Semantics

- In other words, we give a positive answer if the predicate logic representation of the program *P*, together with the predicate logic interpretation of the query, is unsatisfiable (a contradiction).

$$pl(P) \cup \{\neg \exists X_1 \ldots \exists X_k (B_1 \wedge \ldots \wedge B_n)\} \text{ is unsatisfiable} \tag{2}$$

Monotonic Rules

Semantics: Predicate Logic Semantics

- Once the predicate logic interpretation of P is true, ∃X1 . . . ∃Xk (B1 ∧ . . . ∧ Bn) must be true, too.

- That is, there are values for the variables X1 , . . . , Xk such that all atomic formulas Bi become true.

$$pl(P) \models \exists X_1 \ldots \exists X_k (B_1 \wedge \ldots \wedge B_n) \tag{1}$$

$$pl(P) \cup \{\neg \exists X_1 \ldots \exists X_k (B_1 \wedge \ldots \wedge B_n)\} \text{ is unsatisfiable} \tag{2}$$

Monotonic Rules

Semantics: Predicate Logic Semantics

- For example, suppose P is the program
  - p(a)
  - $p(X) \to q(X)$

- Consider the query $q(X) \to$
  - q(a) follows from pl(P). Therefore, $\exists X\, q(X)$ follows from pl(P), thus pl(P)$\cup\{\neg\exists X q(X)\}$ is unsatisfiable, and we give a positive answer.

- But if we consider the query $q(b) \to$ then we must give a negative answer because q(b) does not follow from pl(P).

Rewrite (2) using disjunctive statements.

Monotonic Rules

Semantics: Predicate Logic Semantics

- The components of the logical language (signature) may have any meaning we like. A predicate logic model, A, assigns a certain meaning. In particular, it consists of
  - a domain dom(A), a nonempty set of objects about which the formulas make statements,
  - an element from the domain for each constant,
  - a concrete function on dom(A) for every function symbol,
  - a concrete relation on dom(A) for every predicate.

Monotonic Rules

Semantics: Predicate Logic Semantics

- When the symbol = is used to denote equality (i.e., its interpretation is fixed), we talk of Horn logic with equality.

- The meanings of the logical connectives ¬, ∨, ∧, →, ∀, ∃ are defined according to their intuitive meaning: not, or, and, implies, for all, there is.

- This way we define when a formula is true in a model A, denoted as A |= φ.

Monotonic Rules

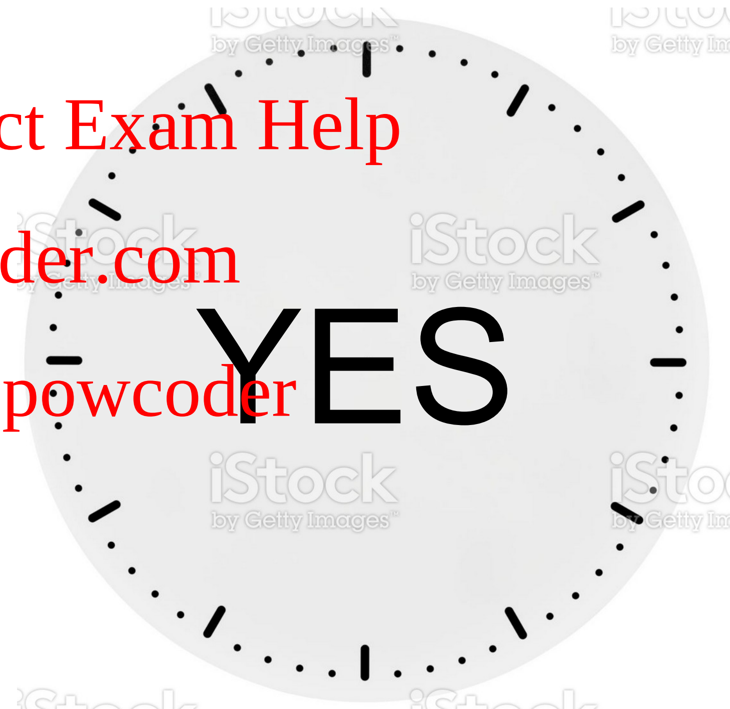
Semantics: Ground and Parameterized Witnesses

- So far we have focused on yes/no answers to queries. However, such answers are not necessarily optimal.
- Suppose that we have the fact p(a) and the query p(X) →
- The answer yes is correct but not satisfactory.

Monotonic Rules

Semantics: Ground and Parameterized Witnesses

- It resembles the joke where you are asked, “Do you know what time it is?” and you look at your watch and answer “yes.”



Monotonic Rules

Semantics: Ground and Parameterized Witnesses

- In our example, the appropriate answer is a substitution
- {X/a} which gives an instantiation for X, making the answer positive. The constant a is called a ground witness.

- Given the facts p(a) and p(b), there are two ground witnesses to the same query: a and b. Or equivalently, we should return the substitutions: {X/a} {X/b}

Is that enough?

Monotonic Rules

Semantics: Ground and Parameterized Witnesses

- While valuable, ground witnesses are not always the optimal answer.

- Consider the logic program add(X,0,X), add(X, Y, Z) → add(X, s(Y), s(Z)). *s* is defined as the “successor function,” which returns as value the value of its argument plus 1.

$$add(X, s^8(0), Z) \rightarrow$$

Monotonic Rules

Semantics: Ground and Parameterized Witnesses

$$add(X, s^8(0), Z) \rightarrow$$

$$\{X/0, Z/s^8(0)\}$$

$$\{X/s(0), Z/s^9(0)\}$$

$$\{X/s(s(0)), Z/s^{10}(0)\}$$

$$\ldots$$

Monotonic Rules

Semantics: Ground and Parameterized Witnesses

- The parameterized witness Z = s8(X) is the most general way to witness the existential query

$$\exists X \exists Z \; add(X, s^8(0), Z)$$

- It represents the fact that add(X, s8(0), Z) is true whenever the value of Z equals the value of X plus 8.

- The computation of most general witnesses is the primary aim of a proof system, called SLD resolution, beyond the scope of this subject.

OWL2 RL: Description Logic Meets Rules

- Integration of Horn logic and description logics

- Consider the intersection of both logics: the part of one language that can be translated in a semantics-preserving way to the other language, and vice versa.

- OWL2 RL seeks to capture this fragment of OWL.

OWL2 RL: Description Logic Meets Rules

- A triple of the form (a, P, b) in RDF can be expressed as a fact P(a,b)

- An instance declaration of the form type(a, C), stating that *a* is an instance of class C, can be expressed as C (a)

- C is a subclass of D: $C(X) \to D(X)$
- C is the domain of property P : $P(X, Y) \to C(X)$

OWL2 RL: Description Logic Meets Rules

- equivalentClass(C, D):

  $C(X) \rightarrow D(X)$

  $D(X) \rightarrow C(X)$

Horn logic for

- P1 is a sub property of P2
- equivalentProperty(C, D)

OWL2 RL: Description Logic Meets Rules

- Transitivity of a property P is easily expressed as

  P (X, Y), P (Y, Z) → P (X, Z)

- Intersection of classes C1 and C2 is a subclass of D:

- C is a subclass of the intersection of D1 and D2:

OWL2 RL: Description Logic Meets Rules

- Transitivity of a property P is easily expressed as

  $P(X, Y), P(Y, Z) \rightarrow P(X, Z)$

- Intersection of classes C1 and C2 is a subclass of D:

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

- C is a subclass of the intersection of D1 and D2:

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

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

OWL2 RL: Description Logic Meets Rules

- The union of C1 and C2 is a subclass of D

- C is a subclass of the union of D1 and D2

OWL2 RL: Description Logic Meets Rules

- The union of C1 and C2 is a subclass of D

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

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

- C is a subclass of the union of D1 and D2

  would require a disjunction in the head of the corresponding rule, which is not available in Horn logic.

Can you think of any special cases where "C is a subclass of the union of D1 and D2" can be expressed in Horn logic?

OWL2 RL: Description Logic Meets Rules

:C rdfs:subClassOf [rdf:type owl:Restriction ; owl:onProperty:P ; owl:allValuesFrom :D] .

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

OWL2 RL: Description Logic Meets Rules

[rdf:type owl:Restriction ; owl:onProperty :P ; owl:someValuesFrom :D] rdfs:subClassOf :C .

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

- https://www.youtube.com/watch?v=km0bkWG068w&ab_channel=MinhTr%E1%BA%A7n%C4%90%E1%BB%A9c
- https://www.youtube.com/watch?v=QsFmwwxU8aY&ab_channel=OpenHPITutorials