Interfaces in Biomedical Ontology

Part One

Ask not what you can do for your ontology,
but what your ontology can do for you

Alan Ruttenberg

alanruttenberg@gmail.com

University at Buffalo School of Dental Medicine

Ontology :: Computation
OWL and its Relation to Ontology

This section

- Discuss how we *leverage* ontology in computation
- Representation systems
- OWL as a representation language
- Representation language does not determine ontology
- There are choices for representation language
- First order logic as a representation language
- An example: How time in addressed in different representations

Leveraging ontology in computation

- An important use of ontology is in representations used in information systems
- Many uses
  - Consistency checking, nonsense rejection
  - Normalization for keyword-based document retrieval
  - Making it easier to have information be explicit rather than implicit, via inference
  - As elements of a *language* to pose questions, in a system with a well-defined method of answering them
  - As a tool to make data integration easier/possible
  - As a way to achieve shared understanding of system developers
- Hypothesis: Information systems based on ontology make it easier to do some tasks, makes others feasible, and can do so with high reliability

Computational representation systems

- Assume we know what our natural language definitions of a universal mean
- We can answer various questions about what particulars
  - Is this particular one of a type?
  - How are particulars of one type related to particulars of another type?
  - …
- A representation system is a way of encoding definitions in such a way as to enable a computer program answer certain questions we might answer
- Representation systems have as components
  - Formal language
  - A way to formulate some questions
  - A mechanism by which the components of the formal language can be manipulated to answer those questions

Web Ontology Language (OWL)

- Conceived as an information system for use on the web
- As such builds on existing web standards
  - URLs
  - RDF
  - XML
- An abstract model with several different syntaxes
- Documentation of the semantics of the language as a translation from abstract model to first order logic
- Specifies reasoning tasks that a conformant processor of the language should be able to accomplish
- What is called an "ontology" in OWL can be a mixture of specifications of classes, relations, and instances
- Along with other standards coined "The semantic web"

The Semantic Web 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

International Resource Identifier (IRI)

- Generalization of URL,URI = uniform resource locator
- A resource, in our terms, is a portion of reality (POR, aka *thing*)
- e.g. http://purl.obolibrary.org/obo/OBI_0000070
- There are other *schemes*, e.g. DOI, ISBN but we recommend HTTP for terms)
- IRIs are case sensitive
- Part of the HTTP IRI scheme is the domain name: purl.obolibrary.org
- Domains are leased by people who want to *mint* IRIs
- IRIs with different domains can't overlap by mistake
- HTTP IRIs serve double duty:
  - globally unique symbols
  - locates a web page that has documentation about the term
- Commonly written as *CURIEs* obo:OBI_000100
- *Prefixes* ("obo") are defined within a single document in a section near the start
- IRIs I use will look like :OBI_0000070 – no prefix means implies a default prefix
- IRIs are meant for machines. A human readable label is associated via the rdfs:label property
- Good user interfaces always let you use the labels

OWL is a model theory

- Fundamental idea: Interpretation
- Interpretation
  - Domain of discourse - A set of resources (*PORs*)
  - Set of properties. For a given property the set of pairs of things for which the property holds (property extension)
  - Set of classes. For a given class the set of things which are members of the class (class extension)
  - A mapping of URIs (symbols) to things in the domain of discourse, properties and classes
- Analogy to BFO *(for now)*
  - The domain of discourse is all entities (particulars)
  - Properties are relations
  - Classes are universals or defined classes
  - Property extensions are those pairs of particulars which are related by the relation
  - The URIs are the symbols that denote particulars, classes, relations
  - Call this the *automatic correspondence*

Semantics of OWL is in terms of conditions on interpretations

- Conditions are given by axioms in the language
- A basic operation: is an interpretation even possible?
- An interpretation is possible if the conditions are not contradictory
- More advanced: Are there additional conditions consistent with all interpretations (entailments)?
- A model is a mathematical structure over which the conditions can be evaluated – a way of simulating interpretation
- From a realist perspective,
  - There is only one valid interpretation – what actually is the case in the world
  - Axioms should be *true*
- Conditions are *ambiguous* to the extent that they allow for multiple interpretations

Simple example

- Reality: Alan participates in this presentation
- Formulation
  - Domain of discourse: {Alan, Barry, this presentation}
  - Property: participates in
  - Classes: presentation, person
  - IRI assignment:
    - :person1->Alan, :person2->Barry, :presentation1->this presentation
    - :participates_in (a property)
    - :person, :presentation (classes)
- Note: Domain of discourse does not include properties or classes
- Axioms (conditions)
  - Only :person :participates_in :presentation
  - :person1 *instance of* :person
  - :person2 *instance of* :person
  - :this_presentation *instance of* :presentation
  - :presentation and :person are disjoint

There is more than one interpretation

- Consistent interpretations
  - :person -> {Alan, Barry}
  - :presentation->{this presentation}
  - :participates_in
    - {Barry, this presentation}
    - {Alan, this presentation}
    - {Alan, this presentation}, {Barry, this presentation}
- The above can be computed by manipulations on models (only)
- Of course, only one of them is actually true
- Nonetheless this computation can be leveraged in many ways

Checking consistency and entailments depends on what axioms are allowed in the language

- Decidability: Given a set of axioms are there algorithms for checking consistency and entailment that will always terminate?
- Soundness: Do the algorithms always get the right answer?
- Completeness: Do the algorithms get all the answers?
- First order logic is not decidable (not possible for there to be a sound, complete algorithm that always terminates)
- OWL designed to be decidable, sound, complete
- OWL specifies axiom form, semantics, and questions
  - Consistency
  - Entailment
  - Query answering
- Algorithms to answer questions were known at the time of publication

Expressivity

- Authors are responsible for the
  - Assignment of URIs to entities, classes, properties
  - Translating truths into axioms
- Garbage in, garbage out
- Two senses of expressivity
  1. The range of allowable axioms, which indirectly constrain the sort and form of things we say about the world
  2. Given some axioms, to what extent can the *system* derive true things we care about (entailments)
- The first kind of expressivity is fairly unconstrained because you can have properties whose values are natural language text
- Action is in #2

OWL is actually a set of languages

- Two dimensions
  - *Semantics*: Description Logic(DL) or RDF-based
  - *Profile*
- Full set of allowable axioms defined by *abstract model*
- Profiles restrict what axioms are allowable
  - OWL-QL – Designed so it can be implemented in relational databases
  - OWL-RL – Designed so it can be implemented in rule systems
  - OWL-EL – Designed to be *tractable*. Algorithms are faster
- Total of 8 possibilities: 2*[semantics]* x (1*[unrestricted]* + 3*[profiles]*)
- Defined questions
  - Are axioms consistent or inconsistent
  - Entailment: Given two sets of axioms, are the second true in all models of the first
  - Query answering conceptually similar to entailment
  - If the second set of axioms (the query) has unknowns (variables), what are replacements of variables by terms for which the resultant ontology is entailed by the first
- Semantics are via model theory
- For the DL semantics there are known algorithms that are decidable, sound, and complete
- In the rest of this talk DL semantics is assumed

Distinction between individuals and literals

- OWL has a hard line between individuals (instances/objects) and literals
- Literals are common computer data types: integer, string, boolean
- Classes have only individuals as members
- Data Ranges have only literals as members
- Object properties only relate individuals to individuals
- Data properties only relate individuals to literals
- Annotation properties can relate to both individuals and literals, but have no logical consequences
- Reasons:
  - Sound, complete algorithms aren't known when classes includes both instances and literals
  - Literal types have "built-in" structure that classes don't, for example >,< for numbers

Class axioms

- SubClass axioms assert that the extension of one class is a subset of some other
- EquivalentClasses axioms assert that the extensions of two classes are the same
- DisjointClasses axioms assert that no individual is a member of more than one of the specified classes
- In the above, classes can be *named* (by a URI) or described by a *class expression*
- A class expression is the logical form for constraints on members
- Class expressions allow one to represent:
  - Union of classes
  - Intersection of classes
  - Complement of class
  - Classes whose members are completely specified
  - Classes for which members are always related to at by minimum, maximum, or an exact number of individuals of some type
  - Classes for which members, if they are related by a property, the object of that property is of a type
- Example:
  - EquivalentClasses(:doctor ObjectIntersectionOf(:homo_sapiens ObjectSomeValuesFrom(:has_role :Doctor_role)) (*functional syntax*)
  - :doctor EquivalentTo: :homo_sapiens and (:has_role some doctor_role) (*manchester syntax*)
  - Doctor ≡ ∃ has_role.doctor_role (*DL syntax*)
  - ∀ x doctor(x) ⇔ ∃ y doctor_role(y) ∧ homo_sapiens(x) ∧ has_role(x,y) (*FOL*)

Properties

- Object properties can be asserted
  - Transitive (if r(a,b) and r(b,c) then r(a,c))
  - Functional (for each subject there can be only one object)
  - Symmetric (if a relates to b, then b relates the same way to a)
  - Antisymmetric (if a relates to b then b doesn't relate in the same way to a)
  - InverseFunctional (for every object there can be only one subject)
  - Reflexive (every object is related to itself)
  - Irreflexive (no object is related to itself)
  - To have an Inverse (r1 is inverse of r2 means if r1(a, b) then r2(b, a)
  - To have a Domain (if domain of r is A, then if r(a,b) then a is an instance of A)
  - To have a Range (if range of r is B, then if r(a, b) then b is an instance of B)
  - To be a SubProperty of another (if r1 subproperty of r2 then r1(a,b) => r2(a,b))
  - To be Disjoint from another (if r1 disjoint r2 then r1(a,b) => not r2(a,b))
  - To be Equivalent to another property
  - To Chain (if r1, r2 are properties, r1 o r2 means: If r2(a,b) and r1(b,c) then r1 o r2 (a, c))
- Caveat: Not all of these can be used together
- Data properties have similar assertions (except that a literal can't be the subject of any relation)

Individuals

- Can be asserted to be members of a class
- Can be related by properties to other individuals or literals or it can be asserted that they are NOT related to another individual
- Individuals can be asserted to be the same (or rather: URIs can be constrained to denote the same resource)
- Individuals can be asserted to be different
- There can be individuals for which there is no IRI (anonymous)
- Examples
  - ClassAssertion(:alan, :homo_sapiens)
  - ObjectPropertyAssertion(:participates_in :alan :presentation)

IRIs in OWL

- The same IRI can map to each of an individual, a property, a class
- Even though the same IRI is used, *statements using it in any of these ways don't interact with each other.* Sometimes called "punning."
- Only rarely is punning a good idea
- SameIndividuals(:a,:b) does NOT imply EquivalentClasses(:a, :b)
  - If the IRI mappings *did* interact, it would
- There is no *unique name assumption*
  - More than one IRI can map to an entity
  - Motivation: Different people on the web might not know they are talking about the same thing and so give different URIs for it
  - Consequence: Sometimes surprising entailments or non-entailments

Annotations

- Annotation properties are ignored in models, interpretations
- Used to associate information with an IRI or un-named element of the syntax: label, definition, …
- Consequence: If you pun, you can't associate information with *just* the class, property, or individual the IRI maps to
- In addition to IRIs, subjects of annotations be axioms, class expressions, property expressions, or even other annotations

OWL syntaxes

- There are multiple syntax specifications
- Functional syntax is very close to the abstract model. Verbose.
- RDF/XML is *normative*
  - Was designed to be compatible with an existing RDF model and syntax
  - Unnatural for expressions of many OWL axioms
  - BUT: The model that must be used in popular query(SPARQL) engines
  - Important as a practical matter, but I won't discuss here
- Manchester syntax was designed as a more comfortable way to write OWL
  - Still unfriendly: Have to use URIs instead of labels
  - A modified version that does use labels is used in Protégé (editor discussed in future session)
- An XML syntax
  - For people who like XML (yes, they exist!)

Correspondence between Ontology and OWL

What you should be thinking about when using a computable representation for ontology

- Assume you can express your ontology using FOL and that one can straightforwardly understand FOL as BFO
- How do I translate my definitions into the representation
- Given a representation, how do I translate it to FOL (and so understand what something means in BFO)
- What inferences can the computational system make given the representation in your computational system
- Are the inferences made *sound* – if you translate them in FOL, are they sensible from a BFO perspective

Classical first order logic

- Held, by some, to be "gold standard"
- Why not just use that?
- Difficult to compute with – algorithms not guaranteed to halt.
- In practice algorithms frequently don't halt, or need 'help' to work with a set of assertions
- Although powerful, not necessarily enough expressivity
- Not often enough validated
- Not necessarily perspicuous

- Example BFO2 definition of g-depends, using a dialect of FOL

> b **g-depends on** c **at** t_1 means: b exists **at** t_1 and c exists **at** t_1
>
> & for some type B it holds that (c **instantiates** B at t_1)
>
> & necessarily, for all t (if b exists **at** t then some **instance_of** B exists **at** t)
>
> & not (b **s-depends_on** c **at** t_1).

- 2 clauses trivially satisfied by choosing B<-entity
- What do we care about that "some instance of B", which isn't otherwise related
- Information entities conceived as GDC, but they don't stop existing when the *second to last* copy ceases to exist
- Does the use of *necessary* mean this is actually modal logic?

Just because it is expressed in FOL doesn't mean it's better

Correspondence between ontology and representation language

- What are the elements and how do they work?
  (covered in first part of presentation)
- What do we want to say?
- How do we use the elements to say what we want?
- The *automatic* choice
  - Particulars in BFO are individuals in OWL
  - Universals, attributive classes in BFO are classes in OWL
  - Relations in BFO are object properties in OWL
  - is_a in BFO is subClassOf in OWL
  - Necessary or sufficient conditions are in term of class expressions in OWL
  - They are made necessary by using subClassOf
  - They are made sufficient by using equivalentClasses

Problems

- What is the correspondence between time-indexed relations in BFO and OWL?
- OWL only has binary properties
- Time indexed expressions are usually ternary: a part_of b at t
- Usual answer: What time indexing?
- Second answer: OWL can't do temporal relations. Don't use OWL.
- Common use of OWL for BFO-based ontologies use binary in place of ternary relations
- As a result the interpretation of such OWL files wrt. to BFO ontology is uncertain
- What does continuant_part_of(a, b) mean? Not defined in BFO.
- Consequence: Some OWL you read will be easily understood using BFO, some not
- Shouldn't be the end of the story

Requirements for productively using a representation language for ontologies based on BFO

- Have a clear *reading* of the OWL version in terms of BFO reference
  - A reading is a mapping from the OWL model - axioms, entities, relations, etc. to the formal language used by BFO (currently FOL)
  - A reading can be considered a data transformation that takes asserted and inferred axioms and produces FOL that use types defined in BFO 2 reference
  - The translation should be complete - no assertions in the OWL file can be left untranslated
- We should make maximal use, subject to the above, of the semantics of the representation language to ensure sound maximally complete readings
  - 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 the best chance of finding conceptual errors

You don't have to use the automatic correspondence Example: Reification of Relations

- For a statement involving a relation in BFO, create an instance for the relation, and use *other* relations to connect it to the subject, object, time
- BFO: c part of c1 at t
- OWL: 'reification'
  - part_of1 a PartOfRelation
  - part_of1 has_subject c
  - part_of1 has_object c1
  - part_of1 at_time t
- Trade offs
  - Now not all instances correspond to particulars
  - Bunch of OWL expressivity unusable: TransitiveProperty, et. al

Case study: Representations with "part of" in the wild

- "part of" is not used consistently
- Sanctioned uses of "part of" in the community of BFO developers
  - Class-Class relations – **nucleus** part of **cell**
  - FOL ternary predicate on instances – part_of(**nucleus-1, cell-2, 12pm**)
  - OWL binary predicate on instances (PropertyAssertion)
  - OWL class expressions: **nucleus** part of *some* **cell**

Instance-instance relation (FOL)

- In FOL written as
  - part_of(c, c1, t) for c,c1 continuant
  - part_of(o,o1) for o,o1 occurrents
- t is a temporal region (could be point or not, continuous or not)
- Continuant and occurrent relations are different!
- In BFO literature often written, for continuants, as: c part_of c1 at t

Class-class relations

- Relations in biomedical ontologies
- Defined in terms of instance-instance relation
- Relation between continuant classes: C ***part_of*** C1
  - for all c, t,
    if C(c,t) then there is some c1 such that C1(c,1t) and c **part_of** c1 **at** t.
- Here C(c,t) means: c is an instance of C at t
- In english: C ***part_of*** C1 if whenever an instance of C exists, it is always part of some instance of C1 (though not necessarily the same one at all times)
- Also (later) called Generic Parthood

Order of quantification matters in FOL It didn't have to be Generic Parthood

- Compare these definitions, where the existential are in different places
- As in Relations in Biomedical Ontology
  - for all c, t, if Cct then **there is some c1** such that C1c1t and c **part_of** c1 **at** t
  - English: C ***part_of*** C1 if whenever a C exists, it is always part of some C1 (though not necessarily the same one at all times)
- Alternative
  - for all c, t, **there is some c1** and if Cct then C1c1t and c **part_of** c1 **at** t
  - English: C ***part_of*** C1 if there is a C1 such that whenever a C exists it is **always** part of that C1
  - Currently called *Permanent Specific Parthood*
- Order of quantification matters
- There are *choices* when making definitions

OWL's two place instance relations

- c part_of c1 is a property assertion. It expresses a relation between two individuals.
- C subclassOf part_of some C1 is a class expression
- forall (c) c instanceof C => there is some instance of C1 such that c part of c1
- In OWL the instances are related in models (classes, properties used are as constraints)
- The classes are only related in that they appear in the same expression, not in the ontological sense – they are not in the domain of discourse
- Note there is no "at t", and there can be no Generic parthood
- Given any external temporal interpretation, OWL version would correspond to Specific parthood

Another variation on the automatic correspondence

- Move quantification over time into the instance-level relation
- For part_of(c1,c2,t) define:

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

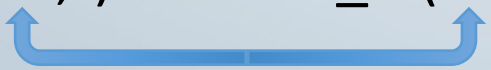
  exists(t) exists_at(c1,t) & exists_at(c2,t) & part_of(c1,c2,t)

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

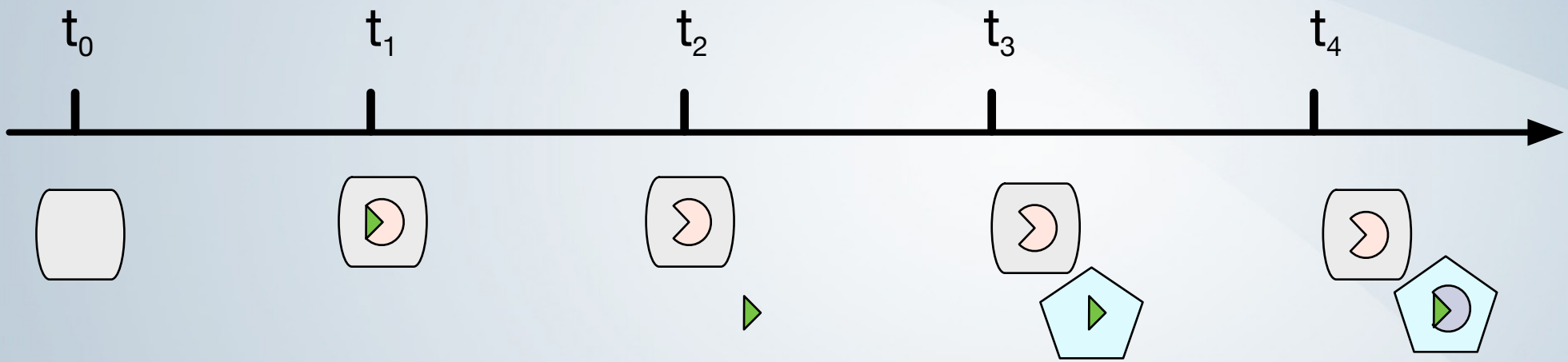
  forall(t) exists_at(c1,t) -> exists_at(c2,t) & part_of(c1,c2,t)
- Note:
  - only part-of-at-all-times is transitive
  - part-of-at-all-times is not the inverse of has-part-at-all-times

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

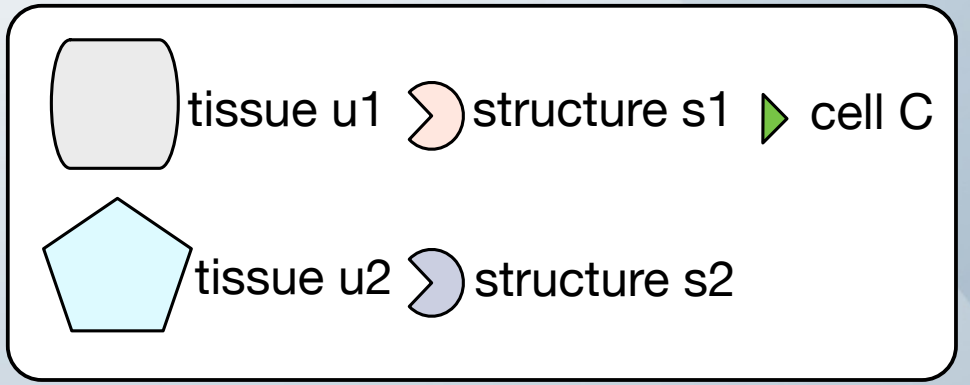
  forall(t) exists_at(**c2**,t) -> exists_at(**c1**,t) & part_of(c1,c2,t)



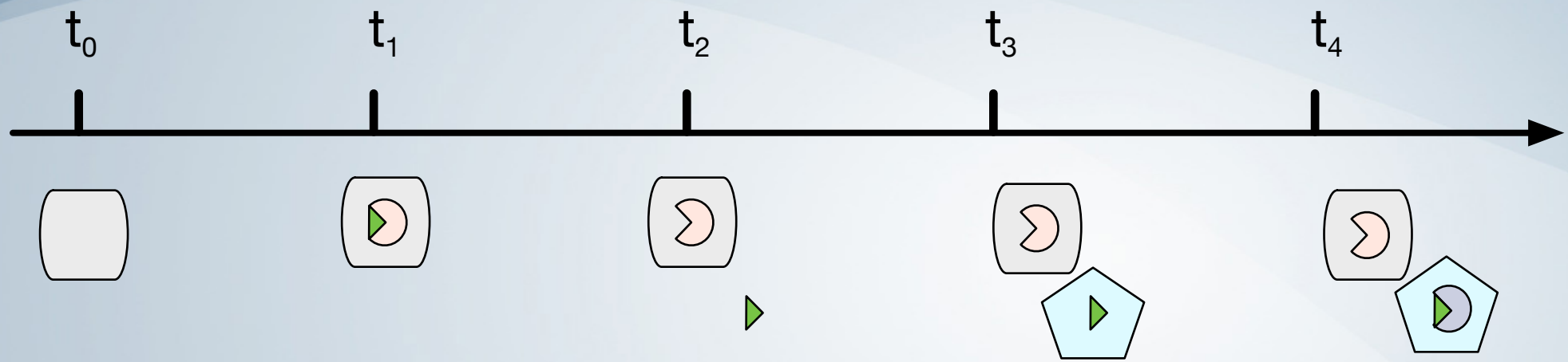
The distinction between uses of part_of matters Example: Studying a development mechanism



Situation: A cell develops in one tissue, migrating to a different tissue when that develops



At different times observations are made and recorded, following BFO



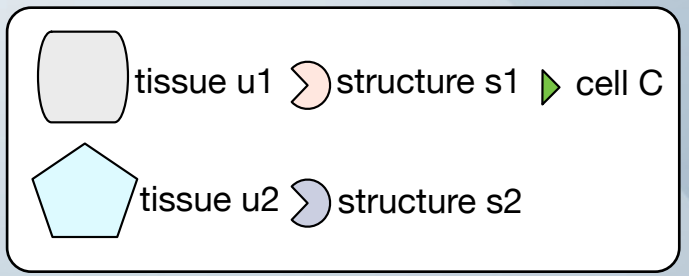
not(c part of u1 at t3)

s1 part of u1 at t1
c part of u1 at t1

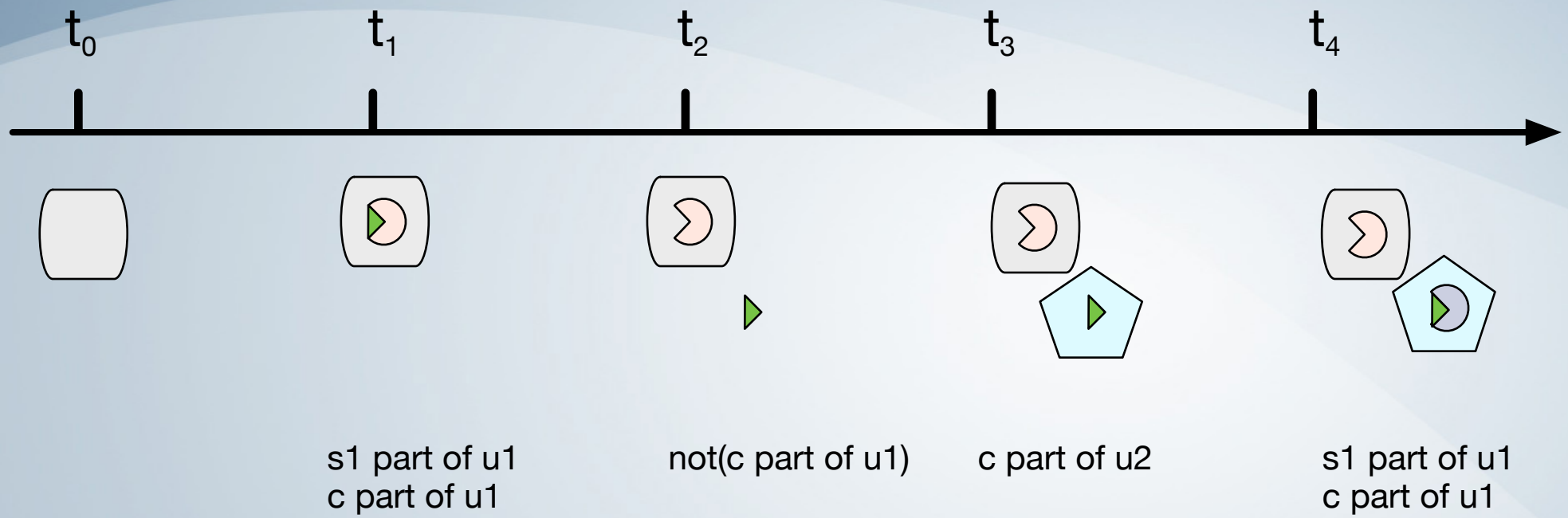
c part of u2 at t3

s2 part of u2 at t4
c part of s2 at t4

No problems



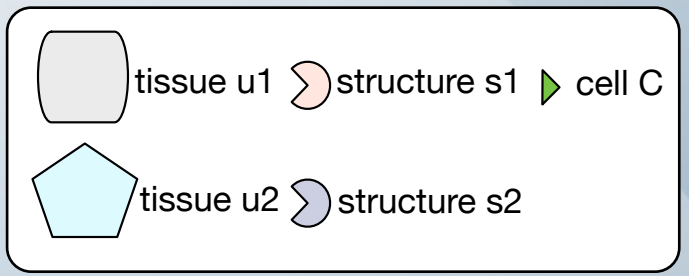
The same situation, but observations encoded using automatic correspondence with OWL, which only has binary relations



Problem: inferences can be made that don't match real situation

part of: transitive
c part of s1
s1 part of u1
=>
c part of u1 (contradicts biology)

c part of s1
not (c part of s1)
=>
logical contradiction



Yet another variation

- Switch to talk of histories
- BFO defines a 1:1 relation (has history) between a material entity and a process called its history
- History of entity =def process that occupies, at any time, the same spatial region as the entity
- Some things that might be asserted using time indexing can be instead asserted about parts of histories
- 'temporal part of' some ('history of' some nucleus)
  SubClassOf
  'part of occurrent' some ('temporal part of' some ('history of' some cell))
- Every temporal part of the history of a nucleus is part of a temporal part of some history of some cell
- Open problem: How do we ensure that assertions using history work well with assertions using the material entity?

Wrap up

- No representation language has all of what you want
- Not being careful with representation leads to systems that come to wrong conclusions
- OWL has limitations and benefits
  - Benefits
    - Widely used
    - Tools available
    - Proven to be helpful
    - Scales (trading off expressivity for size of problem)
  - Limitations
    - Not necessarily easy to express things we want to express
    - Sometimes too easy to express some things that aren't interpretable
- You don't have to be limited by an assumed mapping of ontology to OWL
- Current representations of BFO's time-indexed relations in OWL need to be fixed
- Notable: *instantiation of continuants is also time indexed* in BFO, but not when using the automatic correspondence of BFO to OWL