PUCPR

Graduate Program in Production Engineering and Systems (PPGEPS)

Important Terms in an Ontology (Part 5/6)

Dr. Yongxin Liao

PUCPR

Course Outlines

- Important Terms in an Ontology (5/6)
  - SWRL (a Semantic Web Rule Language)
    - SWRL and OWL
    - Knowledge Base
    - SWRL Syntax
      - Class Atoms
      - Object Property Atoms
      - Data Property Atoms
      - Individual Equality/Inequality Atoms
      - Built-Ins Atoms
        - » Built-Ins for Comparisons
        - » Built-Ins for Math
        - » Built-Ins for Strings
- Protégé Practices

PUCPR

Important Terms in an Ontology

- Several Important Terms
  - Axioms
  - Concepts (Individuals and Classes)
  - Relationships (Class Assertions, Subclasses Disjoint/Equivalent Classes, Individual Equality/Inequality Properties, Property Assertions, Property Characteristics, and Property Descriptions)
  - Complex Class Expressions (Enumeration of Individuals, Propositional Connectives, Object Property Restrictions, Necessary and Sufficient Conditions, Data Property Restrictions)
  - Data Ranges (Data Types and Data Type Restrictions)
  - **Reasoning Rules**
  - **Knowledge Base (T-box and A-box)**
  - SPARQL Query

Part 1/6

Part 2/6

Part 3/6

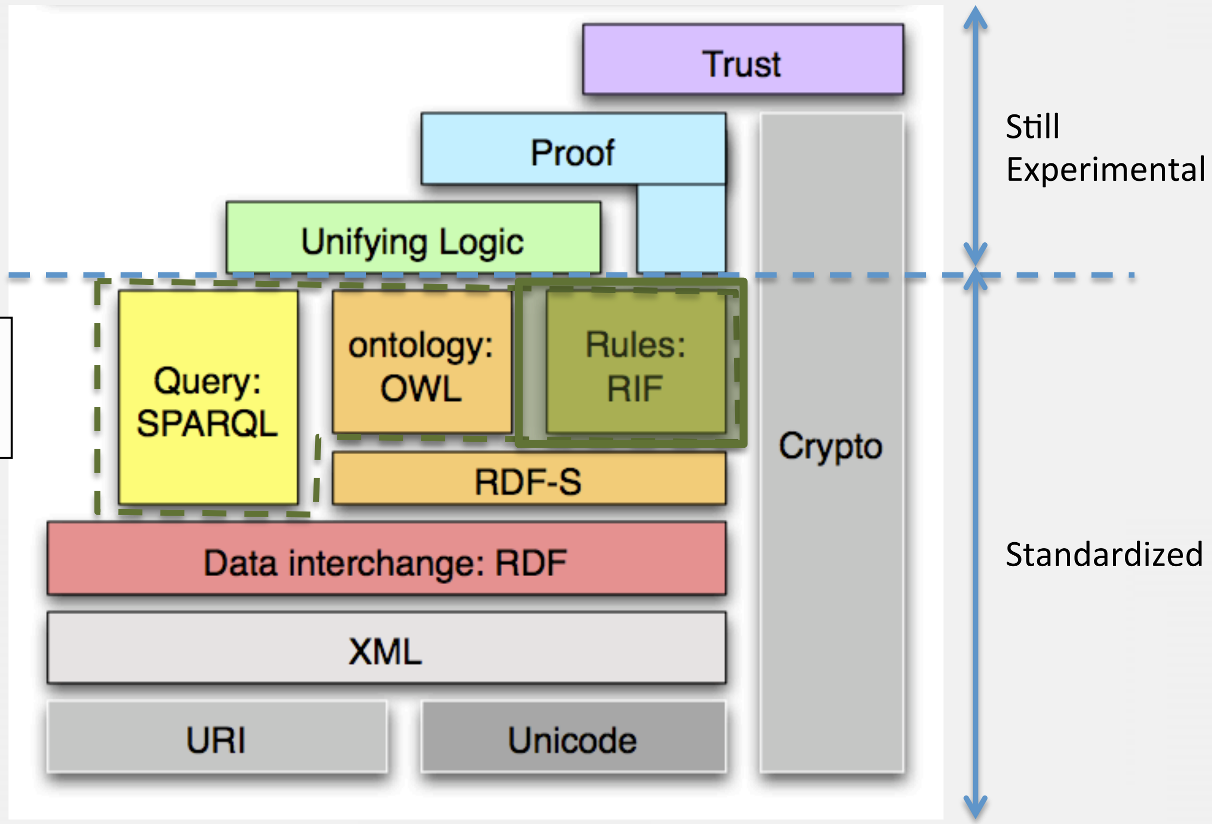
Part 4/6

Part 5/6

Part 6/6

PUCPR

Reasoning Rules



SWRL (a Semantic Web Rule Language)

- SWRL is an OWL-based rule language
  - As a complement to OWL expressions
  - A W3C submission published in 2004
  - Human readable syntax
  - A rule is considered as a rule axiom in an ontology
  - All rules are expressed through terms in OWL (classes, properties, individuals).
  - Protégé OWL editor and its reasoners provides supports for SWRL.

4 Keywords for a Knowledge Base

- T-Box: Terminological Box.
  - A set of “schema” axioms
  - It describes a set of **Classes** and **Properties** and the ways that they related to each other.
    - Classes (Lecture 02)
      - E.g. Pizza, PizzaTopping, PizzaBase
    - Subclasses Axioms (Lecture 02)
      - E.g. MargheritaPizza is a subclass of NamedPizza
    - Disjoint/Equivalent Classes Axioms (Lecture 02)
      - E.g. MargheritaPizzais disjoint with AmericanHotPizza
    - Properties (Lecture 03)
      - E.g. hasTopping, hasWeightInGrams
    - Data Ranges (Lecture 05)
      - E.g. decimal, decimal[>= 350, <= 450]
    - Complex class expressions (Lecture 04 and 05)
      - E.g. hasTopping some MozzarellaTopping
        hasWeightInGrams exactly 1 decimal [>= 360 , <= 380]

PUCPR

4 Keywords for a Knowledge Base

- A-Box: Assertion Box.
  - A set of “data” Axioms
  - It describes a set of **Individuals** with the **Classes** that they belong to and the ways that those individual related to each other.
    - Individuals (Lecture 02)
      - E.g. PizzaID1, PizzaID2
    - Class Assertion Axioms (Lecture 02)
      - E.g. PizzaID1 is members of Pizza
    - Individual Equality/Inequality Axioms (Lecture 02)
      - E.g. PizzaID1 is different from PizzaID2
    - Property Assertion Axioms (Lecture 03)
      - E.g. PizzaID1 hasTopping MozzarellaToppingNo1-1
PizzaID1 hasWeightInGrams 470

PUCPR

4 Keywords for a Knowledge Base

- Reasoning Rules (Lecture 06)
  - As a complement of T-Box and A-Box.
    - A set of “rule” Axioms
      - E.g. A rule to discover all the pizza individual for the class *HeavyCheesePizza*. They should fulfill the following conditions:
        - » Each of them has a cheese topping
        - » The weight of the cheese topping is heavier than 50% of its total weight.
- Reasoning Results
  - The inferred new knowledge
    - A set of new axioms that discovered by reasoners
      - E.g. inferred new subclasses of *InterestingPizza* (Lecture 02 to 05)
      - E.g. inferred new members of class *HeavyCheesePizza* (Lecture 06)

PUCPR

Knowledge Base

T-Box

+ A-Box

(+ Reasoning Rules)

+ Reasoning Results

= Knowledge Base

PUCPR

SWRL Syntax

- Rules
  - A rule is consist of an antecedent and a consequent.

A Rule: atom, …, atom -> atom, …, atom

- An antecedent or a consequent may have one or more atoms
- Atoms in an antecedent or a consequent are separated by ","
- An antecedent is connected to a consequent via "->"

E.g.

$\underbrace{\text{Human(?x), hasMother(?x, ?y), hasBrother(?y, ?z)}}_{\text{Antecedent}}$ -> $\underbrace{\text{hasUncle (?x ,?z)}}_{\text{Consequent}}$

atom atom atom atom

SWRL Syntax

- Atoms
  - An atom is an expression of the form:

    predicate symbol (argument$_1$, argument$_2$, … , argument$_n$)

    - The predicate symbols can be Classes, Properties, Data Ranges, Build-Ins.
    - The arguments can be Individuals, Data Values, or Variables

E.g.

$$\underbrace{\text{Human}}_{\text{Class}}(\overset{\text{variable}}{?x}),\ \underbrace{\text{hasMother}}_{\text{Property}}(\overset{\text{variable}}{?x},\ \overset{\text{variable}}{?y}),\ \underbrace{\text{hasBrother}}_{\text{Property}}(\overset{\text{variable}}{?y},\ \overset{\text{variable}}{?z})\ \text{->}\ \underbrace{\text{hasUncle}}_{\text{Property}}\ (\overset{\text{variable}}{?x}\ ,\overset{\text{variable}}{?z})$$

SWRL Syntax

- Variables
  - A variable can be considered as a storage location that contains some quantity of individuals or data values that make its atom become true.
    - It is indicated by using *a question mark ("?")* as its prefix and then associating by *a symbolic name*.

a question mark

?x

a symbolic name

E.g.

Human(?x), hasMother(?x, ?y), hasBrother(?y, ?z) -> hasUncle (?x ,?z)

SWRL Syntax

Class (argument_1)

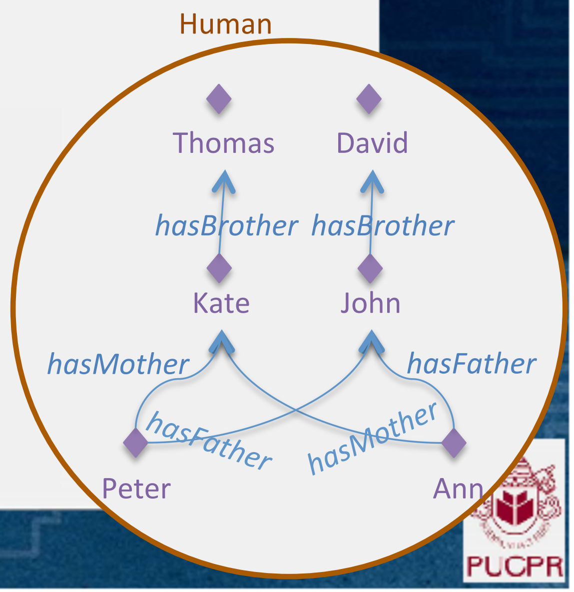
- Class Atoms
  - A class atom consists of a named class (or a complex class expression) and <u>a single argument</u> that representing an individual.
  - If the individuals that represented by the argument are members of this class, then this class atom becomes true.

E.g.

- Class Atom: Human(?x)
  If ?x is Thomas, David, Kate, John, Peter, or Ann,
  then Human(?x) is true.

- Class Atom: Human(Thomas)   Class Atom: Human(Pappy)

  true



SWRL Syntax

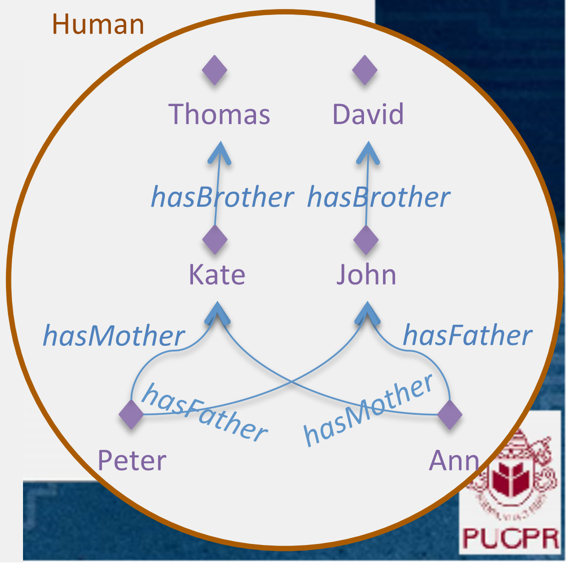
Object Property (argument$_1$,argument$_2$)

- Object Property Atoms
  - An object property atom consists of an object property and <u>two arguments</u> that represent two individuals.
  - If the individual that represented by argument$_1$ is connect to the individual that represented by argument$_2$ via this object property, then this object property atom becomes true.

E.g.

- Object Property Atom: hasMother(?x,?y)
  (1) If ?x = Peter and ?y = Kate, then hasMother(?x,?y) is true.
  (2) If ?x = Ann and ?y = Kate, then hasMother(?x,?y) is true
- Object Property Atom: hasMother(?x, Kate)
  (1) If ?x = Peter, then hasMother(?x, Kate) is true
  (2) If ?x = Ann, then hasMother(?x, Kate) is true
- Object Property Atom: hasMother(Peter, Kate) $\boxed{\text{true}}$

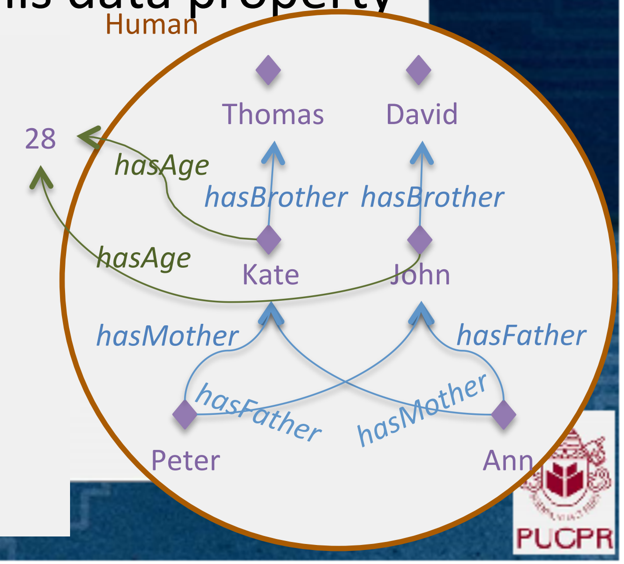
  Object Property Atom: hasMother(Peter, John)



SWRL Syntax

Data Property ($argument_1$,$argument_2$)

- Data Property Atoms
  - An data property atom consists of an data property and two arguments.
    - The first argument represents an individual.
    - The second argument represents a data value.
  - If the individual that represented by $argument_1$ is connect to the data value that represented by $argument_2$ via this data property, then this data property atom becomes true.



E.g.

- Data Property Atom: hasAge (?x,?y)
  (1) If ?x= Kate and ?y= 28, then hasAge (?x,?y) is true.
  (2) If ?x= John and ?y= 28, then hasAge (?x,?y) is true.
- Data Property Atom: hasAge (Kate,?y)
  (1) If ?y= 28, then hasAge (Kate,?y) is true.
- Data Property Atom: hasAge (Kate, 28) true

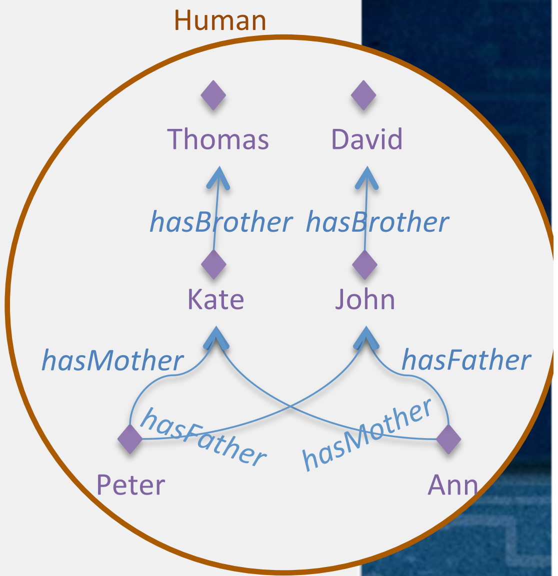
  Data Property Atom: hasAge (Kate, 38)

SWRL Syntax

- Note (1/4)
  - The variables in a given rule stores the individuals or data values that make **all** atoms in the antecedent becomes true.

E.g.

$\underbrace{\text{Human(?x), hasMother(?x, ?y), hasBrother(?y, ?z)}}_{\text{How to make it becomes true?}}$ -> hasUncle (?x ,?z)



a) To make Human(?x) becomes true

?x can be Thomas, David, Kate, John, Peter, or Ann

b) To make Human(?x), hasMother(?x, ?y) becomes true

(1) ?x= Peter ?y= Kate

(2) ?x= Ann ?y= Kate

c) To make Human(?x), hasMother(?x, ?y), hasBrother(?y, ?z) becomes true

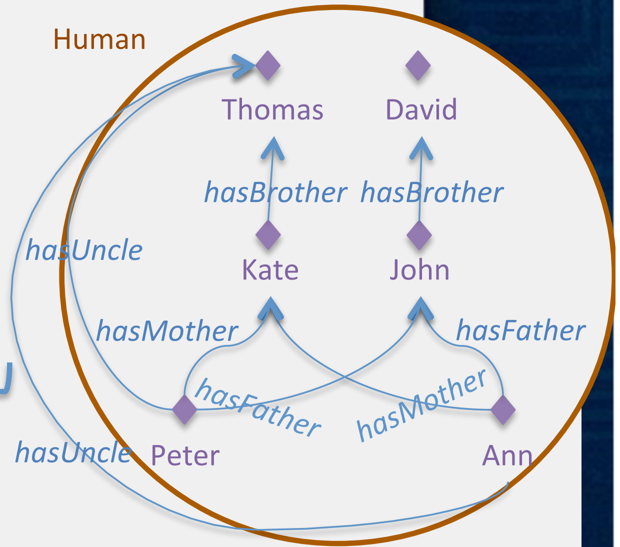
(1) ?x= Peter ?y= Kate ?z= Thomas

(2) ?x= Ann ?y= Kate ?z= Thomas

PUCPR

SWRL Syntax

- Note (2/4)
  – If all the atoms in the antecedent are true, then all the atoms in the consequent must also be true.



E.g.

Human(?x), hasMother(?x, ?y), hasBrother(?y, ?z) -> hasUncle (?x ,?z)

true   true

- If Human(?x), hasMother(?x, ?y), hasBrother(?y, ?z) is true, then hasUncle (?x ,?z) must be true

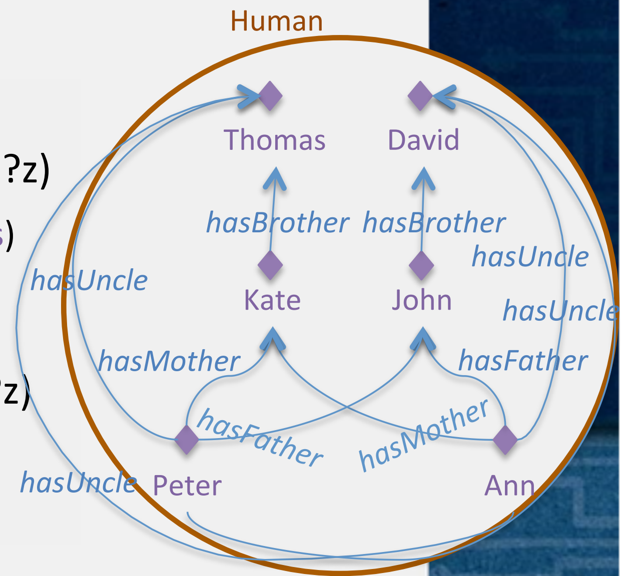
  (1) ?x = Peter, ?y = Kate, ?z = Thomas -> hasUncle (Peter, Thomas)

  (2) ?x = Ann, ?y = Kate, ?z = Thomas -> hasUncle (Ann, Thomas)

PUCPR

SWRL Syntax

- Note (3/4)
  - The scope of a variable is limited inside a given rule.
    - Even if "same" variables (variables with same symbolic names) is used in different rules, they will not affect each other.



E.g.

Rule 1: Human(?x), hasMother(?x, ?y), hasBrother(?y, ?z) -> hasUncle (?x ,?z)

(1) ?x= Peter ?y= Kate ?z= Thomas -> hasUncle (Peter, Thomas)

(2) ?x= Ann ?y= Kate ?z= Thomas -> hasUncle (Ann, Thomas)

Rule 2: Human(?x), hasFather(?x, ?y), hasBrother(?y, ?z) -> hasUncle (?x, ?z)

(1) ?x= Peter ?y= John ?z= David -> hasUncle (Peter, David)

(2) ?x= Ann ?y= John ?z= David -> hasUncle (Ann, David)

SWRL Syntax

- Note (4/4)
  - A rule can not create new classes, properties, or individuals.
    - Predicate symbols in a rule can not contain the classes or properties that do not exist in the ontology
    - Arguments in a rule can not contain the individuals that do not exist in the ontology

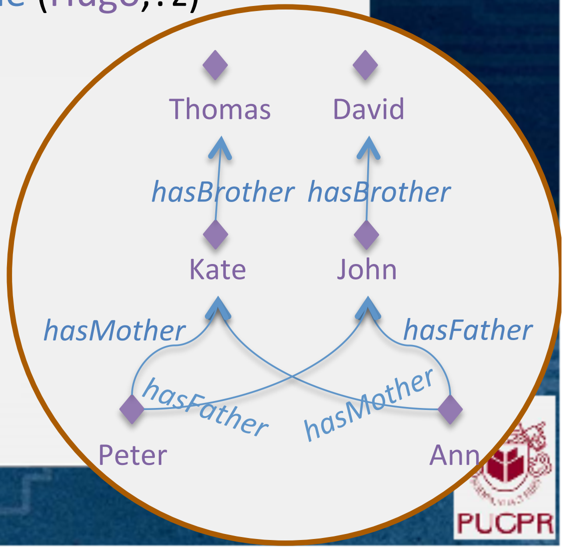
E.g.
MaleHuman(Hugo), isMotherOf(Kate, Hugo), hasBrother(Kate, ?z) -> hasUncle (Hugo,?z)

This property does not exist

This class does not exist

This individual does not exist

Human



PUCPR

SWRL Syntax

SameAs($argument_1$,$argument_2$)

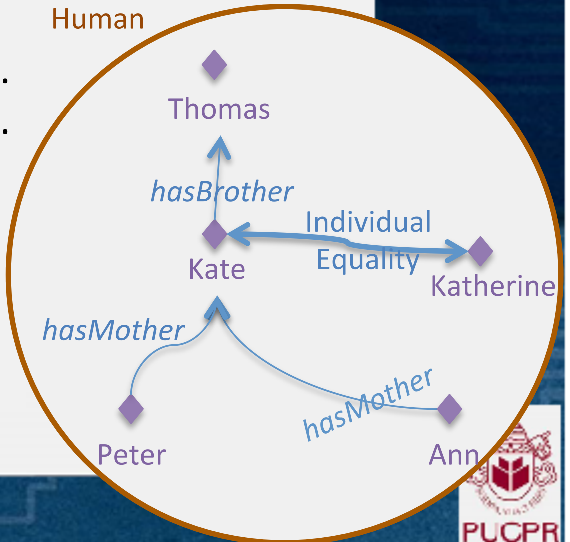
- Individuals Equality Atoms
  - An Individuals Equality atom consists of the predicate symbol "SameAs" and *two arguments* that represent two individuals.
  - If the individual that represented by $argument_1$ is the same as the individual that represented by $argument_2$, then this individuals equality atom becomes true.

E.g.

- Individuals Equality Atom: SameAs(?x,?y)
  (1) if ?x = Katherine and ?y= Kate, then SameAs(?x,?y) is true.
  (2) if ?x = Kate and ?y= Katherine, then SameAs(?x,?y) is true.
- Individuals Equality Atom: SameAs (?x, Kate)
  (1) if ?x= Katherine, then SameAs(?x, Kate) is true
- Individuals Equality Atom: SameAs(Katherine, Kate) $\boxed{\text{true}}$

Individuals Equality Atom: SameAs(Peter,Kate)

All object properties are irreflexive



PUCPR

SWRL Syntax

DifferentFrom($argument_1$,$argument_2$)

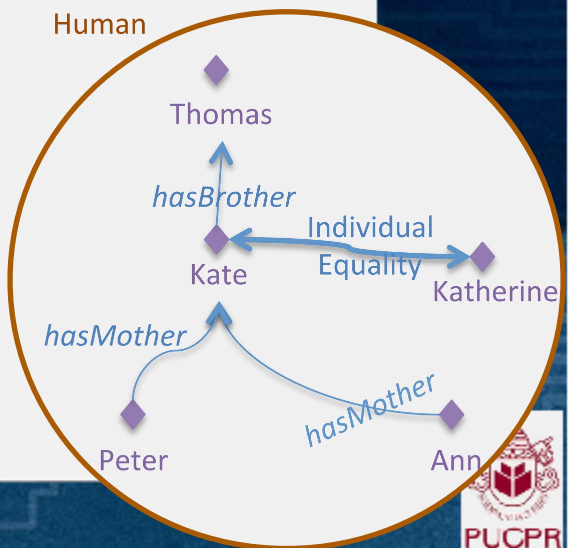
- Individuals Inequality Atoms
  - An Individuals Inequality atom consists of the predicate symbol "DifferentFrom" and two arguments that represent two individuals.
  - If the individual that represented by $argument_1$ is the different from the individual that represented by $argument_2$, then this individuals inequality atom becomes true.

All object properties are irreflexive

E.g.

- Individuals Inequality Atom: DifferentFrom(?x,?y)
  (1) If ?x= Thomas and ?y= Peter (Ann, Katherine or Kate), then DifferentFrom(?x,?y) is true.
  (2) …
- Individuals Inequality Atom: DifferentFrom(?x, Kate)
  (1) ?x= Thomas (2) ?x= Peter (3) ?x= Ann
- Individuals Inequality Atom: DifferentFrom(Peter, Kate)
  Individuals Inequality Atom: DifferentFrom(Katherine, Kate)

true



PUCPR

SWRL Syntax

- Built-In Atoms for Comparison
  - An atom consists of a *comparison symbol* and *two arguments* that represent *two data values*.
    - swrlb:equal ($argument_1$,$argument_2$)
    - swrlb:notEqual ($argument_1$,$argument_2$)
    - swrlb:lessThan ($argument_1$,$argument_2$)
    - swrlb:lessThanOrEqual ($argument_1$,$argument_2$)
    - swrlb:greaterThan ($argument_1$,$argument_2$)
    - swrlb:greaterThanOrEqual ($argument_1$,$argument_2$)
  - If the data value that represented by $argument_1$ and the data value that represented by $argument_2$ fulfill the comparison description then this atom becomes true.

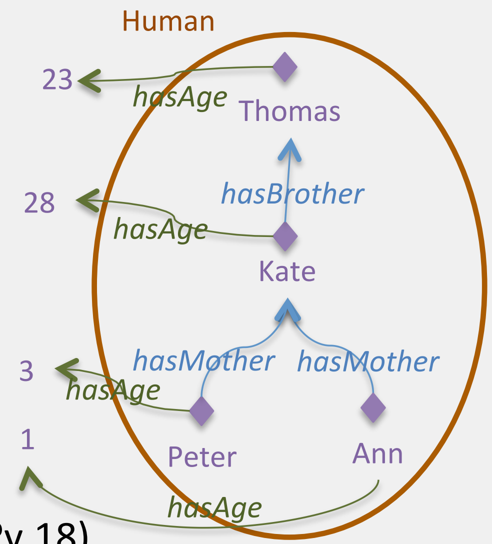
PUCPR

SWRL Syntax

- Built-In Atoms for Comparison
  - An atom consists of a *comparison symbol* and *two arguments* that represent *two data values*.

E.g.

- Human(?x),hasAge(?x,?y), swrlb:equal (?y,23)
  (1) ?x=Thomas   ?y= 23
- Human(?x),hasAge(?x,?y), swrlb:notEequal (?y,23)
  (1) ?x= Kate   ?y= 28
  (2) ?x= Peter   ?y= 3
  (3) ?x= Ann   ?y= 1
- Human(?x),hasAge(?x,?y), swrlb:lessThan (?y,18)
  (1) ?x= Peter   ?y= 3
  (2) ?x= Ann   ?y= 1
- Human(?x),hasAge(?x,?y), swrlb:greaterThanOrEqual (?y,18)
  (1) ?x= Kate   ?y= 28
  (2) ?x= Thomas   ?y= 23



SWRL Syntax

- Built-In Atoms for Math
  - An atom consists of ***a math symbol*** and ***three arguments*** that represent ***three data values***.
    - swrlb:add ($argument_1$,$argument_2$, $argument_3$)
      - $Argument_1$ = $argument_2$ + $argument_3$
    - swrlb:subtract ($argument_1$,$argument_2$, $argument_3$)
      - $Argument_1$ = $argument_2$ - $argument_3$
    - swrlb:multiply ($argument_1$,$argument_2$, $argument_3$)
      - $Argument_1$ = $argument_2$ * $argument_3$
    - swrlb:divide ($argument_1$,$argument_2$, $argument_3$)
      - $Argument_1$ = $argument_2$ / $argument_3$

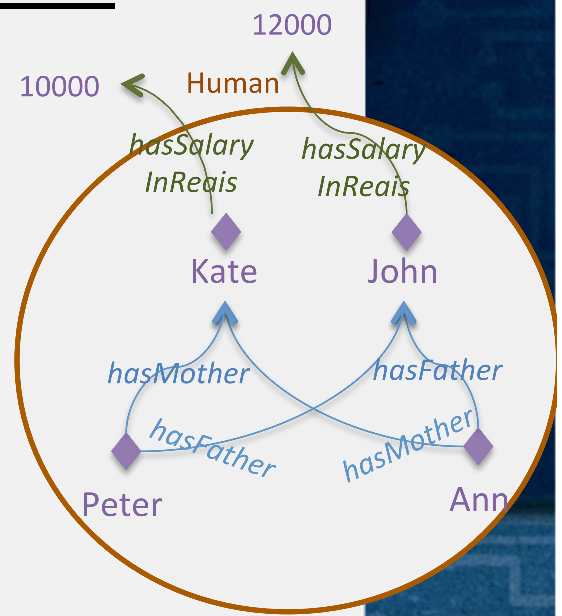
PUCPR

SWRL Syntax

- Built-In Atoms for Math
  - An atom consists of *__a math symbol__* and *__three arguments__* that represent *__three data values__*.

E.g.

- **Everyone earns 1000 Reais more?**
  Human(?x),hasSalaryInReais(?x,?y), swrlb:add (?z, ?y, 1000)
  (1) ?x= Kate ?y= 10000 ?z=11000
  (2) ?x= John ?y= 12000 ?z=13000
- **Everyone earns 1000 Reais less?**
  Human(?x),hasSalaryInReais(?x,?y), swrlb:subtract(?z,?y,1000)
  (1) ?x= Kate ?y= 10000 ?z=9000
  (2) ?x= John ?y= 12000 ?z=11000
- **Salaries In Chinese Yuan?**
  Human(?x),hasSalaryInReais(?x,?y), swrlb:multiply(?z,?y, 2)
  (1) ?x= Kate ?y= 10000 ?z=20000
  (2) ?x= John ?y= 12000 ?z=24000
- **Salaries in euros?**
  Human(?x),hasSalaryInReais(?x,?y), swrlb:divide(?z,?y, 4)
  (1) ?x= Kate ?y= 10000 ?z=2500
  (2) ?x= John ?y= 12000 ?z=3000



SWRL Syntax

- Built-In Atoms for Strings
  - swrlb:startsWith ($argument_1$,$argument_2$)
    - If $argument_1$ starts with $argument_2$, then this atom is true.
    - E.g. swrlb:startsWith (“abc”, “a”)
  - swrlb:endsWith ($argument_1$,$argument_2$)
    - If $argument_1$ ends with $argument_2$, then this atom is true
    - E.g. swrlb:endsWith (“abc”, “c”)
  - swrlb:stringLength($argument_1$,$argument_2$)
    - If $argument_1$ is the length of $argument_2$, then this atom is true.
    - E.g. swrlb:stringLength (3, “abc”)
  - swrlb:contains ($argument_1$,$argument_2$)
    - If $argument_1$ contains $argument_2$, then this atom is true.
    - E.g. swrlb:cantains(“abc”, “ab”)

PUCPR

Knowledge Base

T-Box	✓ YES!	Lecture 02 - 05
+ A-Box	☐	Lecture 06
(+ Reasoning Rules)	☐	Lecture 06
+ Reasoning Results	✓ YES! (Lecture 02 - 05) ☐ (Lecture 06)	

= Knowledge Base

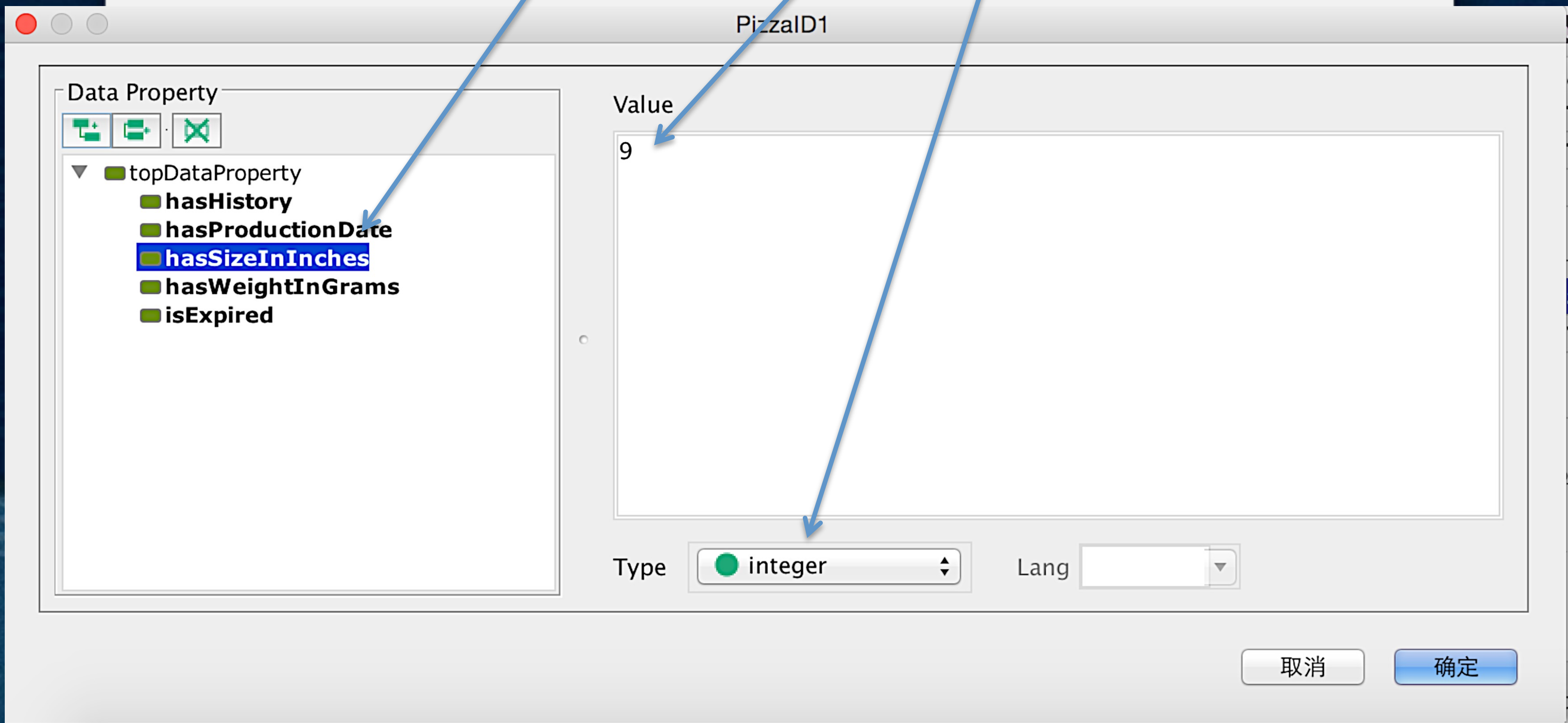
PUCPR

Protégé Practices

- Apply Reasoning Rules in Pizza Ontology
  - Data Preparation for next Two Lectures
    - Reasoning Rules
    - SPARQL Query
  - Make sure the individuals, object and data property assertions are correct!

Protégé Practices

PizzaID1 hasSizeInInches “9”^^integer



Protégé Practice

- The Pizza Ontology
  - Open the given pizza ontology (with individuals)
  - Construct the following rules:
    (1) Use a new object property isTheSameSizeAs to connect those pizza individuals that have the same size in inches.
    (2) Use a new object property isHeavierThan to compare the weights of pizza individuals
    (3) Use a new Class HeavyCheesePizza to classify those pizza individuals that fulfill the following conditions:
      - Each of them has a cheese topping
      - The weight of the cheese topping is heavier than 50% of its total weight.
    (4) Use a new Class ThickCrustPizza to classify those pizza individuals that fulfill the following conditions:
      - Each of them has the size 9 inches
      - Each of them has a pizza base heavier than 300 grams.

PUCPR

Thank you
for your attention!

Any Questions?