Knowledge-Based Systems

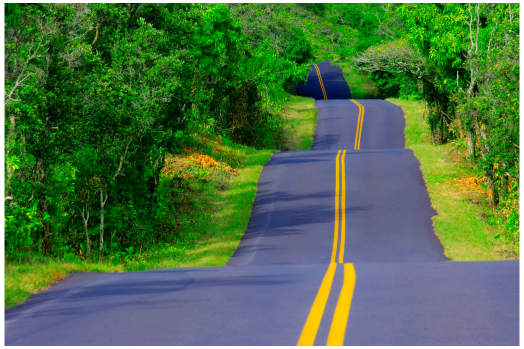
Competency Question Driven Ontology Modelling

Jeff Z. Pan

http://homepages.abdn.ac.uk/jeff.z.pan/pages/

UNIVERSITY OF ABERDEEN

Roadmap

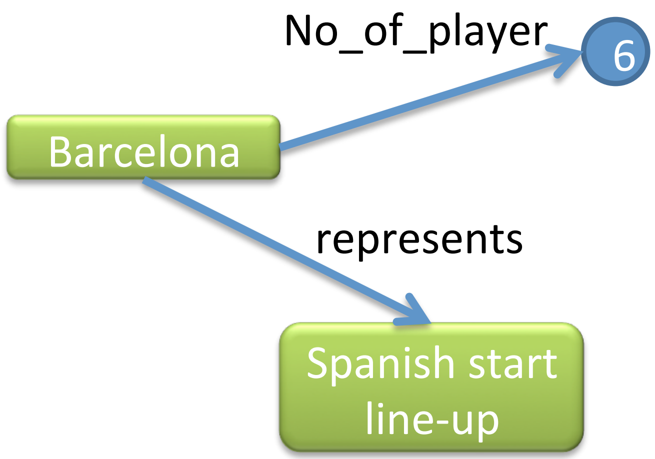


- Foundation
  - KR, ontology and rule; set theory
- Knowledge capture
- Knowledge representation
  - Ontology: Semantic Web standards RDF and OWL, Description Logics
  - Rule: Jess
- Knowledge reasoning
  - Ontology: formal semantics, tableaux algorithm
  - Rule: forward chaining, backward chaining
- Knowledge reuse and evaluation
- Meeting the real world
  - Legal ontologies, Jess and Java, Invited talk

A Process of Refinement

- The procedure of constructing an ontology is also a procedure in which you examine and refine your semantic networks!

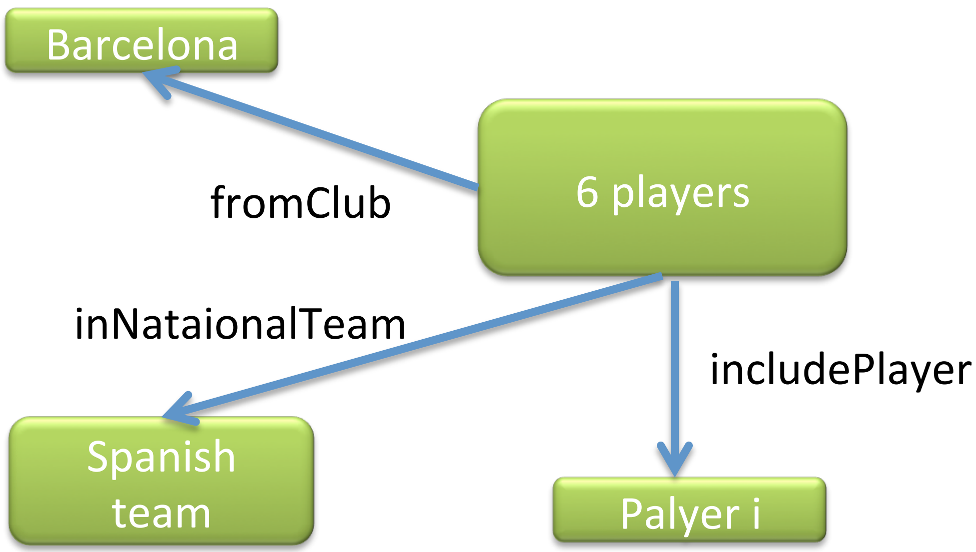
- A typical example: multi-entity relationships
  – E.g. *"Six players from Barcelona were in the Spanish starting line-up"*
    - This is a complex relationship involving a "Country's team", a "club team", and "6" players
  – In semantic networks, one may draw a figure like this or similar:



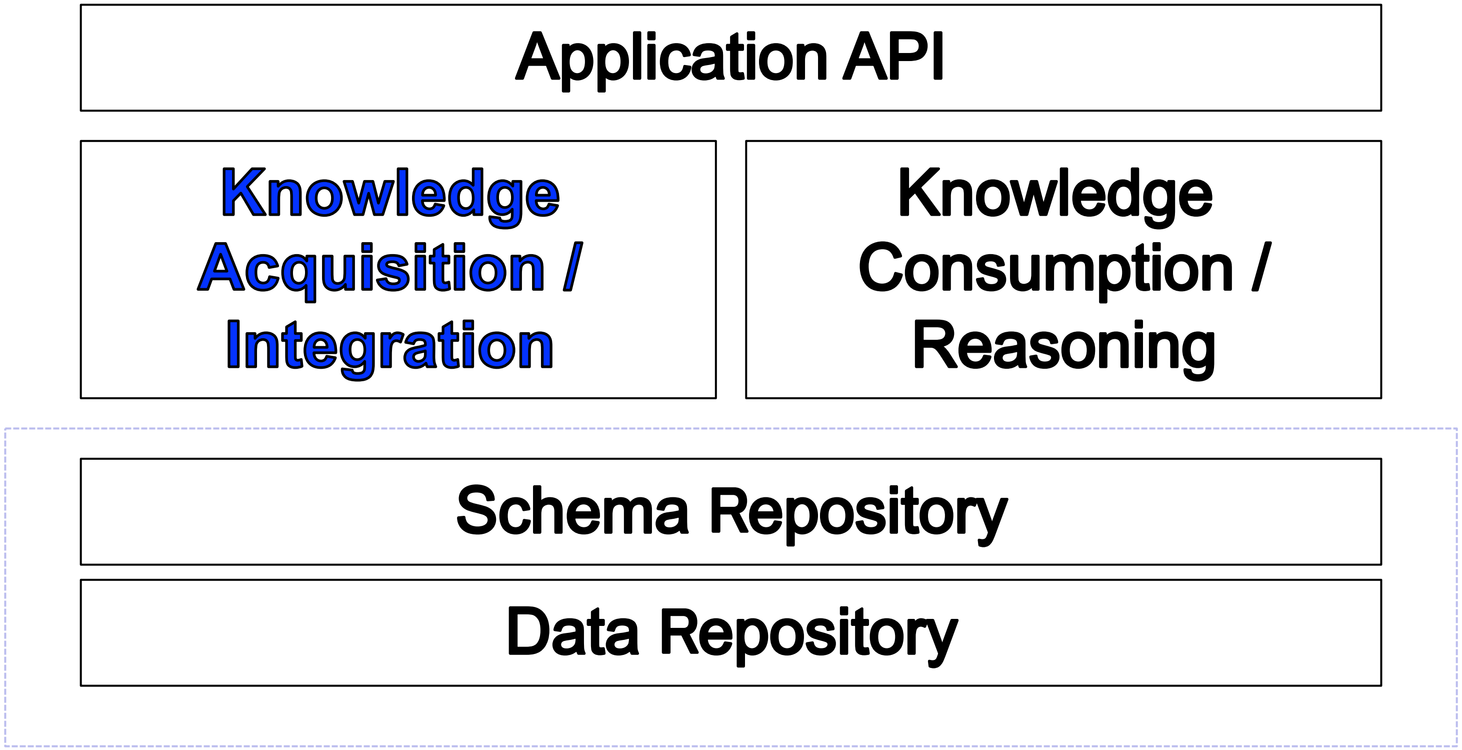
- This is not semantically precise:
  - Barcelona has 6 players?
    - No, it has 6 players in the Spanish start line-up
  - Barcelona represents Spain?
    - No, its 6 players represent Spain in the start line-up
- This relationship makes sense only when you take all these entities into account!

A Process of Refinement

- An alternative representation
  - there could be others
- Constructing ontology helps you re-evaluate and refine your semantic network.



Architecture of Knowledge Based Systems



RDF Schema Entailment Rules (1)

- [rdfs2]
  - [p rdfs:domain C .] [a p b .] => [a rdf:type C .]
- [rdfs3]
  - [p rdfs:range D .] [a p b .] => [b rdf:type D .]
- [rdfs5]
  - [p1 rdfs:subPropertyOf p2 .] [p2 rdfs:subPropertyOf p3 .] =>

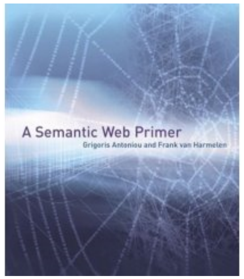
    [p1 rdfs:subPropertyOf p3 .]

RDF Schema Entailment Rules (2)

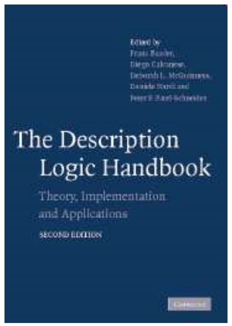
- [rdfs7]
  - [p1 rdfs:subPropertyOf p2 .] [a p1 b .] => [a p2 b .]
- [rdfs9]
  - [C rdfs:subClassOf D .] [b rdf:type C .] => [b rdf:type D .]
- [rdfs11]
  - [C1 rdfs:subClassOf C2 .] [C2 rdfs:subClassOf C3 .] =>
    [C1 rdfs:subClassOf C3 .]

Lecture Outline

- Motivation
- A brief introduction of OWL
- From Competency Questions to OWL Ontologies
- Practical



[Section 4.1 - 4.3]



[Chapter 14]

Motivations

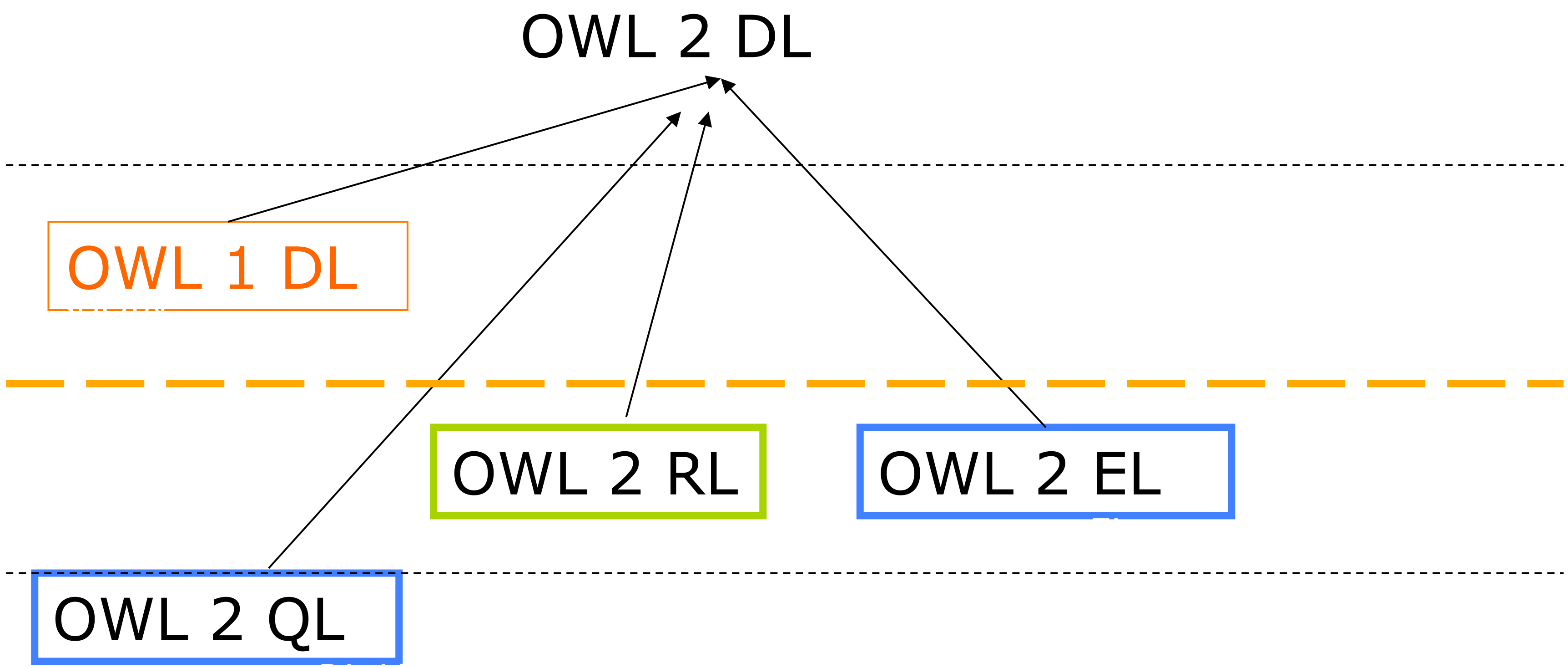
- Constructing ontologies can be expensive
  - Gene Ontology (~5 FT staffs, $25M)
  - National Cancer Institute Metathesaurus (~12 FT staffs, $75M)
  - Health Level 7: $15B?

- Requirement vs. ontologies
  - how to representation requirements for ontology modelling
  - how to ensure ontologies satisfy requirements
  - how to relate requirements to test cases

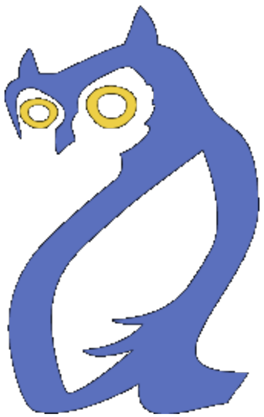
Lecture Outline

- Motivation
- A brief introduction of OWL
- From Competency Questions to OWL Ontologies
- Practical

OWL: Web Ontology Language



How to Write RDF Statements in OWL Axioms



- Normal RDF statements: [my-chair colour tan .]
  - : OWL: Individual (my-chair value(colour tan))
- rdf:type statements: [my-chair rdf:type chair .]
  - OWL: Individual (my-chair type(chair))
- rdfs:subClassOf statements: [chair rdfs:subClassOf furniture .]
  - OWL: SubClassOf(chair furniture)
- rdsf:subProperty statements: [hasTopping rdfs:subPropertyOf hasFishTopping .]
  - OWL: SubPropertyOf(hasTopping hasFishTopping)
- rdfs:domain statements: [hasName rdfs:domain professor)
  - OWL: ObjectProperty(teach domain(professor))
- rdfs:range statements: [teach rdfs:range course]
  - OWL: ObjectProperty(teach range(course))

OWL (DL) Axioms

Abstract Syntax	DL Syntax
Class(A partial C_1 ... C_n)	$A \sqsubseteq C_1 \sqcap \ldots \sqcap C_n$
Class(A complete C_1 ... C_n)	$A \equiv C_1 \sqcap \ldots \sqcap C_n$
EnumeratedClass(A o_1 ... o_n)	$A \equiv \{o_1\} \sqcup \ldots \sqcup \{o_n\}$
SubClassOf(C_1, C_2)	$C_1 \sqsubseteq C_2$
EquivalentClasses(C_1 ... C_n)	$C_1 \equiv \ldots \equiv C_n$
DisjointClasses(C_1 ... C_n)	$C_i \sqsubseteq \neg C_j$, $(1 \leq i < j \leq n)$
SubPropertyOf(R_1, R_2)	$R_1 \sqsubseteq R_2$
EquivalentProperties(R_1 ... R_n)	$R_1 \equiv \ldots \equiv R_n$
ObjectProperty(R super(R_1) ... super(R_n)	$R \sqsubseteq R_i$
domain(C_1) ... domain(C_k)	$\geqslant 1R \sqsubseteq C_i$
range(C_1) ... range(C_h)	$\top \sqsubseteq \forall R.C_i$
[Symmetric]	$R \equiv R^-$
[Functional]	$\mathsf{Func}(R)$
[InverseFunctional]	$\mathsf{Func}(R^-)$
[Transitive])	$\mathsf{Trans}(R)$
AnnotationProperty(R)	
Individual(o type(C_1) ... type(C_n)	$o : C_i, 1 \leq i \leq n$
value(R_1, o_1) ... value(R_n, o_n)	$\langle o, o_i \rangle : R_i, 1 \leq i \leq n$
SameIndividual(o_1 ... o_n)	$o_1 = \ldots = o_n$
DifferentIndividuals(o_1 ... o_n)	$o_i \neq o_j, 1 \leq i < j \leq n$

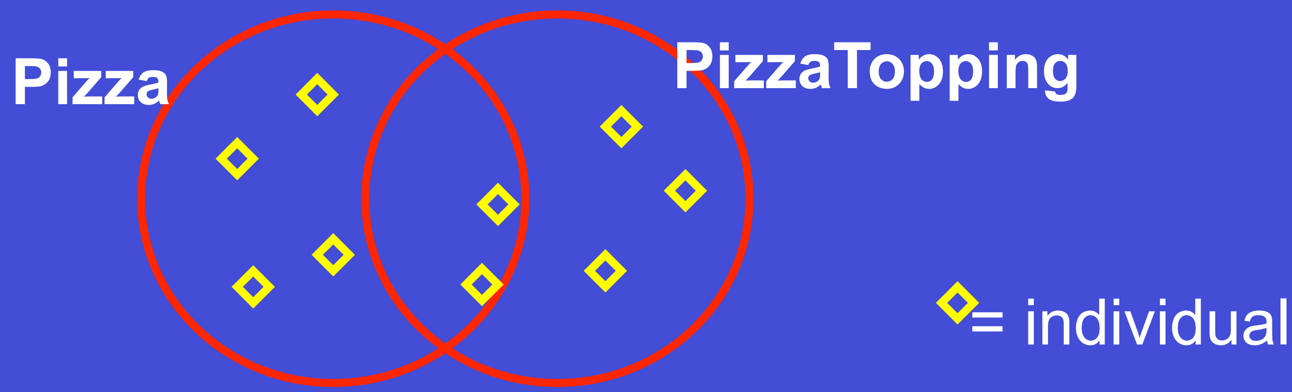
UNIVERSITY OF ABERDEEN

DL Descriptions

Abstract Syntax	DL Syntax
Class(A)	A
Class(owl:Thing)	$\top$
Class(owl:Nothing)	$\bot$
intersectionOf($C_1, C_2, \ldots$)	$C_1 \sqcap C_2$
unionOf($C_1, C_2, \ldots$)	$C_1 \sqcup C_2$
complementOf(C)	$\neg C$
oneOf($o_1, o_2, \ldots$)	$\{o_1\} \sqcup \{o_2\}$
restriction(R someValuesFrom(C))	$\exists R.C$
restriction(R allValuesFrom(C))	$\forall R.C$
restriction(R hasValue(o))	$\exists R.\{o\}$
restriction(R minCardinality(m))	$\geqslant mR$
restriction(R maxCardinality(m))	$\leqslant mR$
restriction(T someValuesFrom(u))	$\exists T.u$
restriction(T allValuesFrom(u))	$\forall T.u$
restriction(T hasValue(w))	$\exists T.\{w\}$
restriction(T minCardinality(m))	$\geqslant mT$
restriction(T maxCardinality(m))	$\leqslant mT$
ObjectProperty(S)	S
ObjectProperty(S' inverseOf(S))	S^-
DatatypeProperty(T)	T

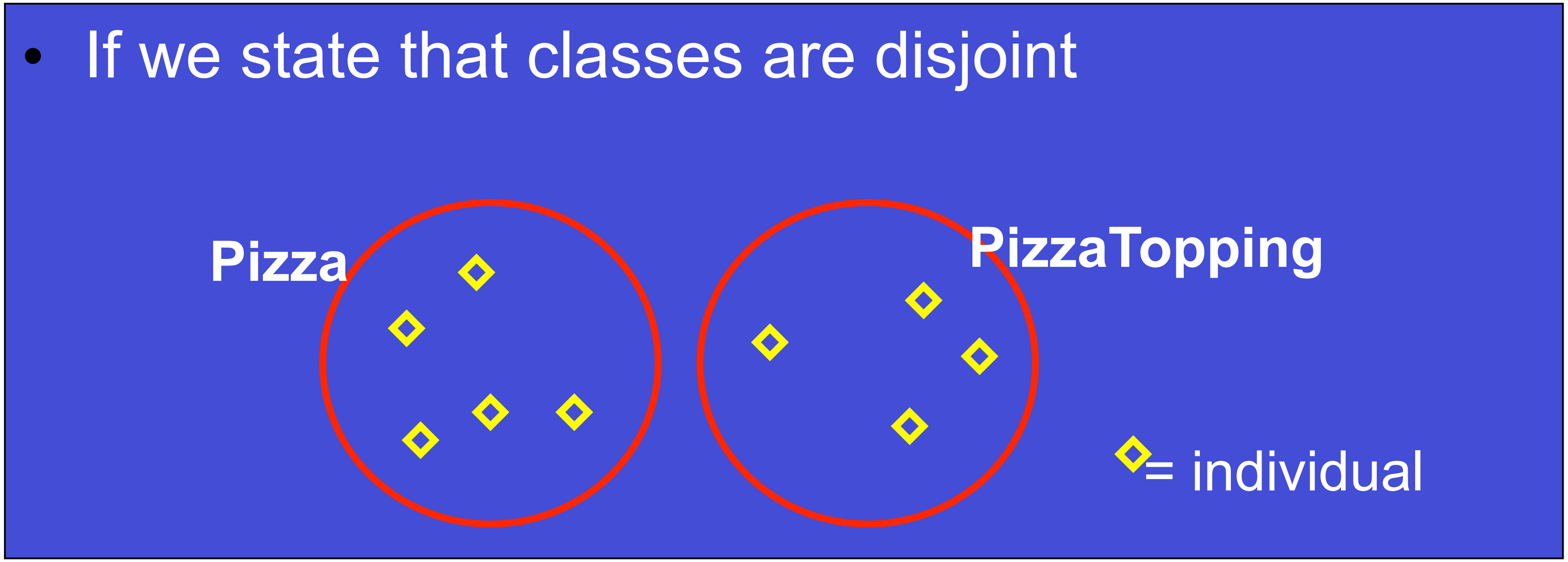
Class Overlapping

- RDF and OWL assume that classes overlap



- This means an individual could be both a **Pizza** and a **PizzaTopping** at the same time
- We want to state this is not the case

Disjointness



- This means an individual cannot be both a **Pizza** and a **PizzaTopping** at the same time
- We must do this explicitly in OWL
  - **DisjointClasses (Pizaa PizzaTopping)**

UNIVERSITY OF ABERDEEN

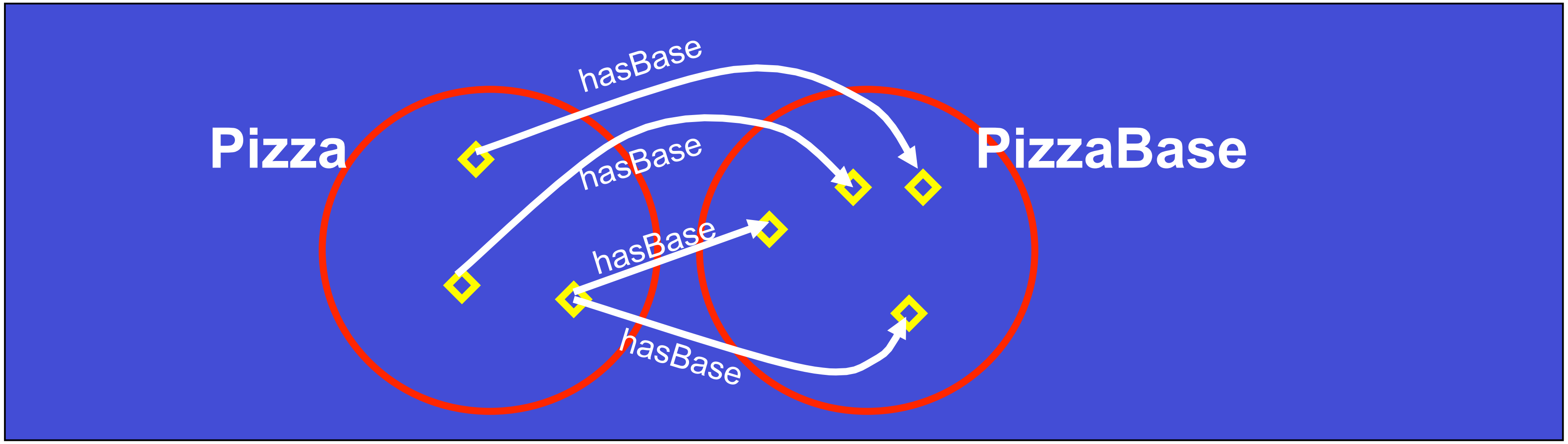
Class Descriptions

Class descriptions are used to categorise sets of individuals

- Named classes are atomic descriptions
- Class descriptions can be built by using constructors

What does this mean?

- We have created a restriction: ∃hasBase.**PizzaBase** on Class **Pizza** as a necessary condition

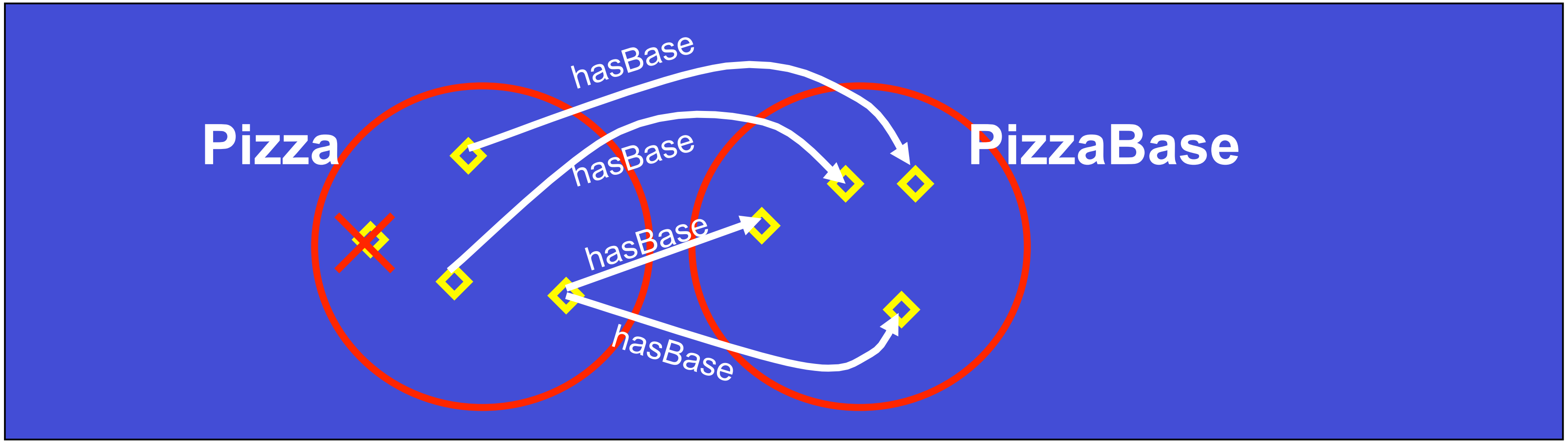


- This restriction categories “individuals that have at least one hasBase relationship with an individual from the class **PizzaBase**”

- “Every individual of the **Pizza** class must have at least one base from the class **PizzaBase**”

What does this also mean?

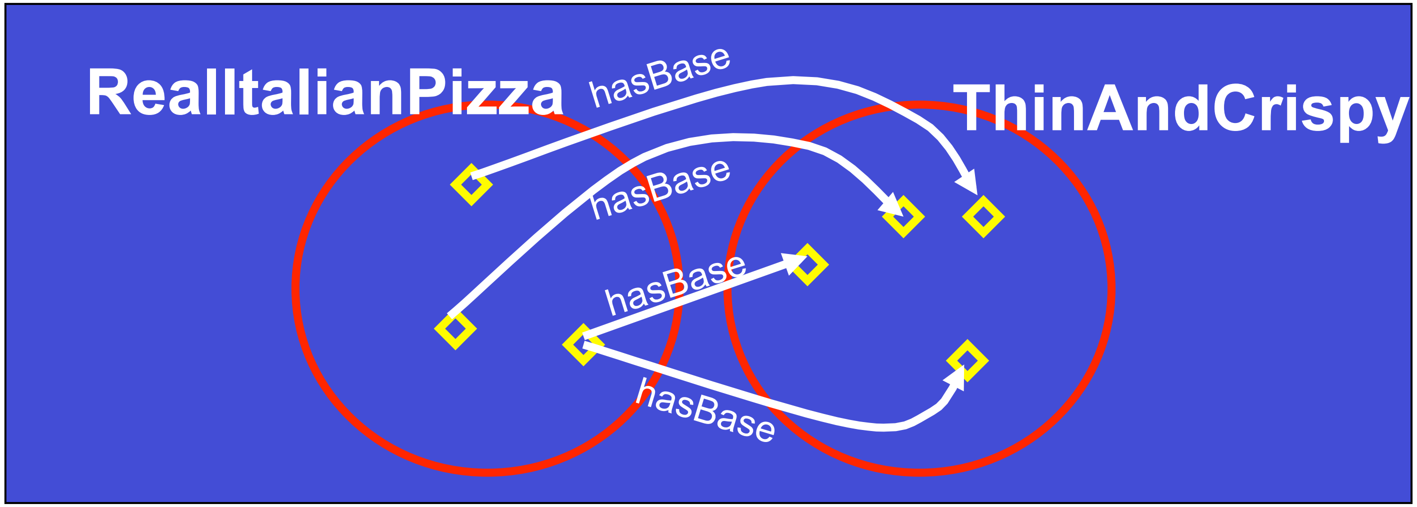
- We have created a restriction: ∃hasBase.**PizzaBase** on Class **Pizza** as a necessary condition



- “There can be no individual, that is a member of this class, that does not have at least one hasBase relationship with an individual from the class **PizzaBase**”

What does this mean?

- We have created a restriction: ∀hasBase.**ThinAndCrispy** on Class **RealItalianPizza** as a necessary condition



- "If an individual is an instance of **RealItalianPizza** , it is necessary that it must only have hasBase relationships with individuals from the class **ThinAndCrispy**"

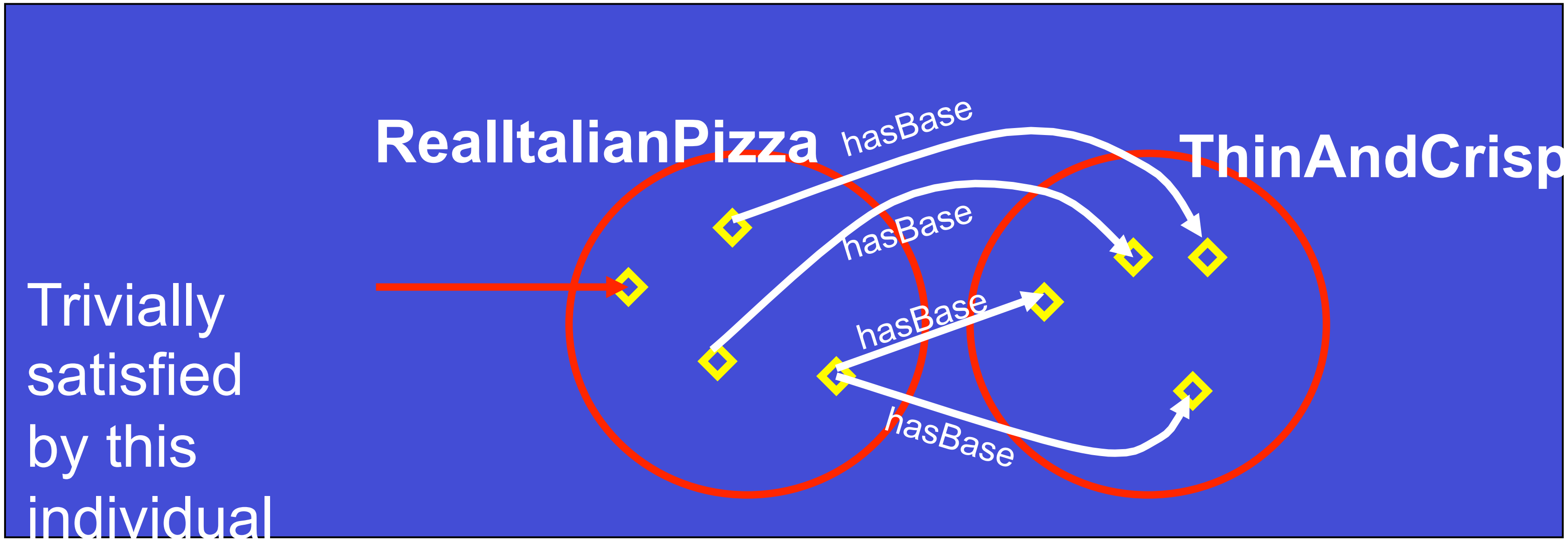
What does this mean?

- We have created a restriction: ∀hasBase.**ThinAndCrispy** on Class **RealItalianPizza** as a necessary condition

- “No individual of the **RealItalianPizza** class can have a base from a class disjoint with **ThinAndCrispy**”

Universal Warning – Trivial Satisfaction

- We have created a restriction: ∀hasBase.**ThinAndCrispy** on Class **RealItalianPizza** as a necessary condition



- "If an individual is a member of this class, it is necessary that it must only have hasBase relationships with individuals from the class **ThinAndCrispy, or no hasBase relationship at all**"
- ie Universal Restrictions by themselves do not state "at least one"

Lecture Outline

- Motivation
- A brief introduction of OWL
- From Competency Questions to OWL Ontologies
- Practical

Basic Notion: Competency Question

- A typical CQ: Which pizza has some cheese topping?

- CQs are questions that people expect the constructed ontologies to answer
- Usually used to determine the scope and granularity of the ontology
  - What domain elements I want to talk about?
  - How are they related to each other in the ontology?
- CQs are particularly useful for novice ontology authors:
  - in natural languages
  - about domain knowledge
  - requires little understanding of ontology technologies

Basic Notion: Competency Question

- A typical CQ: Which pizza has some cheese topping?

- Answers of competency question can have different interpretations

- Answer: empty set

- Possible scenarios
  - Pizza does not exist
  - Cheese topping does not exist
  - Pizzas are not allowed to have cheese topping
  - The ontology has not been populated with any cheesy pizza yet
  - …

UNIVERSITY OF ABERDEEN

Basic Notion: Competency Question

- A typical CQ: Which pizza has some cheese topping?
- When building an ontology, it is less important what the answer is
  - The ontology can be extended and modified so the answer is not final
- More importantly is whether the CQ can be answered meaningfully
  - The ability to ***answer CQs meaningfully*** can be regarded as a functional requirement of the ontology

Basic Notion: Presupposition

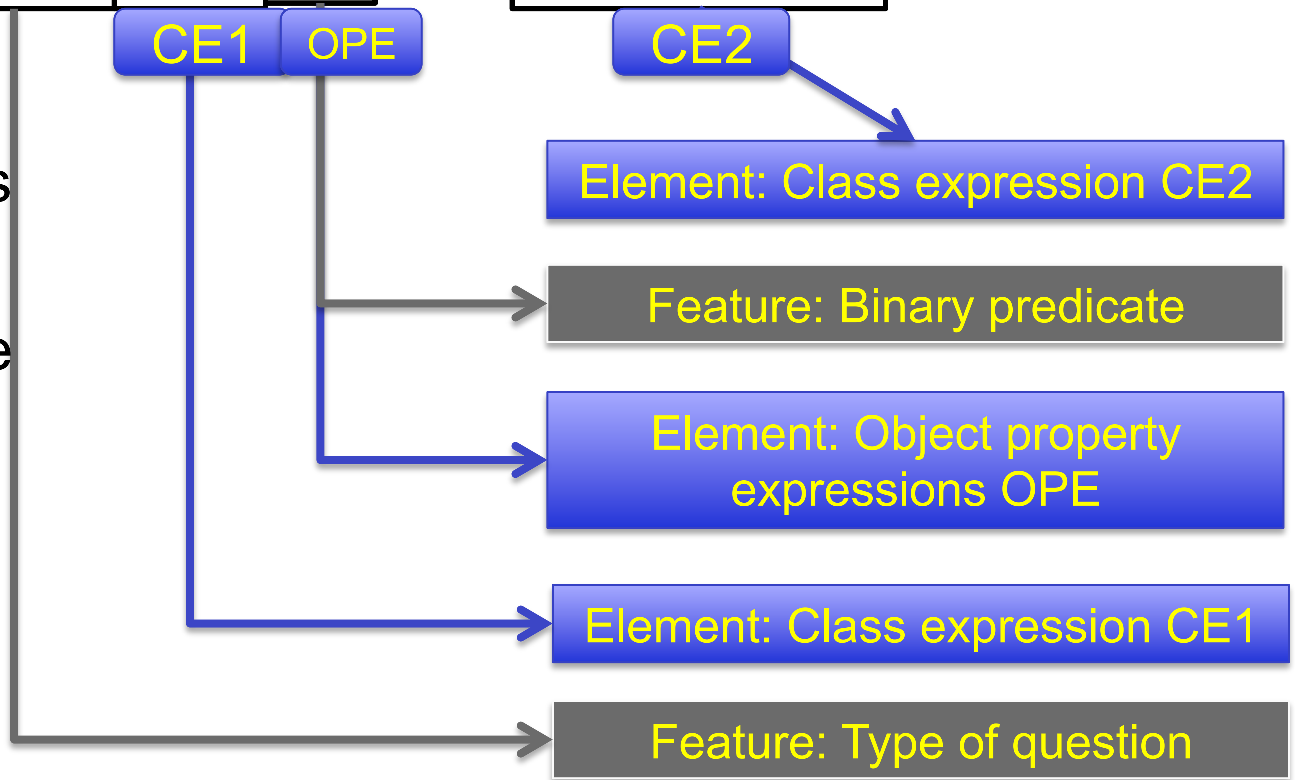
- A typical CQ: Which pizza has some cheese topping?

- A CQ comes with certain *presuppositions*
  - *Some conditions assumed to be met by the speakers*
- A CQ can be *meaningfully answered* only when its presuppositions are satisfied

- Classes *Pizza*, *CheeseTopping* should occur in the ontology
- Property *has(Topping)* should occur in the ontology
- The ontology should *allow Pizza to have CheeseTopping*
- …

Basic Notion: CQ Feature and Elements

- A typical CQ: Which pizza has some cheese topping?



- Real-world CQs usually have clear and relatively simple syntactic patterns

- Features and elements of CQs can be extracted

Basic Notion: Authoring Test

- A typical CQ: Which pizza has some cheese topping?

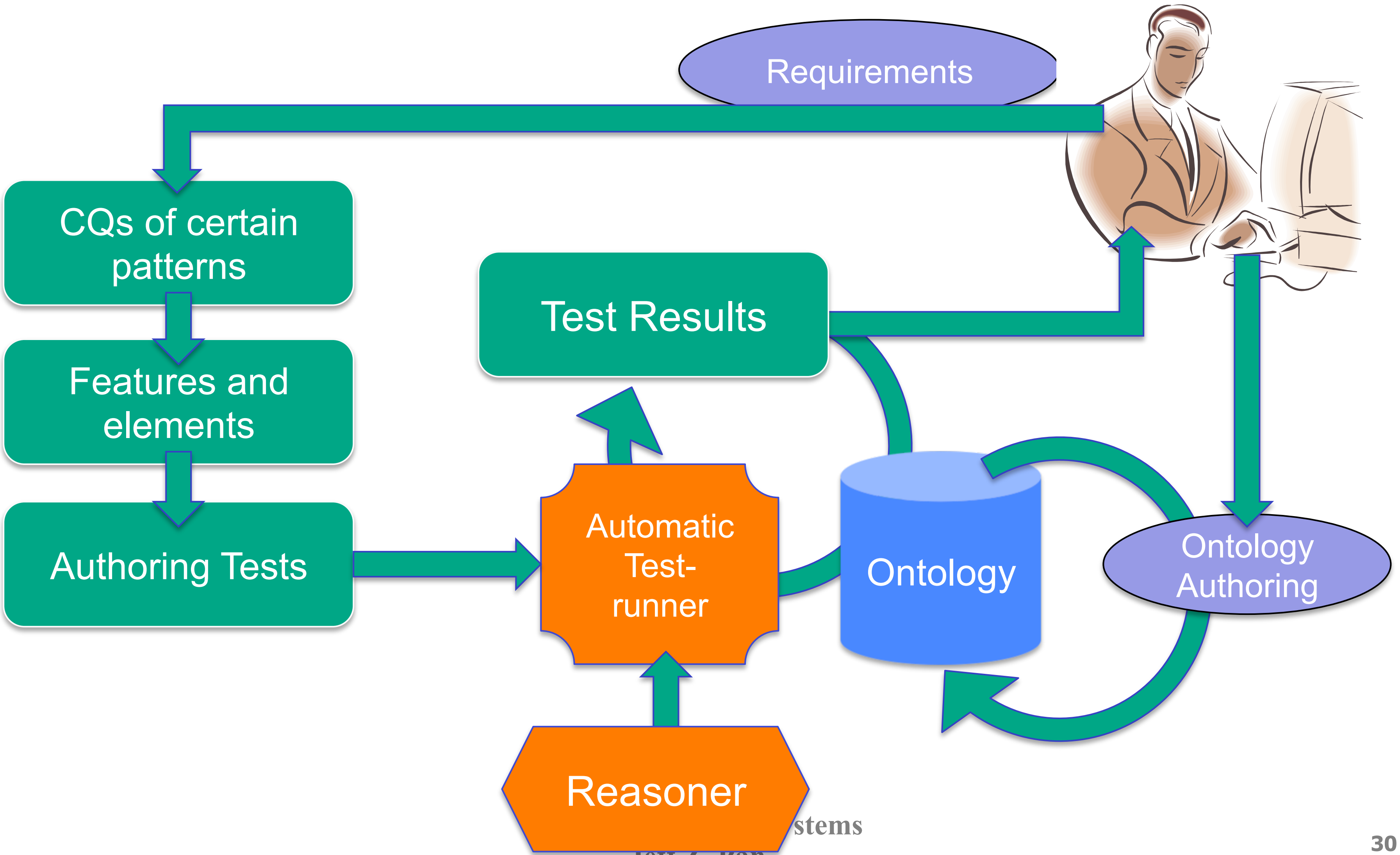
CE1 OPE CE2

- From features and elements of a CQ, a set of **Authoring Tests** can be generated

- Satisfiability of CQ presuppositions can be verified by testing the authoring tests

- Classes *Pizza*, *CheeseTopping* should occur in the ontology
  - [CE1], [CE2] should both ***occur*** in the class vocabulary
- Property *has(Topping)* should occur in the ontology
  - [OPE] should ***occur*** in the property vocabulry
- The ontology should *allow Pizza to have CheeseTopping*
  - CE1⊓ ∃OPE.CE2. should be ***satisfiable***
- …

A Competency Question-driven Ontology Authoring Pipeline



Putting Everything Together

- Example: What is the best software to read this data?
- Elements and features:
  - Class expressions
    - Software
    - Data
  - Object property
    - Read (but will be modelled in a different way)
  - Implicit datatype property
    - Has-performance
  - A 3-ary predicate between software, data and reading performance
  - A numeric modifer
    - “the best” on the reading performance

Putting Everything Together

- Example: What is the best software to read this data?
- Authoring Tests
  - Software, Data, Reading are instances of OWL:Class
  - hasSoftware, hasData are instances of OWL:ObjectProperty
  - hasPerformance is an instance of OWL:DatatypeProperty
  - Reading should be allowed to have software, data and performance
  - Reading performance should be allowed to have multiple different values for different readings
  - hasPerformance, hasSoftware, hasData should be functional for Reading

Putting Everything Together

- Example: What is the best software to read this data?
- Presuppositions:
  - Software, Data, Reading should occur as classes
    - Reading is the reification of the 3-ary predicate
  - hasSoftware, hasData should occur as object properties
  - hasPerformance should occur as a datatype property
  - , , should be satisfiable
  - The range of should be some numeric datatype
  - , should be satisfiable
    - There does not exist a performance value that all readings must have such a value
  - , ,

Summary

- A quick introduction of OWL

- Many real-world CQs can be described with a feature-based framework

- How can we automatically test whether a CQ can be meaningfully answered?
- The presuppositions of CQs can be identified based on the features and parameterised, tested with automatic authoring tests