Formalization and implementation of BFO 2 with a focus on the OWL implementation

With an introduction to some of the underlying technologies

Alan Ruttenberg, University at Buffalo, 2012-08-18

Goals of this session

- Discuss the *implementation* of BFO 2 in a computational setting, in particular the *Semantic Web*
- Explain the role of the BFO2 *Reference*, informal *first order logic* (FOL), *FOL implementation*, and *OWL implementation*
- Give an introduction to OWL 2, the Web Ontology Language
  - Show what constructs are available
  - Explain the role of *reasoner* and standard reasoning tasks
- Go into detail about the BFO 2 OWL 2 implementation
  - Review design, classes, properties
  - Explain approach to temporal representation
  - Review open issues
- Give some pointers to additional resources

How is BFO made

- Historically, Barry Smith developed BFO
- Because BFO will be part of the OBO Foundry, it will now be developed collaboratively
- There is a group of about 10 individuals who have actively worked on the current version
- The output of this process are: The BFO2 *Reference*, informal *first order logic* (FOL), *FOL implementation*, and *OWL 2 implementation*
- There is overlap of the people involved in developing each of these artifacts.
- It should be understood that BFO is not yet finalized
- You should understand phrases such as "BFO has X, or BFO doesn't X", until this development cycle of BFO is finished, as "in my opinion BFO has X" or "the current BFO document has X"

Role of BFO 2 development artifacts

- The Reference tells you *how to think* about BFO 2, as well as explicating the skeleton of the formal expression
- The *informal FOL*, part of the Reference, is a guideline for developing the FOL expression of BFO
- The *FOL expression* of BFO *should* be the arbiter of decision making regarding which statements are true according to BFO
- The *OWL 2 implementation* of BFO 2 is mean to be *used* for computing with knowledge structured according to the principles developed in BFO 2 Reference

Example of informal FOL – **realizes** relation

- Reference: To say that *b* is a *realizable entity* is to say that *b* is a *specifically dependent continuant* that inheres in some *independent continuant* which is not a *spatial region* and is of a type instances of which are **realized** in *processes* of a correlated type.
- First order logic(ish): to say that b **realizes** c **at t** is to assert that there is some material entity d & b is a process which **has participant** d **at t** & c is a disposition or role of which d **is bearer_of at t** & the type instantiated by b is correlated with the type instantiated by c.

Example of FOL – realizes relation

```
(forall (x y t)
    (if (and (hasParticipantAt x y t)
             (SpecificallyDependentContinuant y))
        (exists (z)
            (and (IndependentContinuant z)
                 (not (SpatialRegion z))
                 (specificallyDependsOnAt x z t)
                 (specificallyDependsOnAt y z t)))))
```

FOL is written as documentation, as well as in a computational format called Common Logic Interchange Format (CLIF)

Example of OWL – realizes relation

- Domain: process
- Range: ‘realizable entity’
- realizes o 'inheres in at all times’ -> ‘has participant at some time’

The owl is sometimes simpler that FOL in some ways, and more complicated in other ways. Strictly speaking OWL is a fragment of FOL, with a syntax that differs from standard FOL syntax

An introduction to Web Ontology Language *aka* OWL

Because it might be hard to explain the BFO 2 OWL implementation if you don't know what OWL is

Goals of this section

- Introduce the idea of the *semantic web* of which OWL technology is a part
- Explain what OWL is intended to do, and not
- Introduce you to the elements that OWL lets you work with
- Explain the role of *reasoner* and standard reasoning tasks
- Expose some common sources of error
- Show some examples of uses of OWL
- Give pointers to related tools and online resources

The semantic web in a nutshell

- *Adds to* web standards and practices encouraging
  - Unambiguous names for things, classes, and relationships
  - Well organized and documented in ontologies
  - With data expressed using uniform knowledge representation languages
  - To enable computationally assisted exploitation of information
  - That can be easily integrated from different sources
  - Both within and across public and organizational boundaries

Some elements of a future web hoped for by semantic web researchers

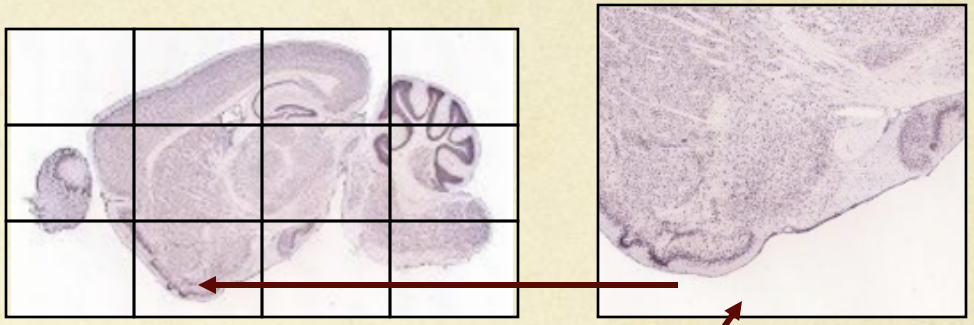
- A web where you can make precise scientific queries on a scale that google now makes available less precise queries
- A web with autonomous agents that you can trust to do tasks for you
- The same cut and paste fluidity with which we can build web pages, but for scientific analysis and visualization
- Increased scientific productivity because of the availability of knowledge to people with many different skills

Allen Brain Atlas & copy/paste

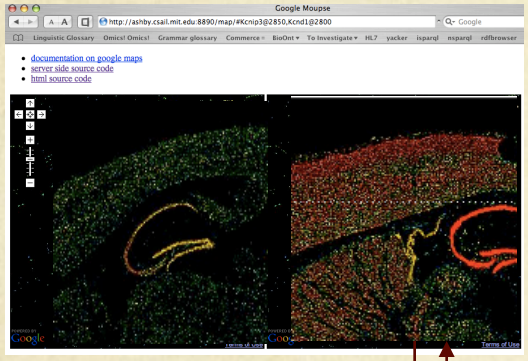
- A remarkable scientific achievement. Mouse brains sectioned and stained for the presence of gene expression.
- 20,000 genes, 400000 images at high resolution.
- At the time available only through an HTML interface.
- Extract the information, convert to RDF (Now there is XML) to enable search by ontology tools

Copy/Paste on the Semantic Web for Science

Allen Brain Institute Servers



Javascript



http://www.brainmap.org://....0205032816_B.aff/TileGroup3/1-0-1.jpg

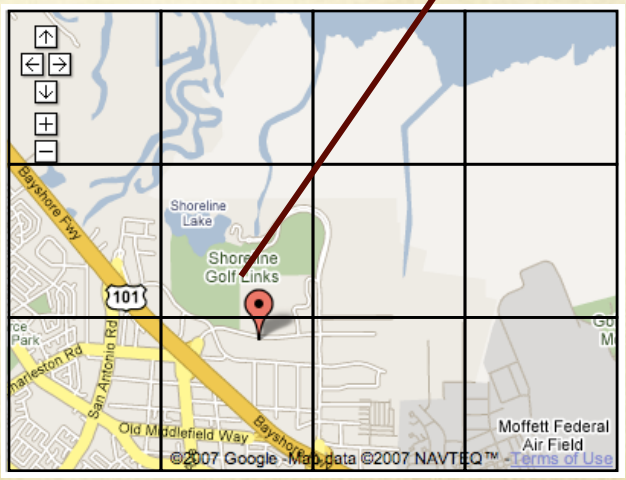
SPARQL
AJAX

Query

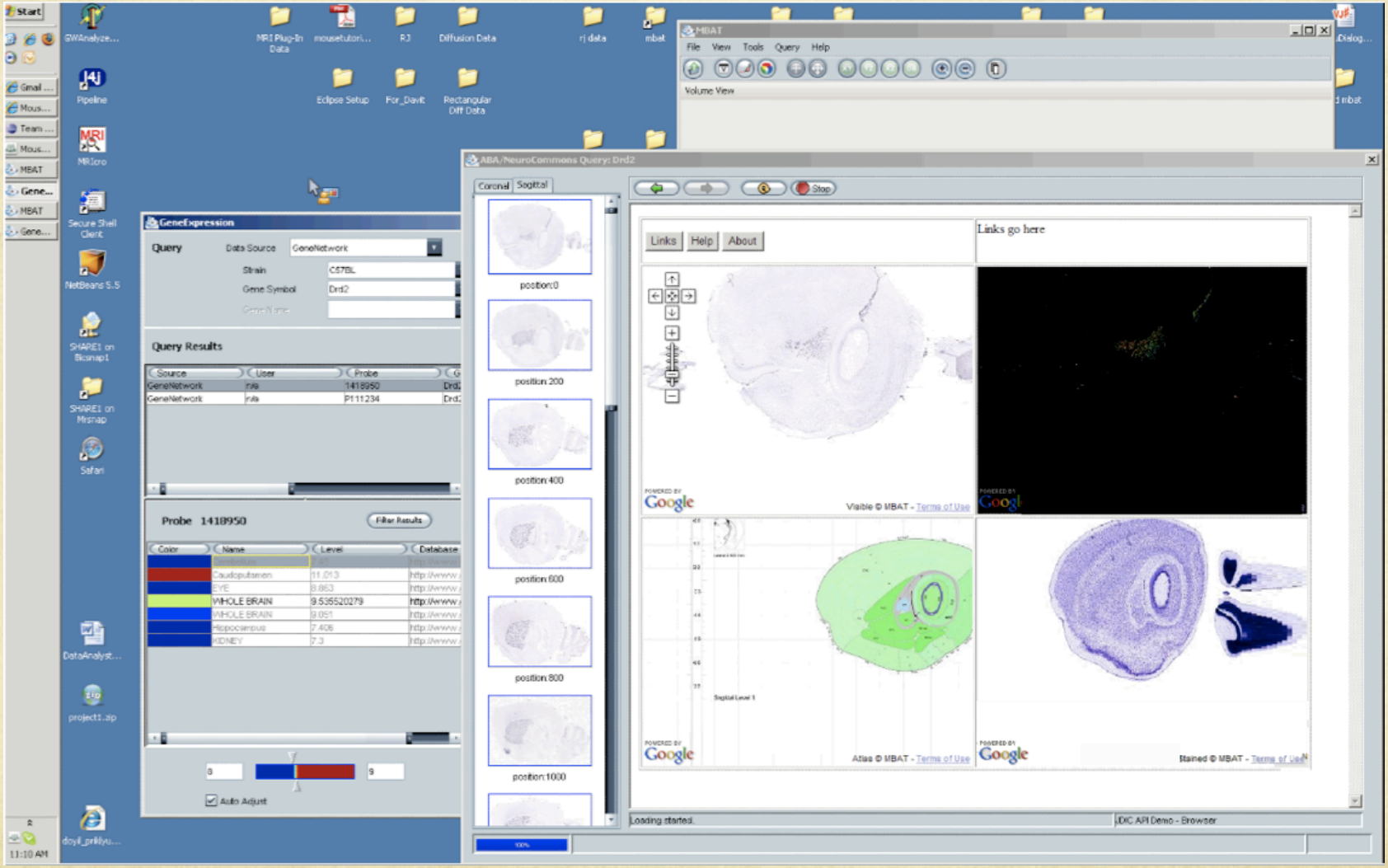
URL

Neurocommons Servers

Google
Maps
API



BIRN: “viewed source”, use our code in their own applications



We are always asking questions

- For what neurological disorders are cell lines available?
- For Parkinsons disease, what tissue and cell lines are available?
- Give me information on the receptors and channels expressed in cortical neurons
- What chemical agents can be used visualizing the nervous system?

A question I was asked

Create a system that will let us prioritize an expected 2000 siRNA hits according to whether there is chemical matter for studying them, e.g. validated antibodies, since we can only follow up on 600.

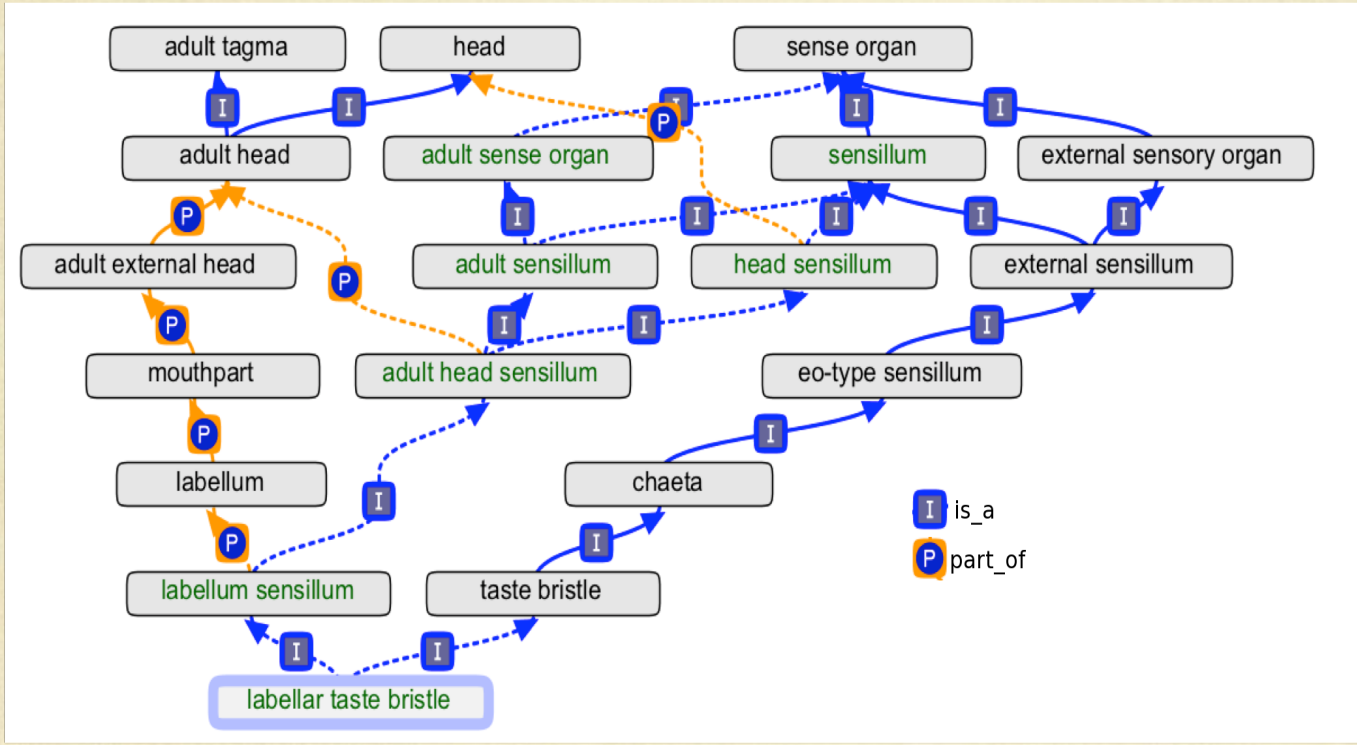
We know how to use Semantic Web technology to answer these kinds of questions (but there is no free lunch)

We build BFO 2 and ontologies based on it to make it possible to answer questions such as this

What is OWL?

- *Web Ontology Language* (acronym inspired by Winnie the Pooh)
- A computer language in which you can *express*, *communicate about* and *reason* over some kinds of ontologies
- One element in a set of interoperable standards and protocols collectively known as "*semantic web*"
- One of a set of such semantic web languages with varying capabilities – *RDF*, *RDFS*, *RIF*
- A *family* of languages – OWL-[Full, DL, EL, RL, QL]. In this presentation we focus on OWL-DL
- A fragment of first (and a bit of second) order logic

Manually maintaining an ontology with multiple *classification schemes* is hard...



Here the desire is to classify by part, by function, and by stage of life

...so automate what you can.

How OWL helps us move in that direction

- Historically, much work done in ontology and representation, but with many different, incompatible, systems
  - OWL standardization helps make it easier to share, deploy, and build on others work.
- In many cases these projects did not use very much reasoning, or used imprecise rules of inference
  - OWL has a clear semantics and there are and open source reasoners that perform adequately and yield complete, correct results
- Even now, a major repository of biomedical ontologies, the NCBO Bioportal, contains a number of inconsistent ontologies

What is OWL not?

- OWL is not a philosophical theory, it is a computational tool
  - Don't confuse OWL Classes with Universals
  - Don't confuse OWL Individuals with Particulars

  (even though we often use the former to represent the latter)
- Using OWL doesn't mean your ontology will automatically be good, any more than using words means you will write a good story
- There are lots of things that OWL's reasoning can't do, some very important, like reasoning about how events relate in time.

Building blocks of OWL

Building block: Individual

- Individuals are singular entities.
- They can be the subject and object of relations
- They can be named or un-named
- Often, but not always, OWL individuals are used to represent particulars.
- In subsequent text I will sometimes use a blue color for text descriptions of individuals

Building block: Class

- A *class* is a set of individuals, but is not determined by them.
- *Class expressions* describe conditions under which an individual is a member of that set
- OWL Classes are often used to represent *universals*, *groups*, or *attributed classes*.
- In subsequent text I will sometimes use a **green** or **orange** for text descriptions of classes

Building block: Datatype and Values

- Data values are singular entities.
- They can be the object of relations, but not subject
- Datatypes are sets of data values along with built-in relations between the values
- Every data value is associated with a datatype
- Example values: 2.7^^float, 3^^integer, “hello”
- Example built-in relations: > (for numbers), *length* (for strings)

Building block: Property

- Properties can be thought of as sets of pairings of individuals with either other individuals or data values. As with classes, the set does not determine the property.
- Often, but not always, OWL properties are used to represent relations.
- In subsequent text I will sometimes use **bold face** for text descriptions of properties
- Example: An OWL property **inheres_in** (at a time)
  - A set of pairs of dependent continuants and the independent continuants they inhere in
  - e.g (this dark blue color, this screen)

Building blocks: statements about properties

- Property *domain* and *range*, let you express what the class of any subject or object of a property is
- Property characteristics: transitive, symmetric, reflexive, functional …
  - Cause there to be additional inferences to be made based on a property assertion

Building block: Quantified Statements

- subClassOf, subPropertyOf
- All-some
  - ALL things related to SOME thing of a type.
  - Person: **has part** *some* head
- All-only
  - ALL things related to ONLY things of a type.
  - PumpingFunction: **realized in** *only* Pumping
- Composition of properties.
  - realizes o inheres_in => has_participant
  - English: for ANY function and ANY processs, if there EXISTs a thing such that function of *thing* is realized in process, THEN *thing* participates in process.

Different ways of writing OWL

Encoding the statement “John is an instance of Person”

DL Syntax

John ∈ Person

RDF/XML

```
<owl:Class rdf:about="http://example.org/1/person">
  <rdfs:label>person</rdfs:label>
</owl:Class>
<owl:NamedIndividual rdf:about="http://example.org/1/john">
  <rdf:type rdf:resource="http://example.org/1/person"/>
  <rdfs:label>john</rdfs:label>
</owl:NamedIndividual>
```

Functional style syntax

```
Prefix(:=<http://example.org/1/>)
Declaration(Class(:person))
AnnotationAssertion(rdfs:label :person "person")
Declaration(NamedIndividual(:john))
AnnotationAssertion(rdfs:label :john "john")
ClassAssertion(:person :john))
```

Manchester syntax

```
Prefix: : <http://example.org/1/>
Class: :person
    Annotations:
        rdfs:label "person”@
Individual: :john
    Annotations:
        rdfs:label "john"@
    Types:
        :person
```

Turtle

```
:john rdf:type :person
:john rdfs:label “john”
:person rdf:type owl:Class
:person rdfs:label “person”
```

Jargon: triples

```
:john rdf:type :person
:john rdfs:label “john”
:person rdf:type owl:Class
:person rdfs:label “person”
```

- Each line has three parts
  - Subject
  - Predicate
  - Object
- This representation originates with RDF

Reasoning and OWL

What is reasoning?

- Reasoning is a mechanism to leverage what you have said in OWL to do more for you
- Typical OWL reasoners perform several tasks
  - Check whether everything you said makes sense together
    - Are all the *classes satisfiable?*
    - Is the ontology *consistent?*
  - Determing which classes *subsume* each other
  - Infer relationships that are implied, but not stated explicitly
  - Determine what types an instance is
  - Answer a variety of queries, based on what you have stated
- Protégé provides access to *Pellet, Hermit and FaCT++*
- Beware: Reasoners are built by people and therefore sometimes have bugs.

Reasoning: a subsumption

- Given
  - An antihypertensive drug is any drug that treats hypertension
  - A calcium channel blocking antihypertensive is any drug that treats hypertension **AND** has a calcium channel blocker as active ingredient
- An owl reasoner would infer that
  - antihypertensive drug *subsumes* calcium channel blocking antihypertensive
  - calcium channel blocking antihypertensive *is a subclass of* antihypertensive drug
  - *Every* calcium channel blocking antihypertensive *is an* antihypertensive drug

(These three statements mean the same thing)

Reasoning: an unsatisfiable class

- Given
  - Every antidepressant drug has **at least one side effect**
  - Dr. Palaasa's recommended drugs are any antidepressant drug that **has no side effects**
- An OWL reasoner would infer that
  - The class Dr. Palaasa's recommended drugs is *unsatisfiable*
  - The class Dr. Palaasa's recommended drugs *can not have any members*

(These three statements mean the same thing)

Reasoning: an inconsistency

- Given
  - John was only treated with analgesics
  - John was treated with some coumarin
  - No coumarin is an analgesic (or coumarin and analgesic are *disjoint*)
- An OWL reasoner would infer that
  - There is an *inconsistency*

(If john was only treated with analgesics and he was treated with coumarin, then coumarin is an analgesic. But we said coumarin is not an analgesic)

Reasoning: determing type

- Given
  - John was treated with some coumarin
  - Anyone who is treated with something is a patient
- An OWL reasoner would infer that
  - John *is an instance of* patient
  - John *has type* patient

(These mean the same thing)

Reasoning: an inferred relation

- John's hand **is part of** John's body
- John's finger **is part of** John's hand
- The **is part of** relation is *transitive*
- An OWL reasoner would infer that
  - John's finger **is part of** John's body

  (even though you didn't explicitly say that)

Reasoning: answering queries

- OWL reasoners can answer queries posed as class descriptions, using previously described reasoning
  - Using subsumption, one can retrieve superclasses or subclasses of the described class
  - Using the ability to determine type one can retrieve instances that satisfy the class description

Reasoning: answering a query

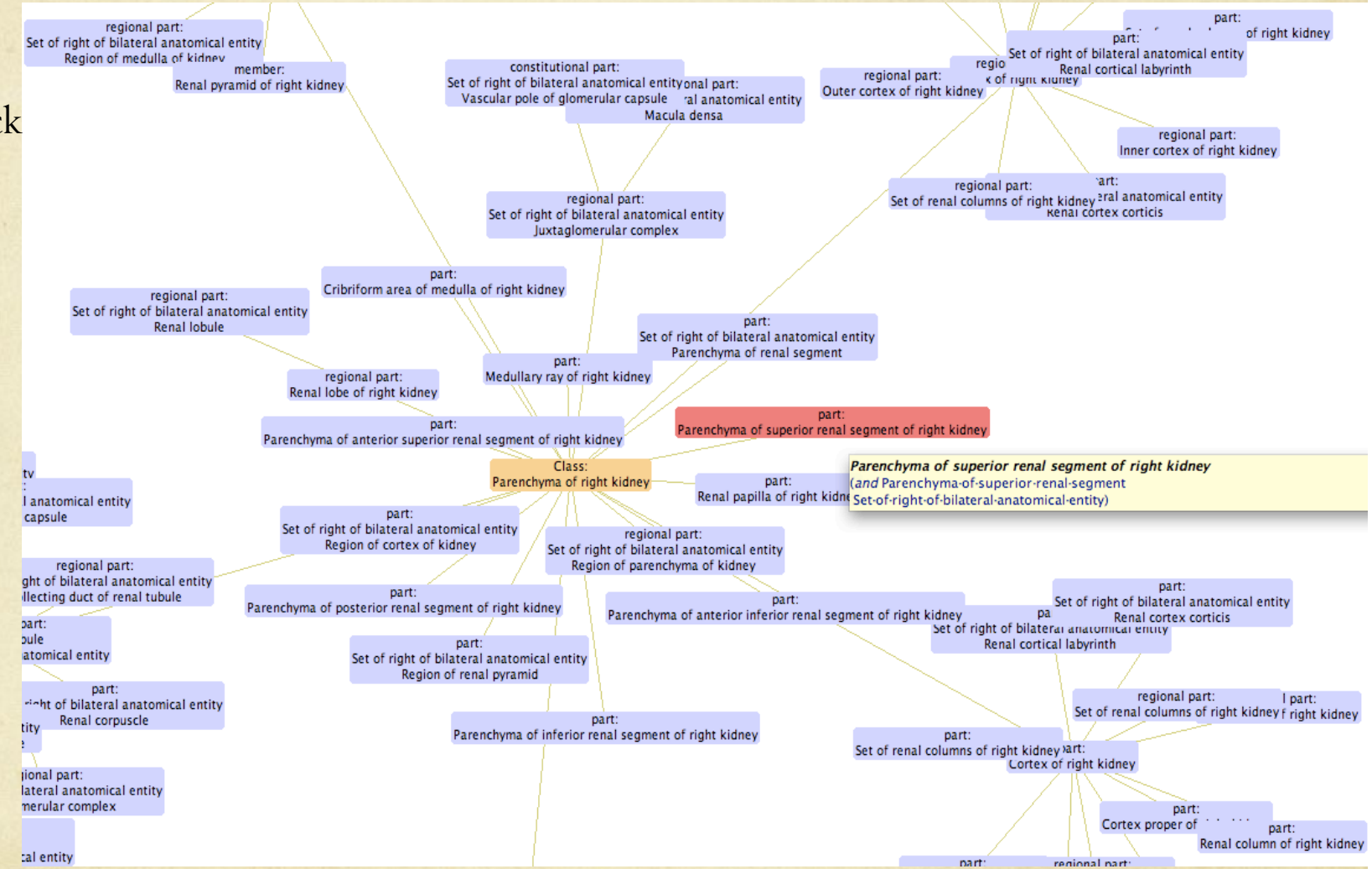
- Given
  - John has a temperature of 98.6
  - Mary has a temperature of 99.2
  - Sam has a temperature of 96.2
  - Elise has a temperature of 101.1
  - Suppose you want to query "which people have a temperature of 99 or above" you would create the class expression
    - Any thing that has a temperature of > 99 (imagine this class is called 'q')
  - Then ask which instances have type q. We describe that kind of reasoning on the slide "reasoning: determining type"

How does a reasoner work?

- Recall
  - Everything is grounded in individuals
  - Classes have *extensions* = the individuals that are members
  - Properties have *extensions* = the pairs of individuals that are so-related.
- OWL Reasoners work by trying to prove all *models* satisfy the constraints of what you have stated in your ontology. They do that by trying to find a model that satisfies the negation, and if they find it they are done.
- A models is (to a first approximation) a choice of these extensions

One visualization of a model

Be happy: reasoners check lots of details, saving you the effort!



Caveat: There are two important assumptions OWL reasoners make

The open world assumption

- What you don't say, isn't known
- Consider a drug formulary listing drugs whose cost will be reimbursed by insurance
  - If a drug is not listed in the formulary, then it won't be reimbursed.
  - This demonstrates the closed world assumption: If something isn't said, it is considered to be false
- Consider a taxonomy of species
  - If a species is not in the taxonomy, the author simply didn't know about it.
  - The open world assumption is well suited for working in science, where what we don't know is much more than what we know

Things can have more than one name

- We say OWL lacks the *unique name assumption*
- Consider the bar code on blood sample tube
  - Some bar codes are assigned with the intention that they are not reused
  - We can safely assume that if we are given two different bar codes, there are two different tubes
  - Therefore, we can use the *unique name assumption* when dealing with bar codes on blood samples
- Consider instead the different names for a chemical
  - Aside from different communities giving chemical different names, there are names for the same thing in many different languages
  - We can't assume that if two things have different names they are different things.
  - Because OWL was built for the world **wide** web, it **doesn't** make the unique name assumption.

Beware

- Forgetting about these two assumptions is a cause for much confusion about how OWL behaves
- Most commonly people are surprise because they expect different results than they get
- I will give examples of some cases of this

A typical open world surprise

- Given
  - John has son Mark
  - John has son Sam
- Describe the class
  - Person that has 2 sons
  - Surprise! – John is not determined to be an instance of this class
- Why?
  - John might have another son – we didn't say these were his only sons
  - In OWL you can say, if appropriate, that Mark and Sam are *different individuals*

Another typical open world surprise

- Given
  - The *domain* of property *has son* is an organism
  - my corolla is a car
  - my corolla **has son** my bicycle
- We've made a mistake!
  - Why doesn't the reasoner complain?
  - my corolla is inferred to be both a car and an organism, but we didn't say that nothing can be both a car and an organism
  - In OWL we need to say that car and organism are *disjoint classes*

A typical (lack of) unique name assumption surprise

- Given
  - John has son Mark
  - John has son Sam
- Create the class
  - Person that has at least 2 sons
  - Surprise – John is not determined to be an instance of this class
- Why?
  - Mark and Sam might be names for the same person. The fix: Assert that Mark and Sam are *different individuals*

OWL 2 added a number of capabilities to OWL

- Annotations everywhere
- No separated vocabulary
- New property types
- Qualified cardinality
- Property chain
- Datatypes
- Profiles

Annotations

- Annotations (e.g. string to display in user interface) have no model theoretic semantics
- OWL 2 lets you annotates more entities. OWL 1: Entities. Now also: Axioms, PropertyAssertions, Annotations themselves.

```
AnnotationAssertion( obo:OBI_0000255
      Annotation( obi:forCommunity "CDISC" )
      obi:alternativeLabel "Investigation Site")
PropertyDomain(
  Annotation(obi:editor_note "Needs more tightly
                              controlled domain restrictions")
  ro:part_of snap:Continuant)
```

No more separated vocabulary

- In OWL 1 DL, an name could not denote more than one of class, object property, datatype property, individual
- In OWL 2 DL this restriction is removed. One can say:

  ClassAssertion(*a:Dog a:Brian) Brian instance of dog.*

  ClassAssertion(*a:Species a:Dog) Dog instance of species.*

  However, inference is limited:

  SameIndividal(*a:Dog a:Canine) does not entail*

  ClassAssertion(*a:Canine a:Brian)*

Property types

- OWL 1 had: Functional, InverseFunctional, Transitive, Symmetric
- OWL 2 adds: Asymmetric, Reflexive, Irreflexive

Nobody can be their own spouse

ObjectProperty(spouseOf Irreflexive)

If A is B's parent, then B is not A's parent

ObjectProperty(biologicalParent ASymmetric)

Unfortunately, transitivity can't be combined with asymmetry so this can't be used to check partonomies for cycles.

Qualified Cardinality Restrictions

- In OWL 1 there were cardinality constraints that constrained the total number of values that a property could have. In OWL 2 we can qualify cardinality with a type.

A safe drug regimen must not contain more than one central nervous system depressant

ClassAssertion(
MaxCardinality(1 CNS-Depressant***)***
SafeDrugRegimen ***)***

Property chains

- Gives ability to compose relations.

Owning something means owning all of its parts

SubPropertyOf(PropertyChain(*owns has_part) owns)

Caveat: There are complex side conditions that rule out certain combinations of axioms

More expressive datatypes

```
Teenagers have an age between 13 and 19
SubClassOf (
  SomeValuesFrom( a:hasAge
       DatatypeRestriction( xsd:integer
            xsd:minInclusive "13"^^xsd:integer
           xsd:maxInclusive "19"^^xsd:integer)
     )    a:Teenager)
```

An experimental extension – linear equations. A set datatype properties are constrained to be solutions of a set of linear equations.

http://www.w3.org/2007/OWL/wiki/Data_Range_Extension:_Linear_Equations

Property chains

Property chains can be used to implement decidable rules on OWL:

$$
\begin{array}{ll}
(1) & \mathsf{NutAllergic}(x) \wedge \mathsf{NutProduct}(y) \rightarrow \mathsf{dislikes}(x, y) \\
(2) & \mathsf{Vegetarian}(x) \wedge \mathsf{FishProduct}(y) \rightarrow \mathsf{dislikes}(x, y) \\
(3) & \mathsf{orderedDish}(x, y) \wedge \mathsf{dislikes}(x, y) \rightarrow \mathsf{Unhappy}(x) \\
(4) & \mathsf{dislikes}(x, v) \wedge \mathsf{Dish}(y) \wedge \mathsf{contains}(y, v) \rightarrow \mathsf{dislikes}(x, y) \\
(5) & \mathsf{orderedDish}(x, y) \rightarrow \mathsf{Dish}(y) \\
(6) & \mathsf{ThaiCurry}(x) \rightarrow \mathsf{contains}(x, \mathsf{peanutOil}) \\
(7) & \mathsf{ThaiCurry}(x) \rightarrow \exists \mathsf{contains}.\mathsf{FishProduct}(x) \\
(8) & \rightarrow \mathsf{NutProduct}(\mathsf{peanutOil}) \\
(9) & \rightarrow \mathsf{NutAllergic}(\mathsf{sebastian}) \\
(10) & \rightarrow \exists \mathsf{orderedDish}.\mathsf{ThaiCurry}(\mathsf{sebastian}) \\
(11) & \rightarrow \mathsf{Vegetarian}(\mathsf{markus}) \\
(12) & \rightarrow \exists \mathsf{orderedDish}.\mathsf{ThaiCurry}(\mathsf{markus})
\end{array}
$$

http://korrekt.org/papers/KroetzschRudolphHitzler_ELP_TR_2008.pdf

More expressive datatypes

- OWL 1 had strings and integers at least, and poorly defined behavior for additional datatypes
- OWL 2 has a wider variety of types with well defined behavior: floats, doubles, dates, decimal, real, rational
- OWL 2 allows extension, by restriction, of datatypes, e.g. integers < 3, strings that match regular expresion “OBI_.*”.

Profiles

- Profiles are subsets of the language with different computational properties
- OWL EL – Drops disjunction, negation, universal quantification (+ some more). Gains polynomical classification time.
- OWL QL – Drops disjunction, universal quantification, enumerated classes. Restricts where existentials and negation can occur (+ some more). Gains ability to work with instance data stored natively in a relational database by translation of main operations to SQL

Profiles

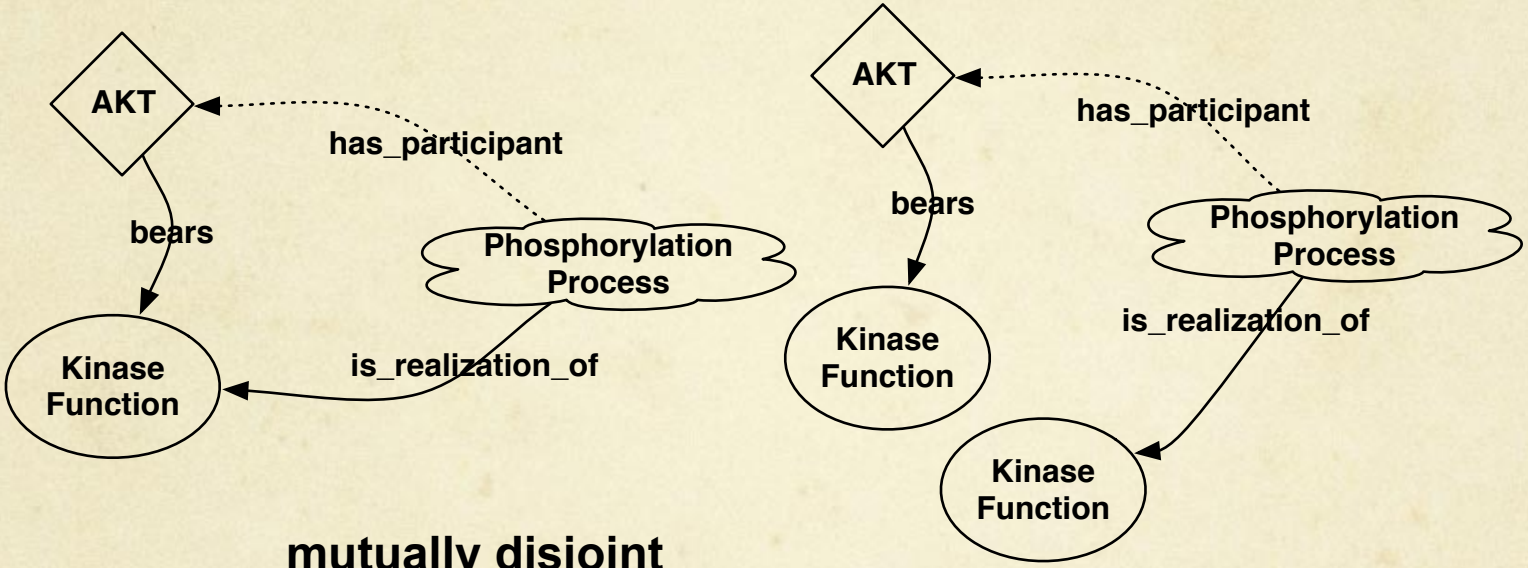
- OWL-RL: Restricts expressiveness so that individuals not explicitly in the ontology can not be inferred. Gains the ability to compute ground entailments (those involving only individuals) by using a forward chaining rule system (but caveat datatype).
- The rules applied to unconstrained RDF graphs give correct, but incomplete, results.

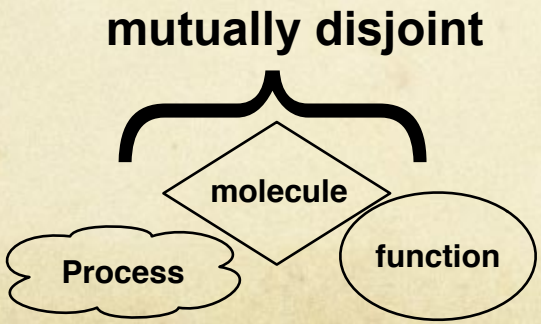
Some things OWL can't compute

- Cycles
- Temporal Reasoning
- Query (other than querying instances of a class)

Cycles

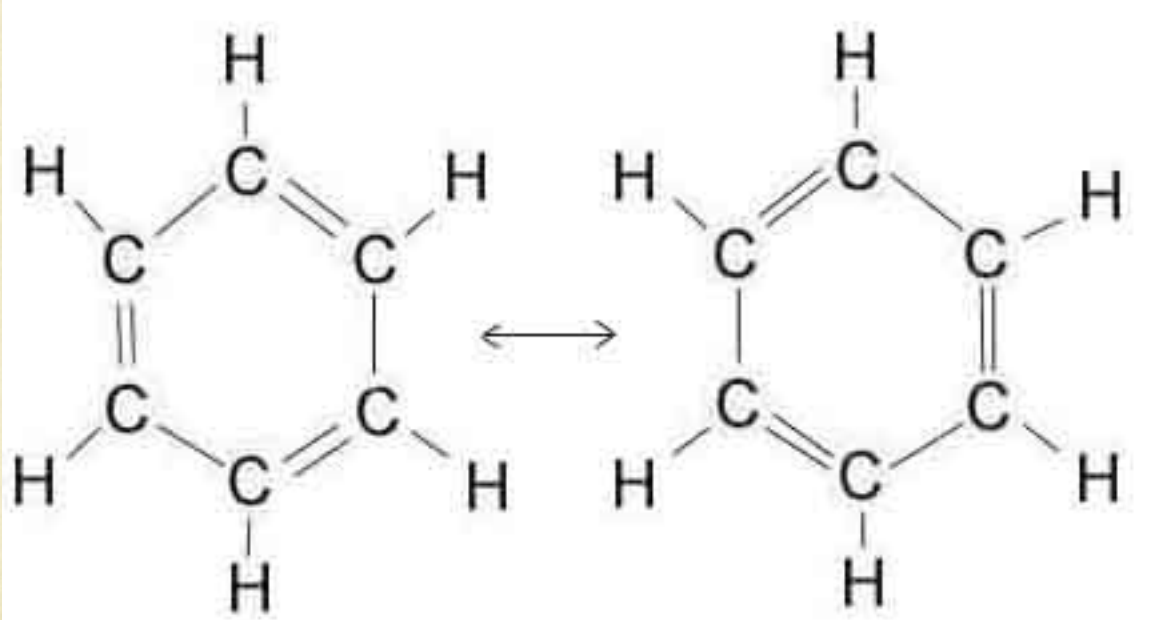
Can't express that there are only 3 entities connected in a cycle. Can't differentially classify cycle from open.





Cycles

Can't write a defined class expression that classifies these as rings



Temporal Expression

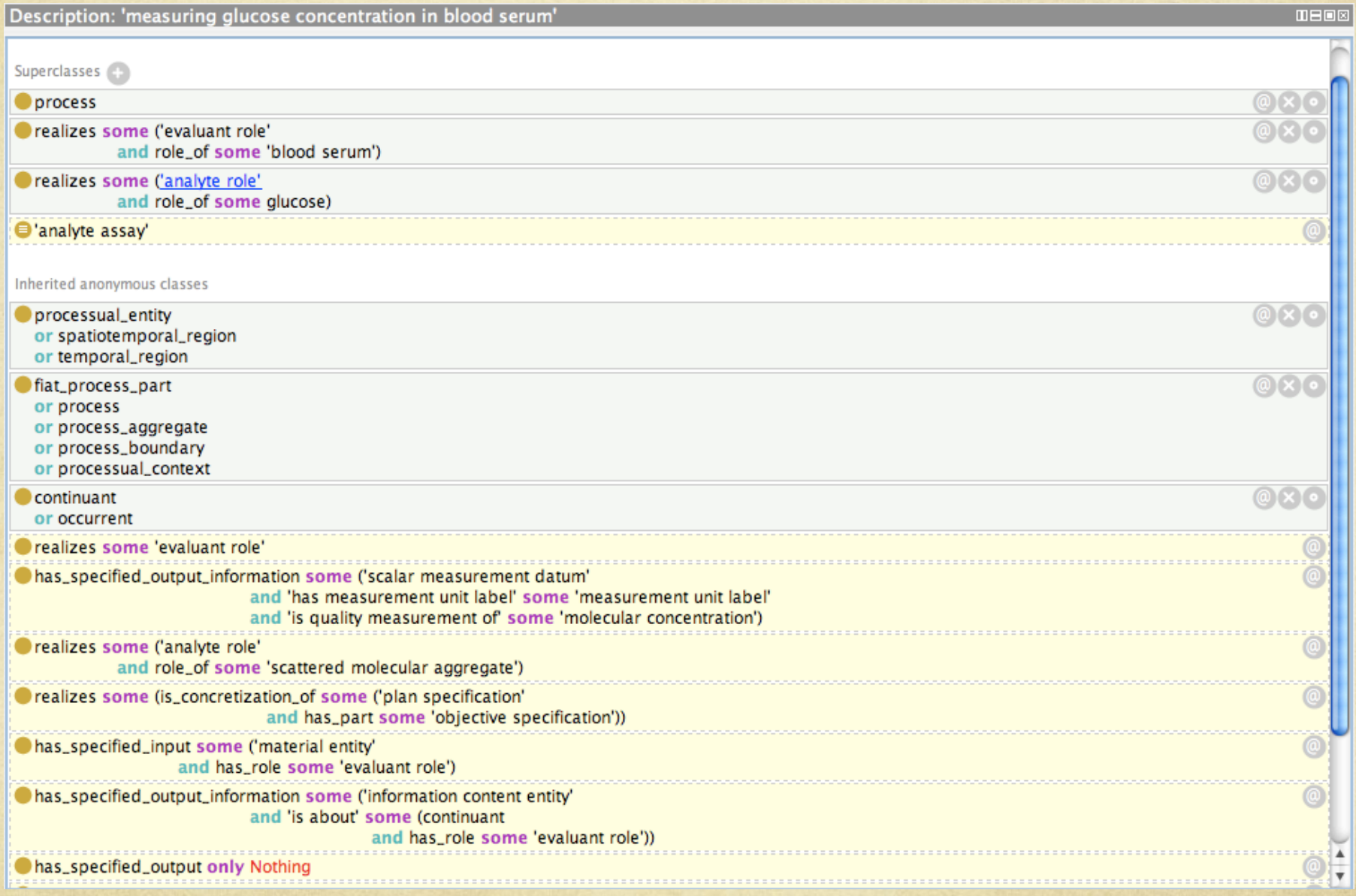
- We can express time in many ways. For example, using annotations.
- PropertyAssertion(
  Annotation(during interval1)
  part_of A B)
- But the annotation carries no semantics. If we have
- PropertyAssertion(
  Annotation(during interval2)
  part_of B C)
- Reasoner will conclude that A part_of C even if interval1 and interval2 don't overlap

Temporal Reasoning

What do we want to say? (relevant for biology)

1) something exists only during an interval

2) If A exists between t1, t2, and B exists between t3, t4, and we say there is some time that A part_of B, can we find an inconsistency if {t1, t2} doesn't overlap {t3,t4}?


Description: 'measuring glucose concentration in blood serum'
Superclasses
process
realizes some ('evaluant role' and role_of some 'blood serum')
realizes some ('analyte role' and role_of some glucose)
'analyte assay'
Inherited anonymous classes
processual_entity or spatiotemporal_region or temporal_region
fiat_process_part or process or process_aggregate or process_boundary or processual_context
continuant or occurrent
realizes some 'evaluant role'
has_specified_output_information some ('scalar measurement datum' and 'has measurement unit label' some 'measurement unit label' and 'is quality measurement of' some 'molecular concentration')
realizes some ('analyte role' and role_of some 'scattered molecular aggregate')
realizes some (is_concretization_of some ('plan specification' and has_part some 'objective specification'))
has_specified_input some ('material entity' and has_role some 'evaluant role')
has_specified_output_information some ('information content entity' and 'is about' some (continuant and has_role some 'evaluant role'))
has_specified_output only Nothing

From English to OWL: Words mix up functions and objects

- Ligand
- Neurotransmitter
- Hormone
- Peptide

Looking for peptides?

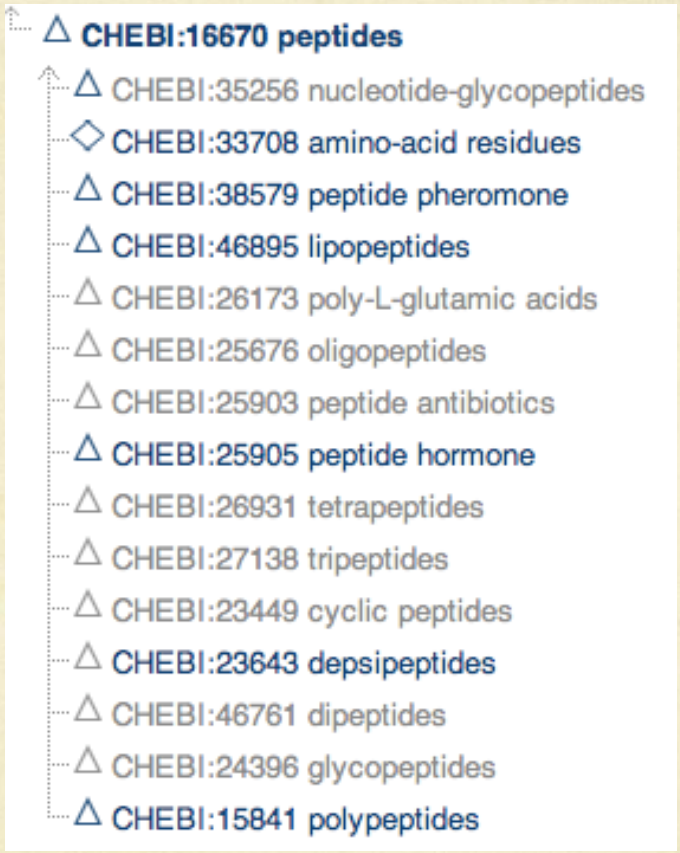
Connect words to their corresponding entities in the world

- *PeptideReceptorLigand* - **A peptide** that has a function which makes it able to bind to a receptor
- *PeptideNeurotransmitter* - **A peptide** expressed in a neuron that has a function which makes it able to regulate another neuron
- *PeptideHormone* - **A peptide** that produced in one organ and having an regulatory effect in another.
- *Peptide* - A "short" polymer of amino acids

Looking for peptides?

Peptides from CHEBI

Chemical Entities of Biological Interest



Hormone Activity from GO Molecular Function

- all : all [439998]
  - GO:0003674 : molecular_function [286402]
    - GO:0005488 : binding [86988]
      - GO:0005515 : protein binding [42794]
        - GO:0005102 : receptor binding [3784]
          - **GO:0005179 : hormone activity [604]**
            - GO:0017045 : adrenocorticotropin-releasing hormone activity [2]
            - GO:0017044 : alpha-melanocyte stimulating hormone activity [0]
            - GO:0046659 : digestive hormone activity [0]
            - GO:0008613 : diuretic hormone activity [12]
            - GO:0016913 : follicle-stimulating hormone activity [4]
            - GO:0016608 : growth hormone-releasing hormone activity [16]
            - GO:0005183 : luteinizing hormone-releasing factor activity [22]
            - GO:0030354 : melanin-concentrating hormone activity [4]
            - GO:0016085 : myoinhibitory hormone activity [2]
            - GO:0016084 : myostimulatory hormone activity [2]
            - GO:0005184 : neuropeptide hormone activity [222]
              - **GO:0016087 : ecdysiostatic hormone activity [2]**
            - GO:0016521 : pituitary adenylate cyclase activating polypeptide activity [0]
            - GO:0008437 : thyrotropin-releasing hormone activity [6]

hormone activity

Term information ⬇ Term lineage ⬇ External references ⬇ Term associations ➡

Term Information

Accession GO:0005179

Ontology molecular function

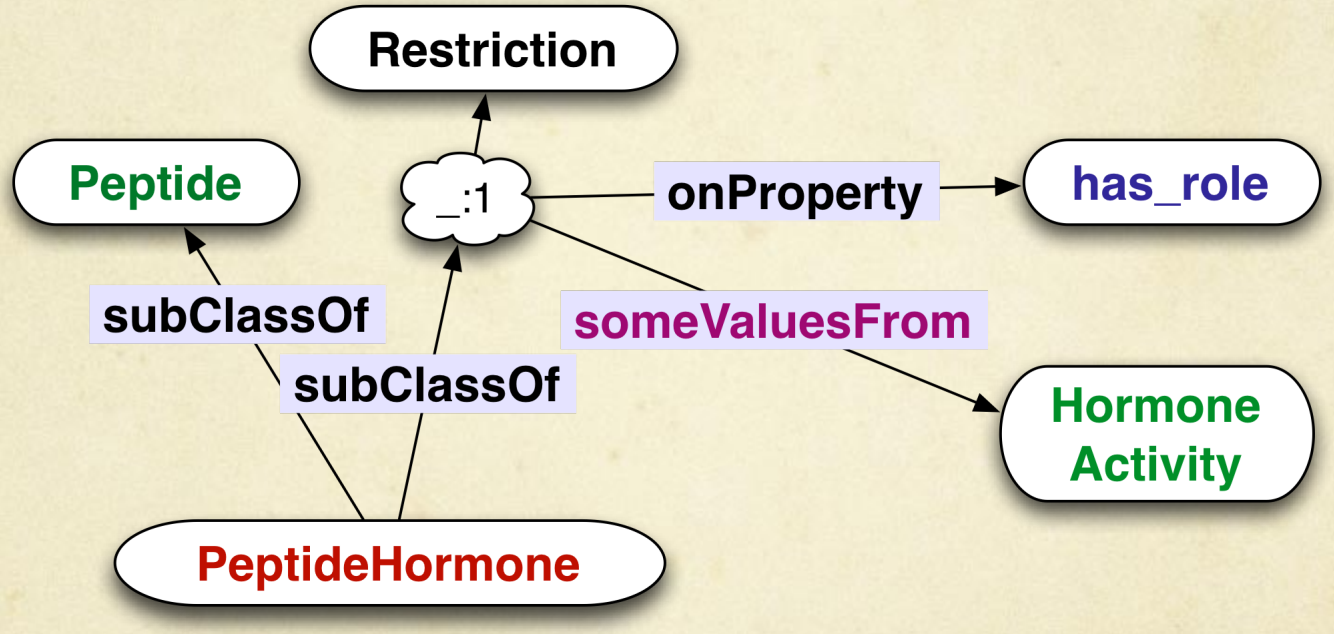
Synonyms
narrow: cAMP generating peptide activity
narrow: glycopeptide hormone
narrow: lipopeptide hormone
narrow: peptide hormone

Definition The action characteristic of a hormone, any substance formed in very small amounts in one specialized organ or group of cells and carried (sometimes in the bloodstream) to another organ or group of cells in the same organism, upon which it has a specific regulatory action. The term was originally applied to agents with a stimulatory physiological action in vertebrate animals (as opposed to a chalone, which has a depressant action). Usage is now extended to regulatory compounds in lower animals and plants, and to synthetic substances having comparable effects. [source: GOC:mah, ISBN:0198506732]

Comment None

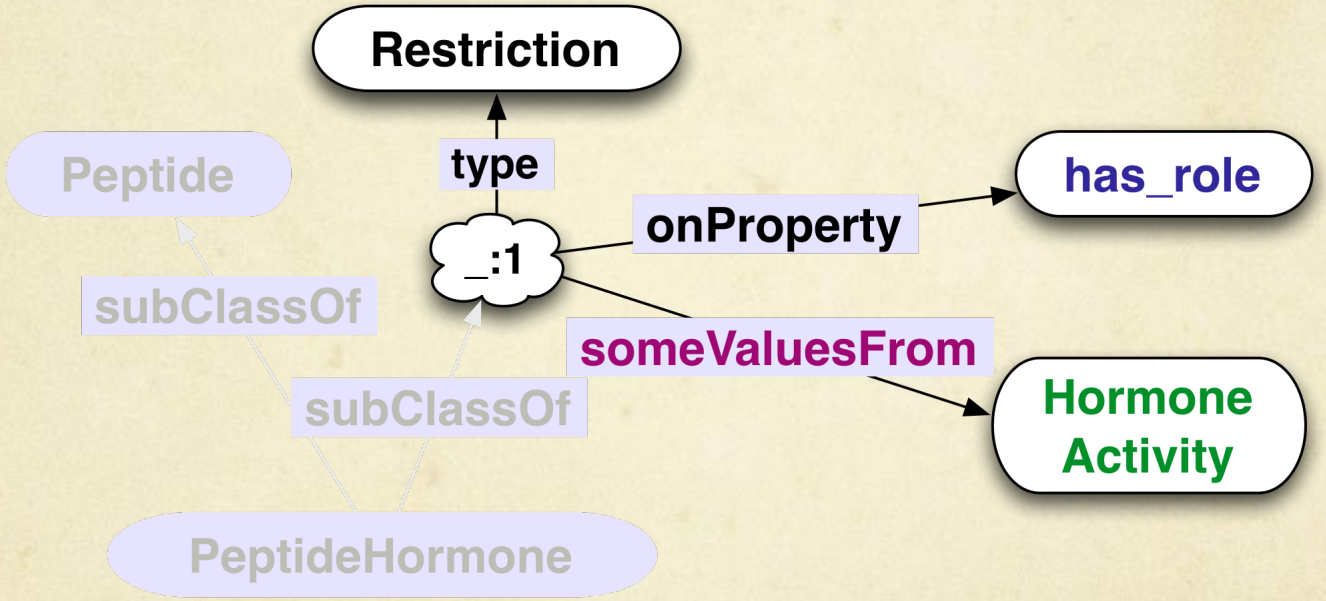
From English to OWL

- *ALL instances of PeptideHormone are an instance of Peptide that has_role SOME instance of HormoneActivity*

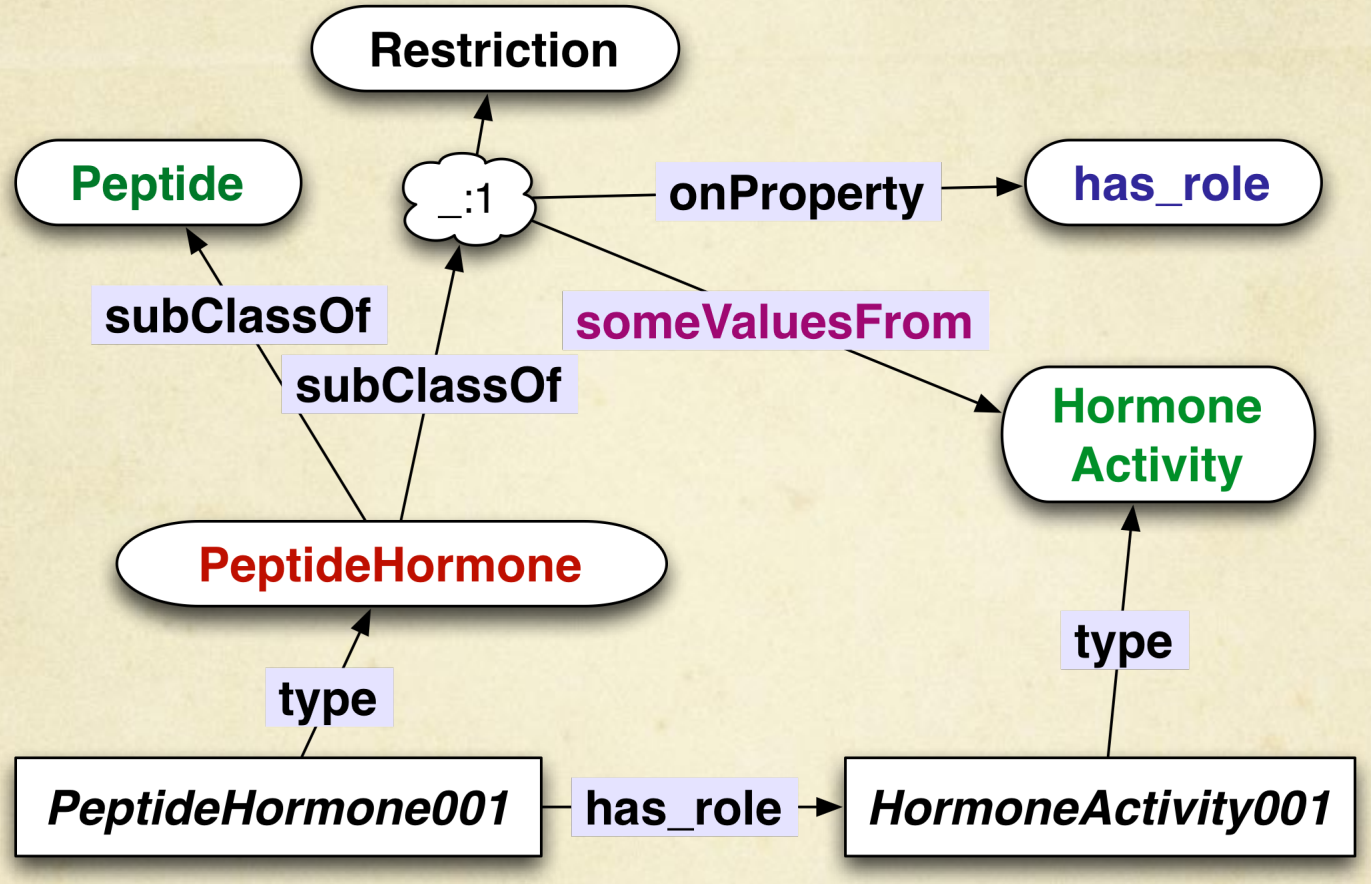


Restrictions - Classes Described by Phrases

- *ALL instances of PeptideHormone are an instance of* ***Things*** *that has_role SOME instance of HormoneActivity*

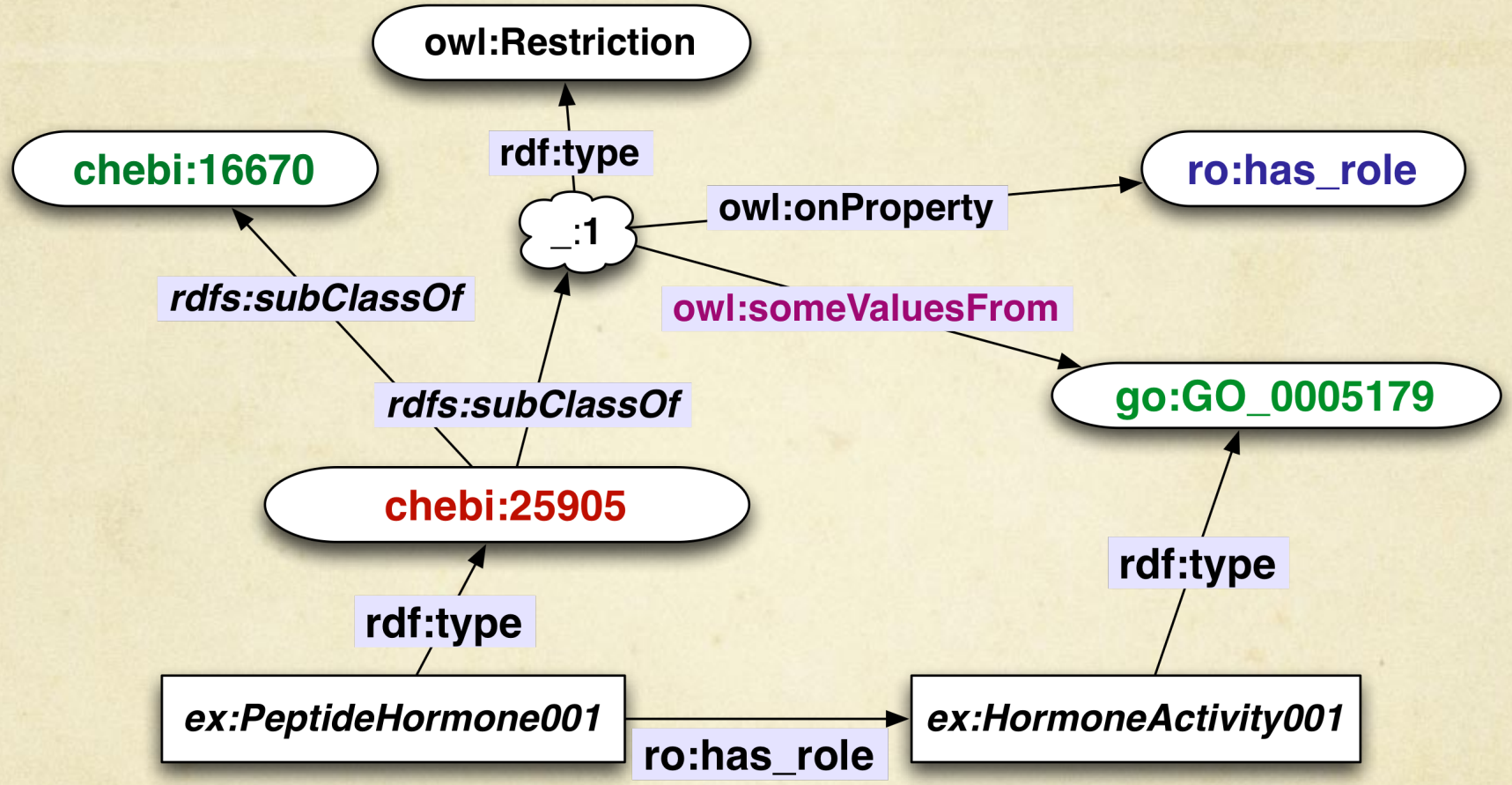


Representing Individuals



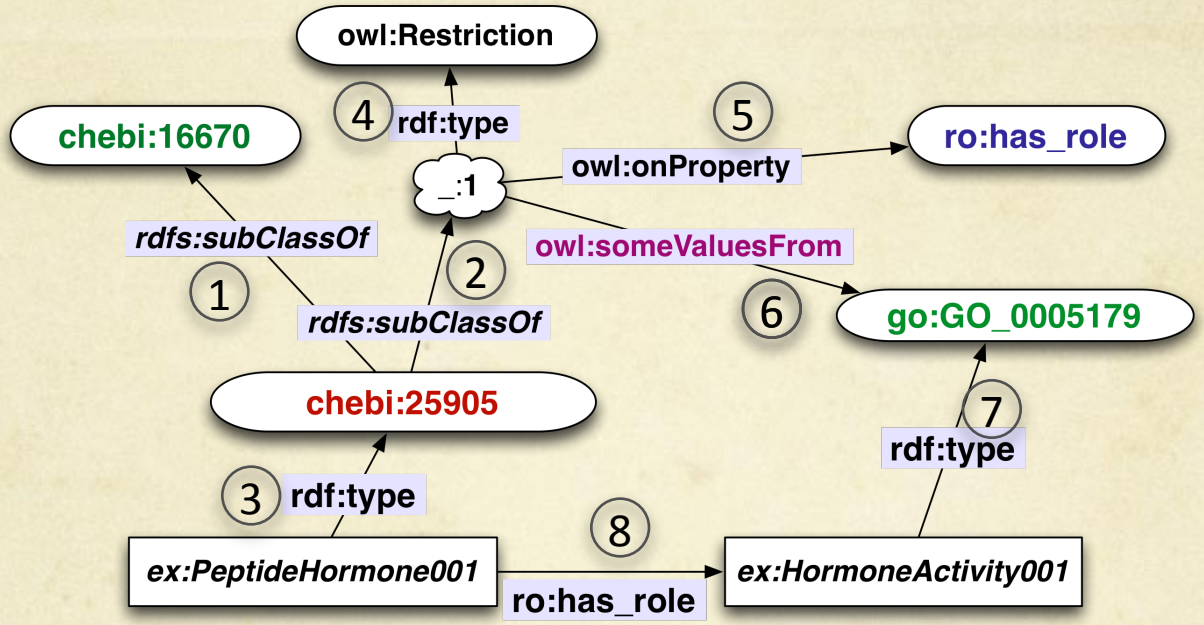
(Those individuals would represent individual molecules)

URIs, the Words of the Semantic Web



chebi:25905 = <http://purl.org/obo/owl/CHEBI#CHEBI_25905>

From Graph to Triples



1. chebi:25905 rdfs:subClassOf chebi:16670.
2. chebi:25905 rdfs:subClassOf _:1.
6. _:1 owl:onProperty ro:hasRole.
5. _:1 owl:someValuesFrom go:GO_00179. …

SPARQL: Matching Patterns of Triples

1. chebi:25905 rdfs:subClassOf chebi:16670.
2. chebi:25905 rdfs:subClassOf _:1.
6. _:1 owl:onProperty ro:hasRole.
5. _:1 owl:someValuesFrom go:GO_00179.

```
select ?moleculeClass
where {   ?moleculeClass rdfs:subClassOf chebi:16670.
          ?moleculeClass rdfs:subClassOf ?res.
          ?res owl:onProperty ro:hasRole.
          ?res owl:someValuesFrom go:GO_00179.  }
```

The match allows us to ***bind*** the variable ?moleculeClass = chebi:25905

Looking forward

- "It's the standard, stupid" More reasoning services are being standardized, which means more use and development. Standardization is good!
- The OWL 2 standard is recently finished and there is active work on implementations on many fronts
- Standard ontology languages are still only weakly expressive, particularly in the realm of mathematics. Expect hybrid systems to be developed
- Look towards combinations of DL, rule systems, spatial reasoners, probabilistic reasoning, symbolic mathematics, as well as approximate reasoning

Tools of the trade: Graphical Editors

- Protégé 3, 4
  - Protégé 3, OWL 1, Legacy, Stable (but obsolete for OWL development)
  - Protégé 4, Written for OWL 2, maturing (still a bit to go, but it's what I usually use)
- OBO edit
- Neon OWL editor
- Topbraid Composer

Tools of the trade: Programmer APIs

- OWLAPI – Java (http://owlapi.sourceforge.net/)
  - Common access to a growing number of reasoners
  - Provide constructors, reasoner invocation, Class expression query, extraction of inferred results
- Jena – Java (http://jena.sourceforge.net/)
  - Semantic Web oriented. RDF more than OWL.
  - Provides some constructors, SPARQL implementation, serialization to/from variety of formats
- Fuxi – Python (http://code.google.com/p/fuxi/)
  - Infix syntax for OWL
  - Reasoner for subset of OWL
- LSW 2 – Common lisp
  - What I use for my development
  - Base on OWLAPI, Hermit, Fact++, Pellet
  - Constructors, reasoning services, graphical view

Tools of the trade: Databases for OWL/RDF content

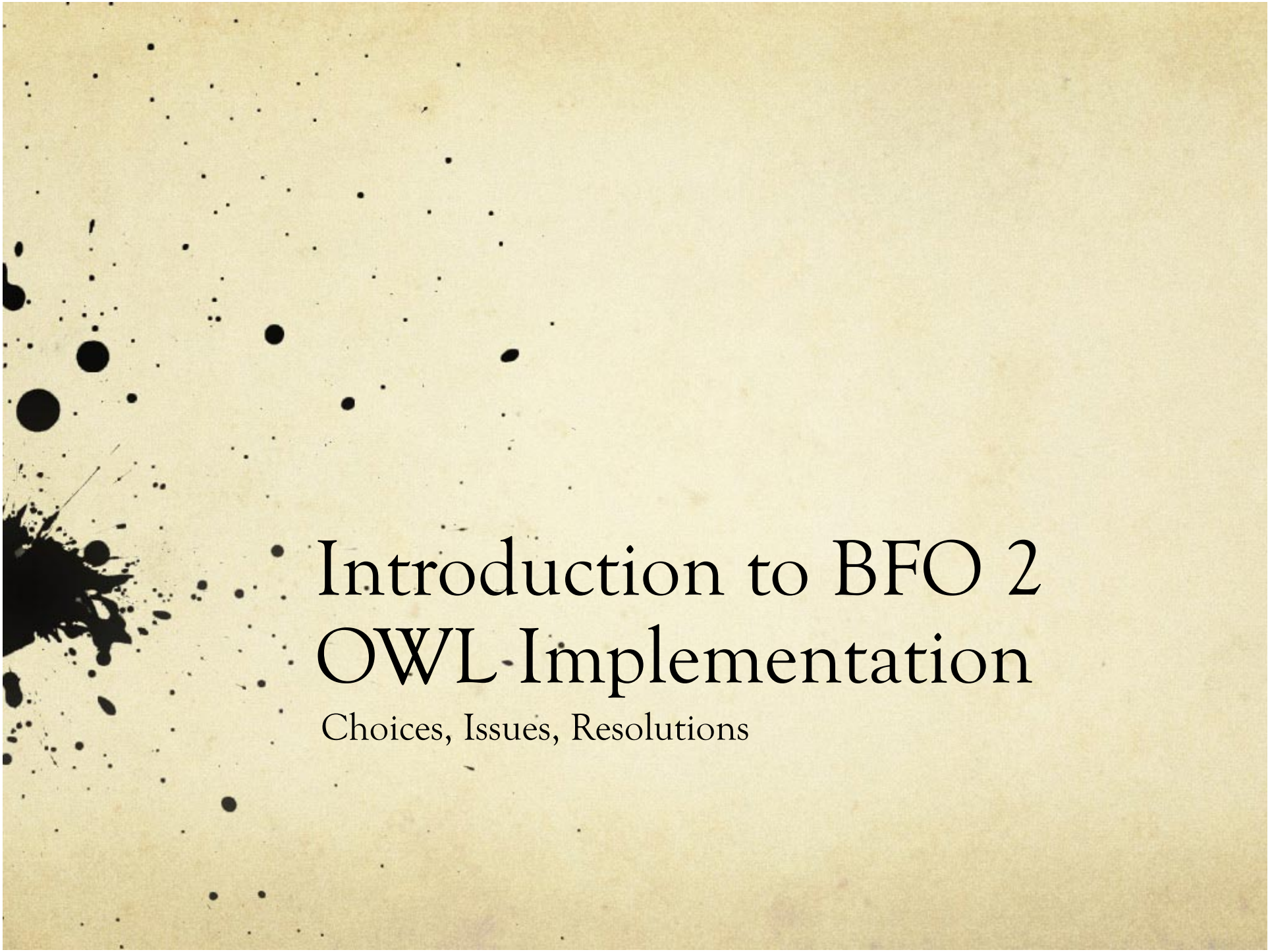
- Openlink Virtuoso: Triple store with small amount of reasoning capability but large capacity. What Neurocommons uses.
- Oracle 11g: Triple store supports OWL-RL, rule additions.
- Pellet, Stardog
- OWLIM: Subset of OWL reasoning over large store
- Sesame: Liked API, smaller data sets.

Tools of the trade: Query Languages

- SPARQL
  - RDF-based, reasoning agnostic, works with large stores. Active working group specifying new version. SPARQL-update extension adds update capability. http based protocol.
- "DL query"
  - Accept class expressions as queries – return subclasss, superclasses, instances, consistency. OWLapi, DL-Tab in Protégé, OWLink
- SPARQL-DL
  - Proposed extension to SPARQL to be OWL aware. Implemented as part of Pellet

Tools of the trade: Flavors of reasoning

- First order and higher order logic systems. E.g. Isabelle, Prover9. Suitable for very expressive, small systems.
- Rule based. Production systems. RIF, SWRL, Answer-Set (DLV)
- Description logics – OWL effective, expressive, reasoning
  - OWL-DL Most expressive. Most expensive to reason about
  - OWL-RL Suitable for implementation using rule based technology
  - OWL-EL Tractable reasoning for large numbers of classes



Introduction to BFO 2 OWL Implementation

Choices, Issues, Resolutions

Quick start

- Get Protégé 4.2 latest version
- Open http://purl.obolibrary.org/obo/bfo.owl
- Follow along

Goals of this section

- Say a few words about BFO
- Describe the BFO 2 OWL design principles
- Make clear the status of BFO 2 OWL
- Discuss contents of BFO 2 OWL
- Review changes relative to BFO 1
  - Give guidelines for migrating from BFO 1 to BFO 2
- Discuss selected aspects of the implementation
  - Strategy for time-indexed properties
- Review selected open issues
- Answer your questions

Status of BFO 2
2012-07-20 "Graz" Release

- This version of BFO represents **a major update** to BFO and **is not backwards compatible** with BFO 1.1. The previous version of BFO, **version 1.1.1 will remain available at http://ifomis.org/bfo/1.1** and will no longer be updated. We **expect to provide automated support** for migrating from BFO 1.1 to BFO 2 at some point **in the future**.

- **This version is a draft release for public comments.** As such we expect it to **generate a lively discussion rather than be a stable ground for building application ontologies**. The release is the product of intensive discussions and a series of prototypes which kicked off with a November 2011 workshop held in Buffalo.

- While we have attempted to produce a file reflecting the current BFO2 specification document, **there remain issues and bugs that will be fixed** in subsequent releases. Our expectation is that **IDs in this version, going forward, will not disappear** and we will **follow the GO/OBI deprecation policy**. In addition, **not all elements of the reference itself are stable and there may be (even substantial) changes** to the reference before we have the first release candidate.

OWL 2

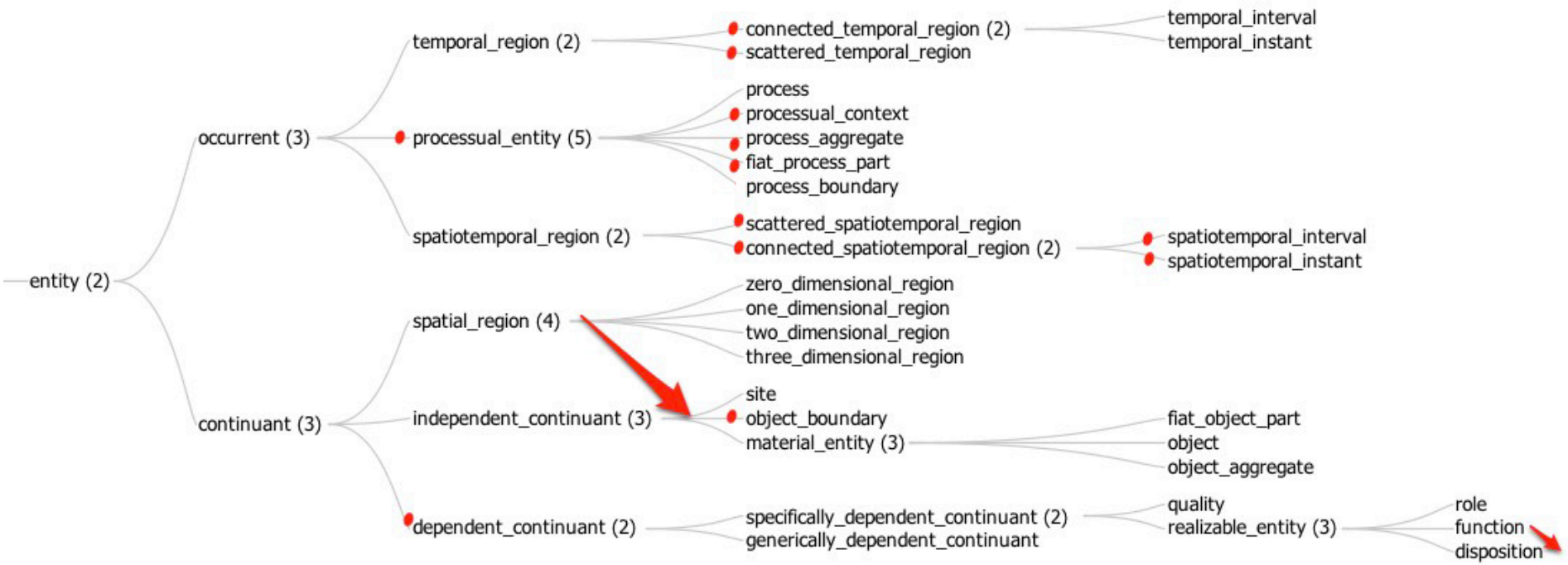
- We make use of the full range of OWL 2 using the direct semantics.
- We may release weakened versions (e.g. dropping axioms that are outside the EL profile) if there is demand
- Features of OWL 2 that are useful for us
  - Property chains
  - Ability to annotate anything (for documentation purposes)
  - Availability of free computational tools for reasoning
  - Built to live on the web, which makes for easy uptake
  - ObjectHasSelf, which allows for local reflexivity
- Problems
  - Running into issues related to the OWL 2 global constraints, for example wanting to use transitive properties in role chains, or make properties disjoint with self-linking properties

Selected design principles

1. A clear *reading* of the OWL version in terms of BFO reference. A translation is a mapping from the OWL model - axioms, entities, relations, etc. To the formal language used in the reference, currently FOL. A reading can be considered a data transformation that takes asserted and inferred axioms and results in FOL using types defined in BFO 2 reference. This translation should be complete - no assertions in the OWL file can be left untranslated.

5. Make maximal use of reasoning to ensure quality/correctness. It is easy to make mistakes when ontologies have no automated procedure to test them. By adding as many constraining axioms which correspond to textual descriptions in BFO 2 Reference, we have he best chance of finding conceptual errors. We will aim to make clear the relation between the FOL and the OWL, independently of the reference. We will examine the reference with a mind to ensuring axioms that are not so labeled are marked explicitly as axioms.

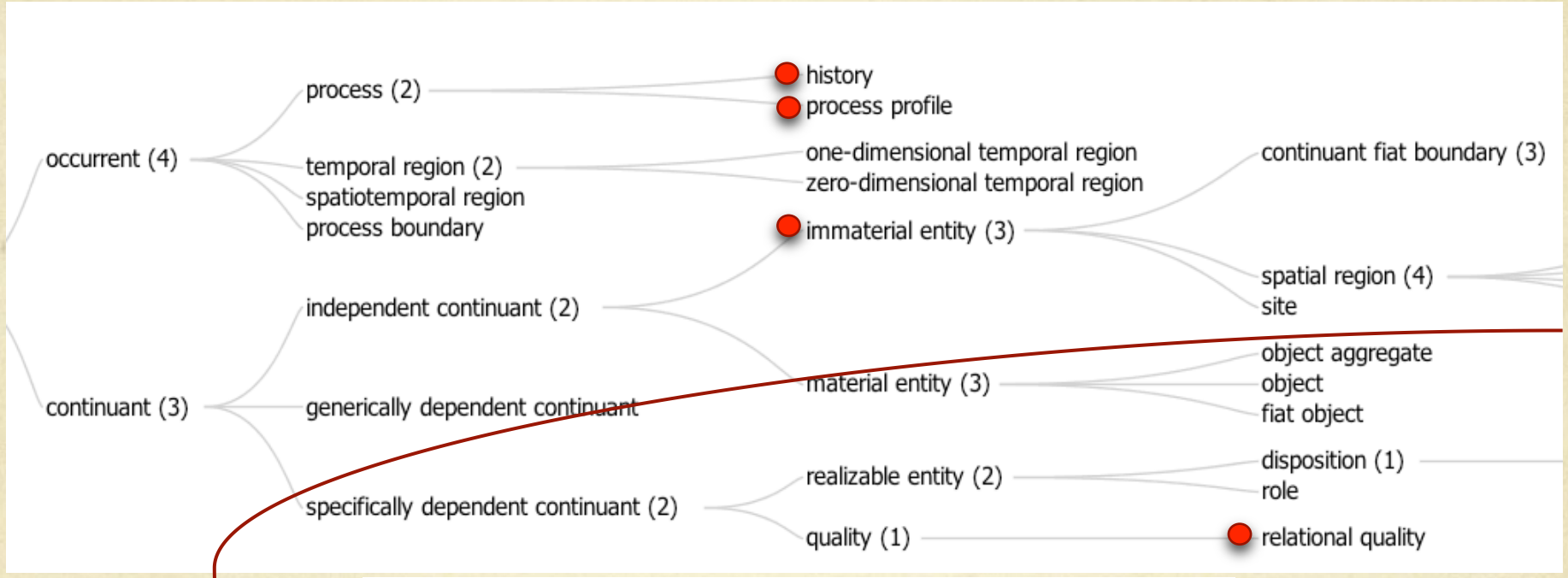
BFO 1.1 Classes



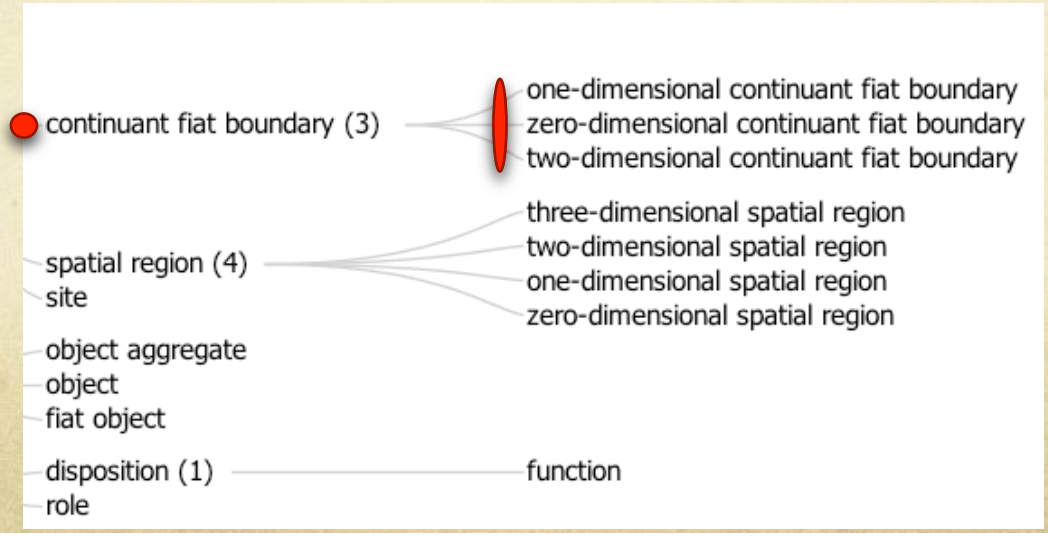
deprecated

moved

BFO 2 Classes



The tree was too wide



● Added

Relations in BFO 2

- has material basis (disposition -> material entity)
- has history (material entity<-> process, 1:1)
- continuant part of (continuant -> continuant)
  - member part of (object <-> object aggregate)
  - continuant **proper** part of
- occurrent part of (occurrent -> occurrent)
- concretizes (sdc -> gdc)
- exists at (entity -> temporal region)
- specifically/generically depends on (sdc, gdc -> site or material entity)
- specifically depends on (many possibilities)
- occupies spatial region (continuant -> spatial region)
- occupies spatiotemporal,temporal region
  - (occurrent -> spatiotemporal, temporal region)
- occurs in (process -> material entity or site)
- located in (material entity or site -> material entity or site)
- has participant (process -> continuant)

sdc = specifically dependent continuant
gdc = generically dependent continuant

No closure axioms in BFO 2

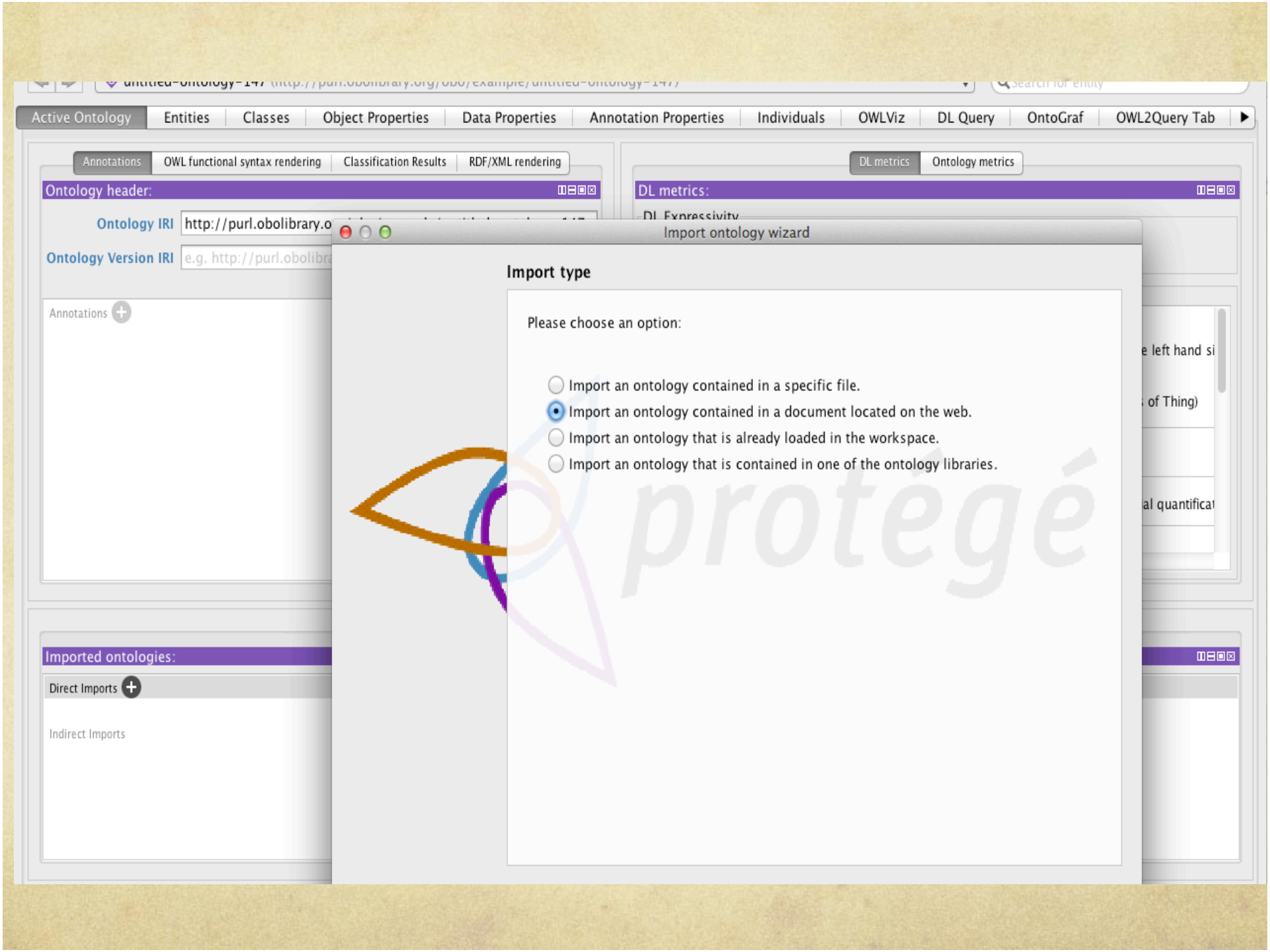
- OWL 1 had closure axioms – axioms that said that at each level the subclasses of a class were the only subclasses.
- These are no longer present in BFO 2
- Example: We don't close specifically dependent continuants because we're pretty sure there are more
- Example: We don't close continuant fiat boundary because the subclasses don't include the case of a sum of non-intersecting zero-dimensional (point) and one-dimensional (line) continuant fiat boundary.

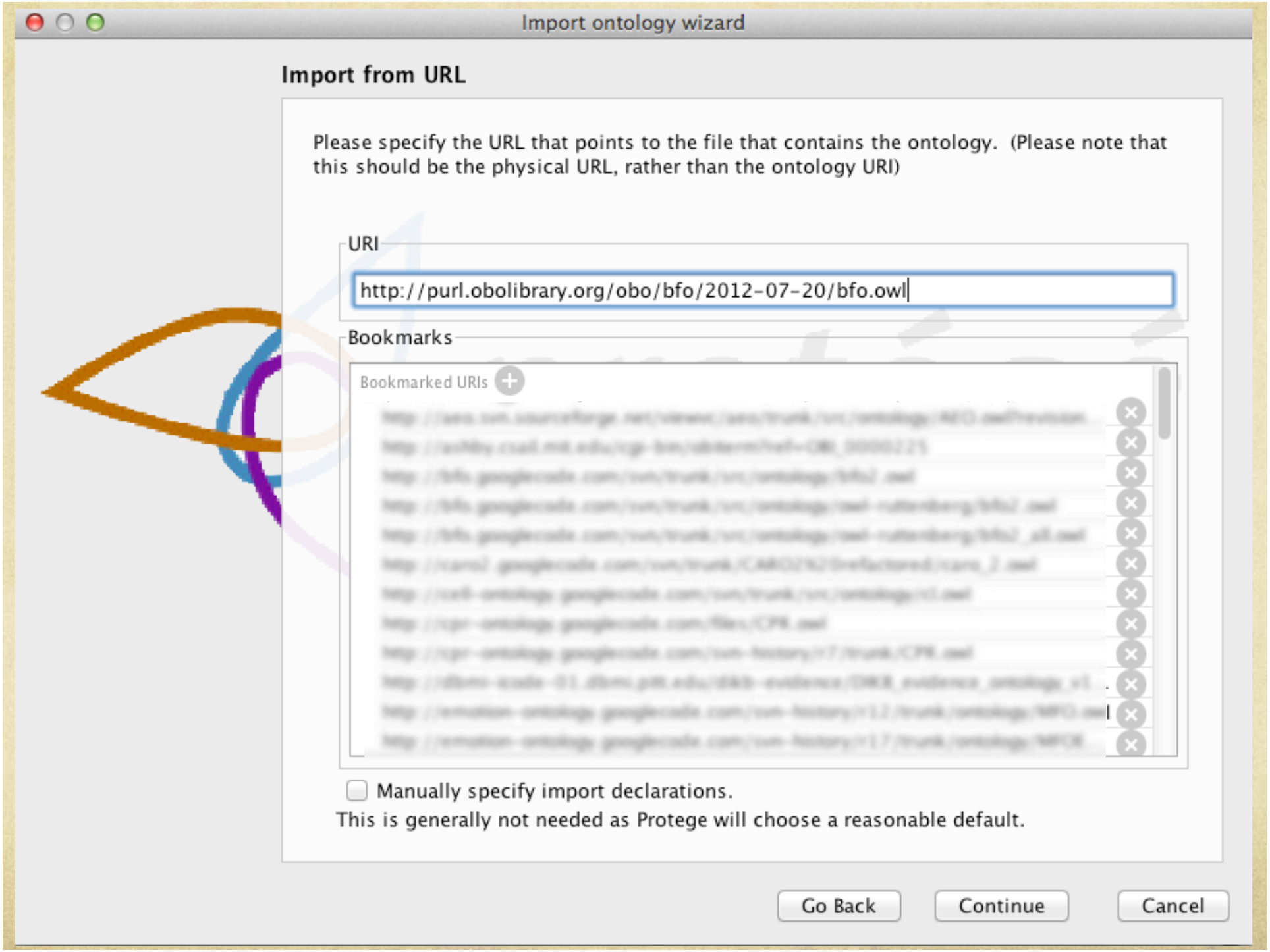
Siblings disjoint except between material entity children

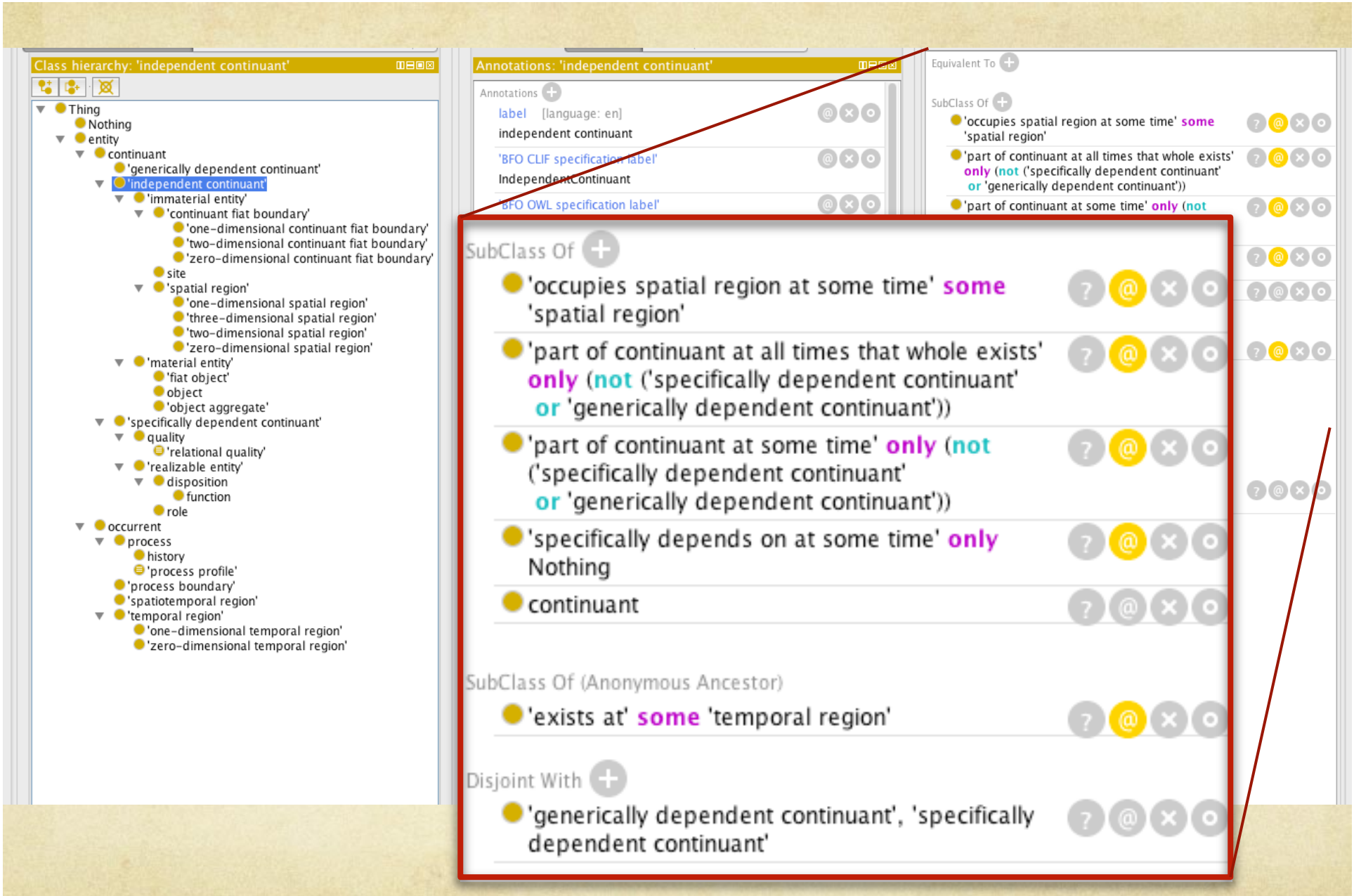
- A paradigm case: not ruling out granularity
- A molecule is, at the same time, an object aggregate (of atoms) and an object (for example as constituent of a crystal)
- In different granular contexts the same physically identifiable thing can justifiably classified as different types
- If we made the children of material entity be disjoint, we would make it hard to extend BFO to deal with granularity effectively.
- So we don't

Importing BFO into your Ontology

- Use the OWL imports functionality that your editing tool provides
- Give the import URI as
  - http://purl.obolibrary.org/obo/bfo/2012-07-20/bfo.owl if you want to make sure you get today's version
  - http://purl.obolibrary.org/obo/bfo.owl if you want to import whatever the most recent version of BFO is.
- Both URIs currently retrieve the same document, however that will change as soon as there is a new version released


Active Ontology
Entities
Classes
Object Properties
Data Properties
Annotation Properties
Individuals
OWLViz
DL Query
OntoGraf
OWL2Query Tab
Annotations
OWL functional syntax rendering
Classification Results
RDF/XML rendering
Ontology header:
Ontology IRI
Ontology Version IRI
Annotations
DL metrics
Ontology metrics
DL metrics:
Import ontology wizard
Import type
Please choose an option:
Import an ontology contained in a specific file.
Import an ontology contained in a document located on the web.
Import an ontology that is already loaded in the workspace.
Import an ontology that is contained in one of the ontology libraries.
Imported ontologies:
Direct Imports
Indirect Imports


Import ontology wizard
Import from URL
Please specify the URL that points to the file that contains the ontology. (Please note that this should be the physical URL, rather than the ontology URI)
URI
http://purl.obolibrary.org/obo/bfo/2012-07-20/bfo.owl
Bookmarks
Bookmarked URIs
Manually specify import declarations.
This is generally not needed as Protege will choose a reasonable default.
Go Back
Continue
Cancel



http://protege.stanford.edu/download/download.html

Get the latest version of Protégé 4.2

Migrating classes from BFO 1.1 to BFO 2

- Note deprecated classes –
  - Specifically dependent continuant
  - Process X classes
  - Scattered/Connected X
- Attend to classes that are similar to earlier classes
  - object boundary, continuant fiat boundary
  - Function now subclass of disposition
- Look for applicability of new classes
  - history, relational quality
- Learn the relations that are now in BFO and apply them.
  - Migrate use of RO to BFO, new RO

Migrating relations to BFO 2

- **Properties redesigned to not overload names as in BFO 1.**
  - part_of (class, class)
  - part_of (continuant, continuant, t)
  - part_of (occurrent, occurrent)
- **Strategies to remove overloading**
  - Drop class/class relations
  - Name properties according to type
  - Use “at-some-time”, “at-all-times” relations to acknowledge temporal indexing. (maybe –during-process relations coming)
- **How to migrate**
  - Attend to part relations which have split continuant/occurrent/at some times/at all times – If you use part relations a lot this will be the biggest work
  - Attend to other temporally indexed relations
  - Learn and use new relations
    - material basis
    - history of
    - specifically/generically depends on
    - concretizes
    - realizes

Why time indexing is important (even in BFO OWL)

- Documentation of BFO has always had time indexed relations, for example parthood between continuant instances is written as part_of(c1,c2,t)
- Our first design principle is that the the OWL file have a BFO reading
- In the prior Relation Ontology OWL implementation the part of relations were binary, and were not given any BFO reading.
- Since BFO defines the meaning of the relations, we have no basis for formally understanding relations if there is not a BFO reading
- We (OBO community) is moving more towards representation of instances. Prior to this only "class level" relations were used, which were not time dependent. However OWL only provides for making quantified sentences over individuals rather than reifying classes.

Understanding the time-indexing approach

- The problem: OWL provides only binary relations
- Time indexed relations are ternary
- Solution: Move quantification over time into the instance-level relation
- For part_of(c1,c2,t) define:

  part-of-at-some-time(c1,c2) =def

  forsome(t) exists_at(c1,t) & exists_at(c2,t) & part_of(c1,c2,t)

  part-of-at-all-times(c1,c2) =def

  forall(t) exists_at(c1,t) -> exists_at(c2,t) & part_of(c1,c2,t)

Note: only part-of-at-all-times is transitive

Note: part-of-at-all-times is not the inverse of has-part-at-all-times

Note: part-of-at-all-times-that-whole-exists can be defined so as to be the inverse of has-part-at-all-times

part-of-at-all-times-that-whole-exists(c1,c2) =def

forall(t) exists_at(**c2**,t) -> exists_at(**c1**,t) & part_of(c1,c2,t)

Known issues in BFO 2

- No proper way to represent non-rigid universals (professor, fetus) yet
- process profile not well liked by part working group
- Acceptability of time-indexing? How to extend it.
- Material basis only for dispositions (what about qualities, and what about qualitative basis?)
- Alan doesn't like specific dependence of processes on material entities
- Additional requested relations – begins/ceases to exist during (continuant -> process)
- BFO 2, OBO Format, and bfo-ruttenberg.owl (vintage 2010-05)
- Are some definitions too formal? (see def. of process, process profile)
- Change in definition of process boundary (now only temporal)

Known issues in BFO 2

- Permanent generic parthood, and similar. Can't say, currently, that at all times every independent continuant is located in some spatial region (different ones at different times)
- Support for granularity (what to do about has grain relation in use in some ontologies)
- Stages (temporal parts of history). Need to axiomitizing that they map uniquely to the material entity the history is of.
- Review of reference and completion of adding axioms. Feedback from OWL implementation back into reference.
- Only relation relating site to host is continuant part of
- How to relate process profile to, e.g. qualities they profile

Resources

- Release notes (start here)
  - http://purl.obolibrary.org/obo/bfo/2012-07-20/ReleaseNotes
- Reference document
  - http://purl.obolibrary.org/obo/bfo/2012-07-20/Reference
  - http://purl.obolibrary.org/obo/bfo/2012-07-20/Reference.htm
- OWL Document
  - http://purl.obolibrary.org/obo/bfo/2012-07-20/bfo.owl
- First order logic formulation
  - http://purl.obolibrary.org/obo/bfo/2012-07-20/fol.pdf
- Discussion list
  - http://groups.google.com/d/forum/bfo-owl-devel
- Tracker/Issues list
  - http://code.google.com/p/bfo/issues/list

Acknowledgements

- **Recent contributors**: Mauricio Almeida, Thomas Bittner, Jonathan Bona, Mathias Brochhausen, Werner Ceusters, Mélanie Courtot, Randall Dipert, Bill Duncan, Janna Hastings, Albert Goldfain, Leonard Jacuzzo, Ludger Jansen, Pierre Grenon, Larry Hunter, Chris Mungall, Fabian Neuhaus, David Osumi-Sutherland, Bjoern Peters, Mark Ressler, Robert Rovetto, Ron Rudnicki, Alan Ruttenberg, Stefan Schulz, Barry Smith.

- **Special thanks** to Stefan Schulz, Janna Hastings, Melanie Courtot for review of the OWL document, and Mathias Brochhausen for ideas for this presentation.