

Data Modeling with the Sequence Ontology

Chris Mungall

Berkeley Drosophila Genome Project

SO Instances

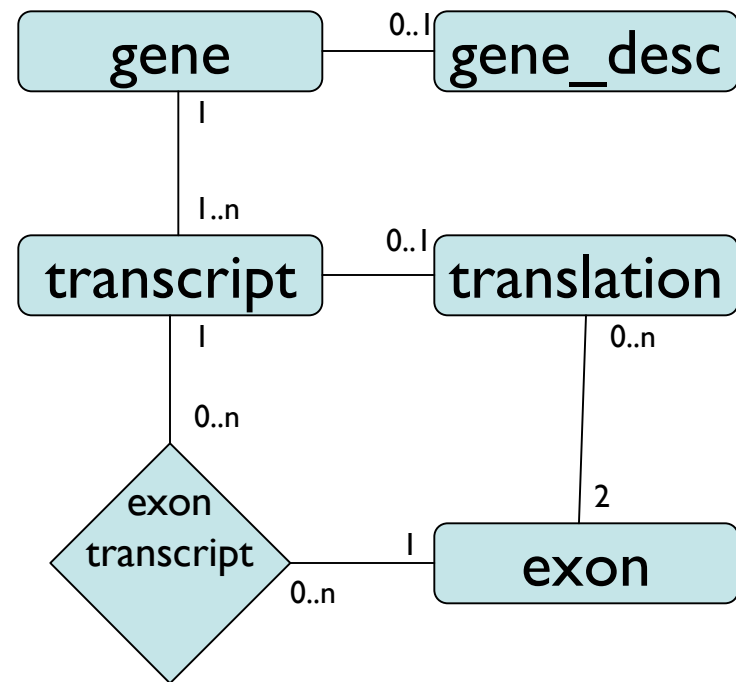
- What would an instance of a SO term look like?
- Minimum info:
 - ID, SO type, names/synonyms/labels
- Very useful info:
 - sequence location, relationships to other instances
- Optional info
 - relationships to other OBO terms, relationships to other values (eg heterozygosity for seq_variants)
- We need a *formalism* to represent SO instances

What do we mean by data model?

- A data model is a framework for capturing knowledge in a way that is computable
 - We need a data model for recording SO instances
- Data model formalisms
 - Relational (eg SQL Databases)
 - Hierarchical (eg XML; NCBI ASN.1)
 - Object Oriented (OO DBs; OO languages; UML)
 - Ontology formalisms (obo; protégé; DLs)
 - Ad hoc flat file formats (tab delimited; custom)
- Multiple formalisms are suitable for modeling SO instances

Relational model

- solid theoretical basis
- excellent for data management and querying
 - robust DBMSs
 - SQL standard
- **BUT: no type system!**
 - rigid models
- degenerate variations:
 - tab delimited files
 - eg GFF



ensembl relational schema

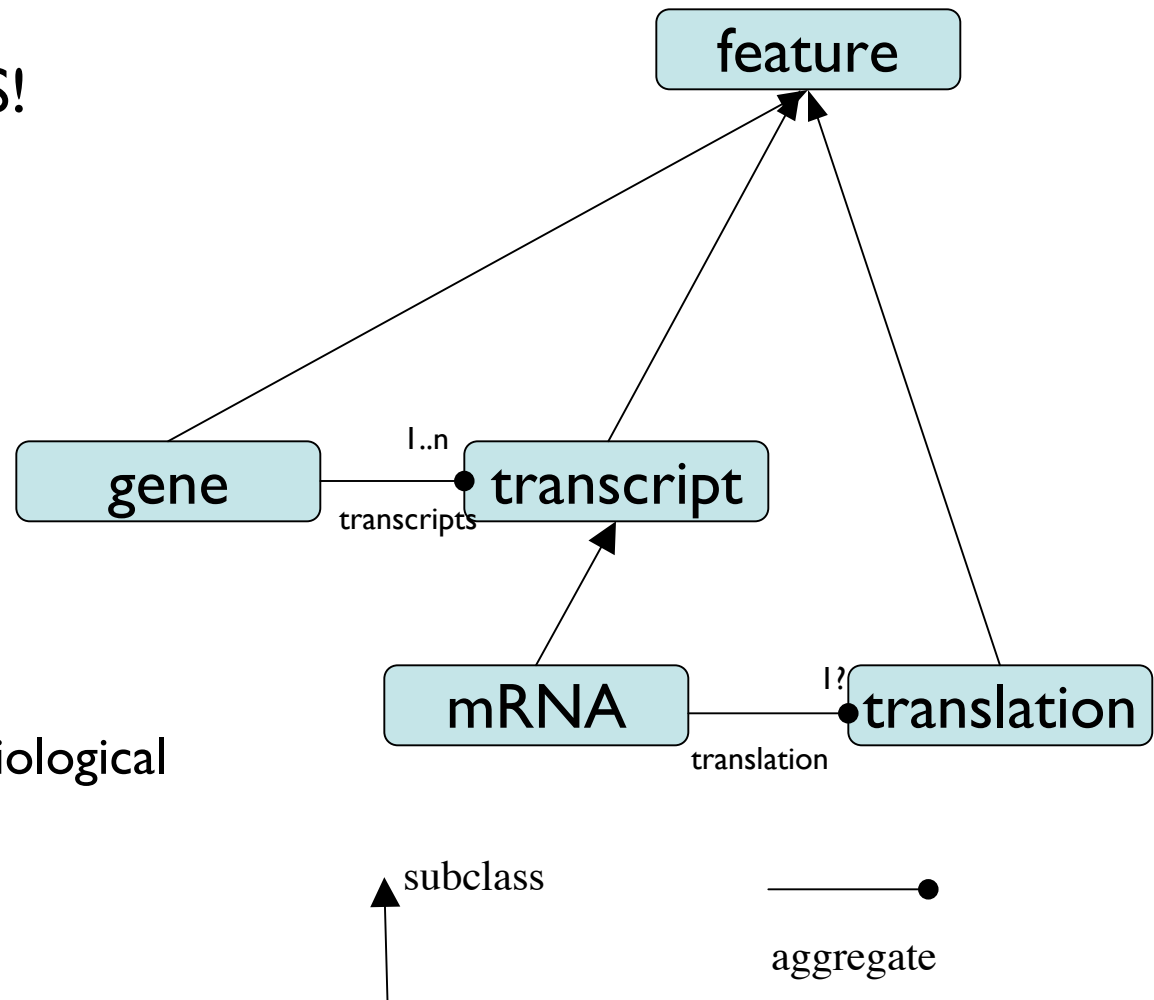
Hierarchical formalisms

- real data is often nested
- good for data exchange
- maps (reasonably) well to relational
- no proper type system!
 - types and properties as elements

```
<gene id="CG1234">
  <name>foo</name>
  <transcript id="CG1234-A">
    <name>foo-A</name>
    <seq>GCCA...</seq>
    <translation id="CG1234-P1">
      <name>foo-protein-1</name>
      <seq>MKVV....</seq>
      <pos start=132 end=561>
    </translation>
    <exon>
      <pos start=200 end=300>
    </exon>
    <exon>
      <pos start=500 end=800>
    </exon>
  </transcript>
</gene>
```

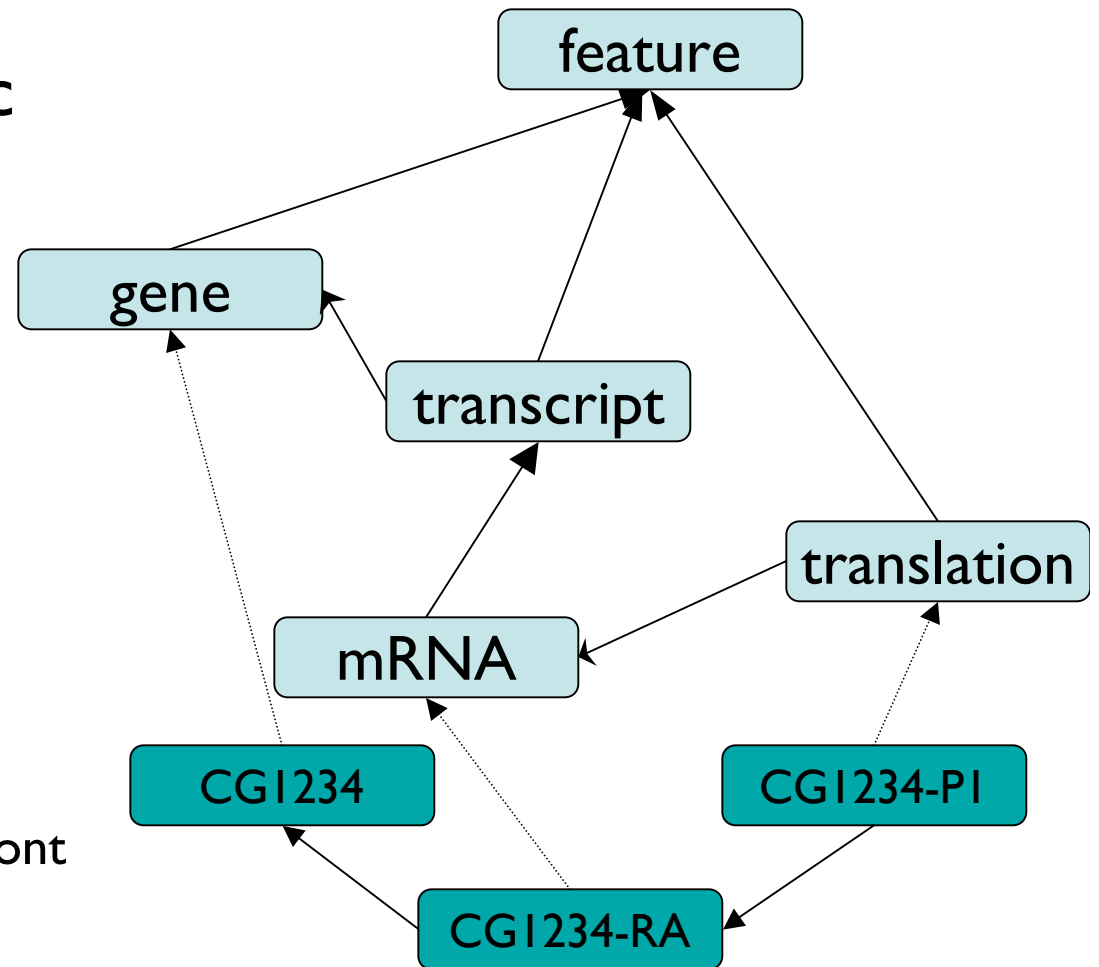
Object Oriented

- type system? YES!
- lacks theoretical foundations
- standards? tools?
 - OQL?
 - UML?
 - Perl/Java code?
- religious belief:
 - OO is bad for biological modeling



Ontology formalisms

- Type system centric
- Variations
 - obo
 - frame-based
 - similar to OO
 - logic-based
- Tools less mature
 - hybrid models are most pragmatic
 - relational or xml + ont

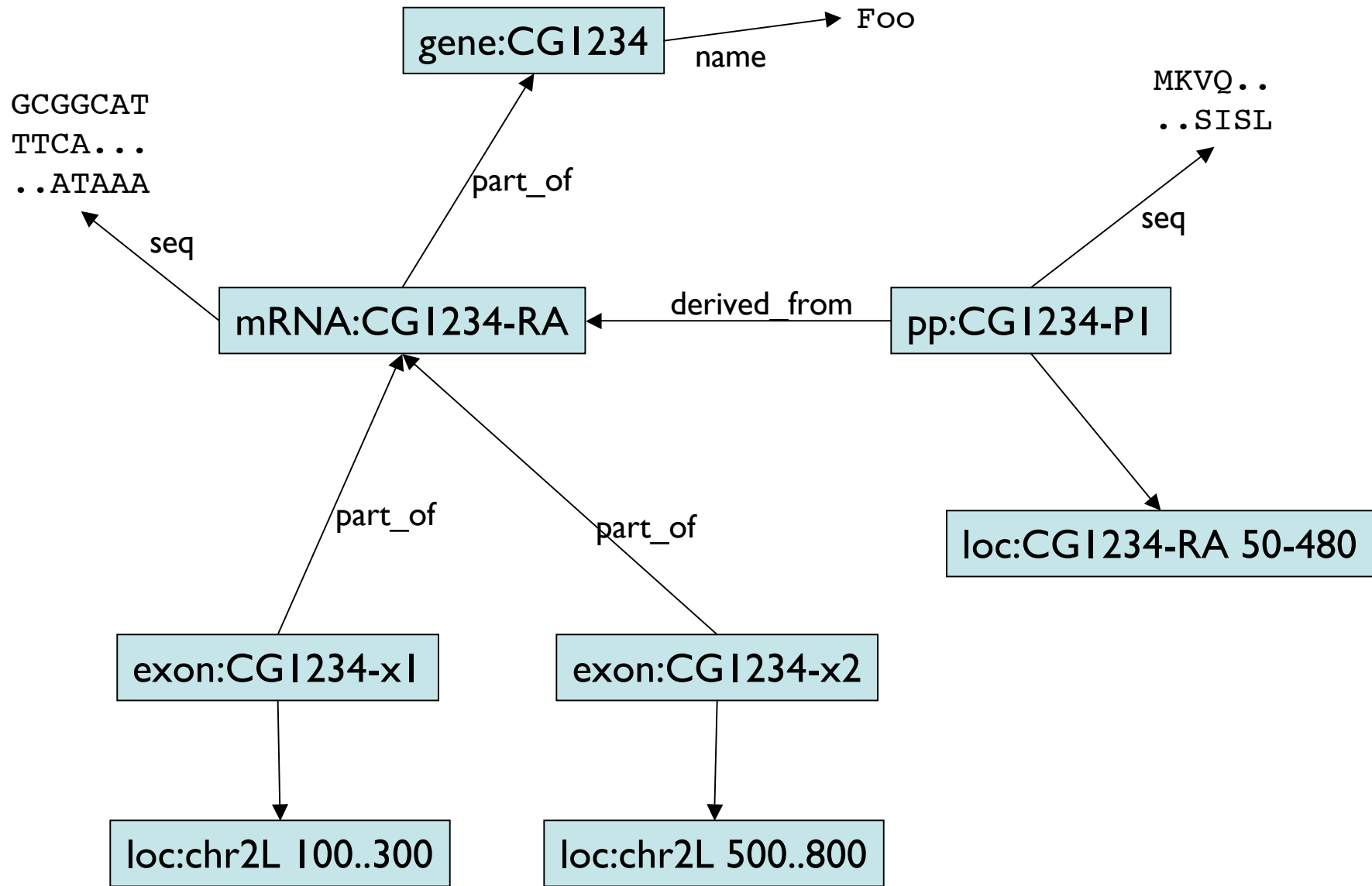


Hybrid Formalisms

- To represent SO instances we must choose a formalism
- Bio-Ontologies are not just for attaching metadata to existing data
- Ontologies can be used to enhance or replace traditional data modeling (relational, xml, object)
 - software and database flexibility and interoperability
- SO can be used for strongly-typing a weakly typed traditional datamodel

SO-based datamodels

- Datamodel has minimal typing
- Classes (SO terms) in ontology
 - C = gene, exon, regulatory_region, ..
- Instances
- Sequence locations
- Relationship and property types
- Relationships and property assignments
 - binary (typed) relationships between instances
 - **CGI234-RA** *part_of* **CGI234**
 - property assignment: between an instance and a datatype value
 - **SNP000000123** *heterozygosity* 0.45



loc:seq		type	loc:start	loc:end	loc:str		relationships
2L		gene	1	800	+		ID=CG1234; Name=Foo
2L		mRNA	1	800	+		ID=CG1234-RA; Parent=CG1234; seq=GCGG..
2L		CDS	151	300	+		ID=CG1234-PI; Parent=CG1234- RA; aa=MVKQ...
2L		CDS	501	780	+		ID=CG1234-PI; Parent=CG1234- RA; aa=MVKQ..
2L		exon	101	300	+		ID=CG1234-x1; Parent=CG1234- RA
2L		exon	501	800	+		ID=CG1234-x2; Parent=CG1234- RA

Advantages of SO-based datamodels

- **Flexibility**
 - unlimited configurations for instances and datatype values in a graph
 - dicistronic genes; trans splicing; RNA editing
- **Robustness**
 - all types belong to SO
 - relationships restricted by SO
- **Practicality**
 - can be combined with a simple relational/xml/object model

Existing SO datamodels

- GFF3 - tab delimited text
- Chado - relational
 - XML mappings:
 - ChadoXML
 - ChaosXML
 - Object mappings:
 - CGL (perl)
- BioPerl?? (only containment hierarchies)
- RDF/OWL?
- Any datamodel can be *enhanced* by SO
the above datamodels are *reliant* on SO for typing

Tools: go-dev

- java: DAG-Edit
- go-perl
 - event based parsers
 - objects for manipulating ontologies as graphs

Tools: BioPerl

- Parsers and writers for GFF3
- Writers for ChadoXML and ChaosXML
- `Bio::SeqFeature::Tools::TypeMapper`
 - maps feature types to SO
 - infers relationship type between features
- `Bio::SeqFeature::Tools::Unflattener`
generates SO feature graph from genbank records

Tools: Chado & Chaos

- <http://www.gmod.org>
- Chado schema
 - cvs: schema/chado/
- ChaosXML tools
 - cvs: schema/chado/chaos-xml/

Chado SO layer

- Chado uses a generic relational model with weak typing
 - feature, featureloc, featureprop, feature_relationship, cvterm
- The Chado-SO layer creates virtual tables for all SO types and relationships

```
SELECT avg(seqlen) FROM exon;  
SELECT count(distinct intron_id)/  
       count(distinct gene_id)  
FROM gene2intron
```

Current shortcomings of SO based datamodels

- Relationship types not yet fully specified in SO
 - domains, ranges
 - datatype properties
 - cardinality [eg dicistrionic transcripts have ≥ 2 pp]
- Lack of easy to use validation tools
- Too flexible???
- Underspecification: many ways of saying the same thing

New obo / DAG Edit extensions: cardinality

[Typedef]

name: derives_from

inverse_of: derives

[Term]

id: SO:xxxxxx

name: dicistronic_mRNA

*is_a: SO:mRNA

*relationship: derives SO:polypeptide
{card=2}

[Term]

id: SO:xxxxxx

name: polycistronic_mRNA

*is_a: SO:mRNA

*relationship: derives SO:polypeptide
{mincard=2}

* indicates completeness (necessary and sufficient conditions)

Relationships

[Typedef]

name: makes_aminoacid

domain: tRNA

range: CO:aminoacid

[Property]

name: encodes_anticodon

domain: tRNA

range: datatype:string

[Term]

id: SO:xxxxxx

name: alanyl_tRNA

relationship: makes_aminoacid CO:alanine

relationship: encodes_anticodon one_of(**GCA GCC GCG GUA**)

Feature Graph Equivalencies

- There are different ways to represent the same data using different feature graphs
- SO does not proscribe or prescribe
- GFF3 and chado/chaos docs suggest (different) specific variations
 - chado: gene, transcript, exon
- GFF3 can “hide” some features like exons as transcript split locations
- SO could be used to provide mathematical definitions in order to do automatic conversions

GFF3 vs chado; gene models

GFF3 - canonical protein coding gene

$I = (\text{gene}, \text{mRNA}+, \text{5'UTR}+, \text{3'UTR}+, \text{CDS}+)$

$R = (\text{mRNA:gene}[m:I], \text{CDS:mRNA}[m:m],$
 $\text{5'UTR:mRNA}[I:I], \text{3'UTR:mRNA}[I:I])$

GFF3 - alternate representation

$I = (\text{gene}, \text{mRNA}+, \text{exon}+, \text{CDS}+, \text{5'UTR}*, \text{3'UTR}*)$

$R = (\text{mRNA:gene}, \text{exon:mRNA}, \text{CDS:mRNA}, \text{UTR:mRNA})$

GFF3 - minimal model (eg when UTR is unknown)

$I = (\text{gene}, \text{CDS}+)$

$R = (\text{CDS:gene}[m:m])$

Chado/Chaos - canonical protein coding gene

$I = (\text{gene}, \text{mRNA}+, \text{exon}+, \text{polypeptide}+, \text{intron}*, \text{UTR}*)$

$R = (\text{mRNA:gene}[m:m], \text{exon:mRNA}[m:m],$
 $\text{polypeptide:mRNA}[m:m])$

Mapping between equivalent feature graphs

- Flexibility can be a good thing
- But it also makes it harder to write programs
- Logical definitions of SO classes can help

Computable definitions

[Rule]

forall exon(x1) exon(x2)
consecutive(t,x1,x2) \Leftrightarrow
exists(intron(i)
start(i,t)=end(x1,t)
end(i,t)=start(x2,t))

[Rule]

forall intron(i) \Leftrightarrow
exists(donor_ss(d)
start(d,t)=start(i,t))
exists(acceptor_ss(a)
end(a,t)=end(i,t))

[Rule]

forall mRNA(t)
derives_from(p,t) \Leftrightarrow
exists(five_prime_utr(u)
start(u,t)=0
end(u,t)=start(p,t))

[Rule]

forall mRNA(t)
derives_from(p,t) \Leftrightarrow
exists(three_prime_utr(u)
start(u,t)=end(p,t)+3
end(u,t)=length(t))

Computable definitions

[Rule]

forall gene_defnX(g)
transcript(t)
part_of(t,g) \Leftrightarrow
connected_by_exons(t,t')
part_of(t',g)

[Rule]

forall gene_defY(g)
gene_product(p)
derives_from_part(p,g) \Leftrightarrow
overlaps_on_genome(p,p')
derives_from_part(p'g)

Future applications

- Validation
 - Check types and feature graph makes sense
 - Check properties and values are valid
- Automatic classification based on definitions
 - eg dicistronic_transcript
- Querying over implicit instances
 - introns, UTRs, splice sites

Acknowledgements

Karen Eilbeck
John Day-Richter
Mark Yandell
Shenqiang Shu
Suzanna Lewis

Dave Emmert
Stan Letovsky

Lincoln Stein
Richard Durbin
Michael Ashburner