Introduction to the Semantic Web for Bioinfomatics

Ken Baclawski
Northeastern University

Outline

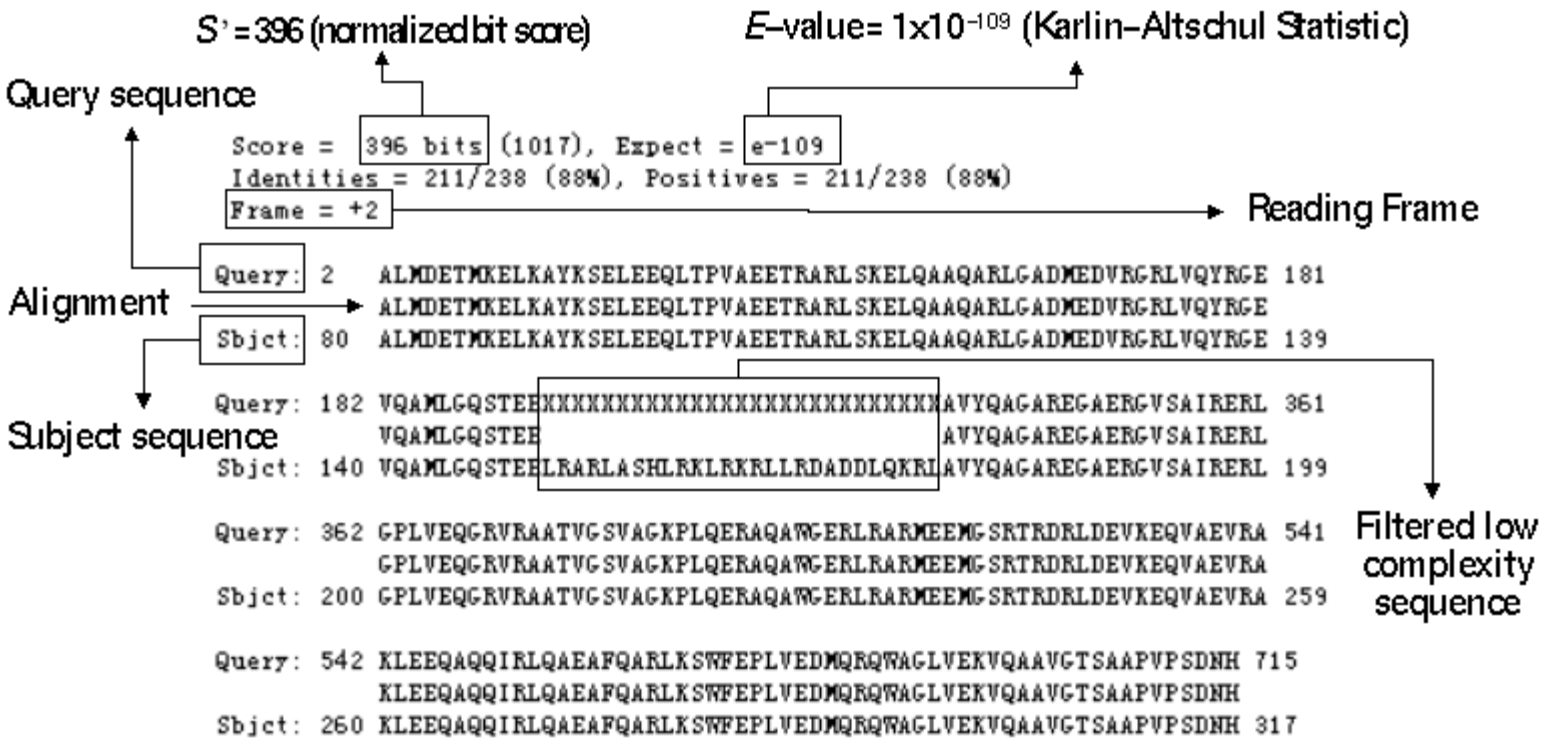
I. **Semantic Web Languages**
   - A. **Hierarchies and relationships**
   - B. **Basic XML semantics**
   - C. **Data semantics**
   - D. **RDF semantics**
   - E. **OWL semantics**

II. Semantic Web Usage
   - A. Ontology based information retrieval
   - B. Transformation languages and tools
   - C. Semantic Web services
   - D. Bayesian Web: Combining logic and probability

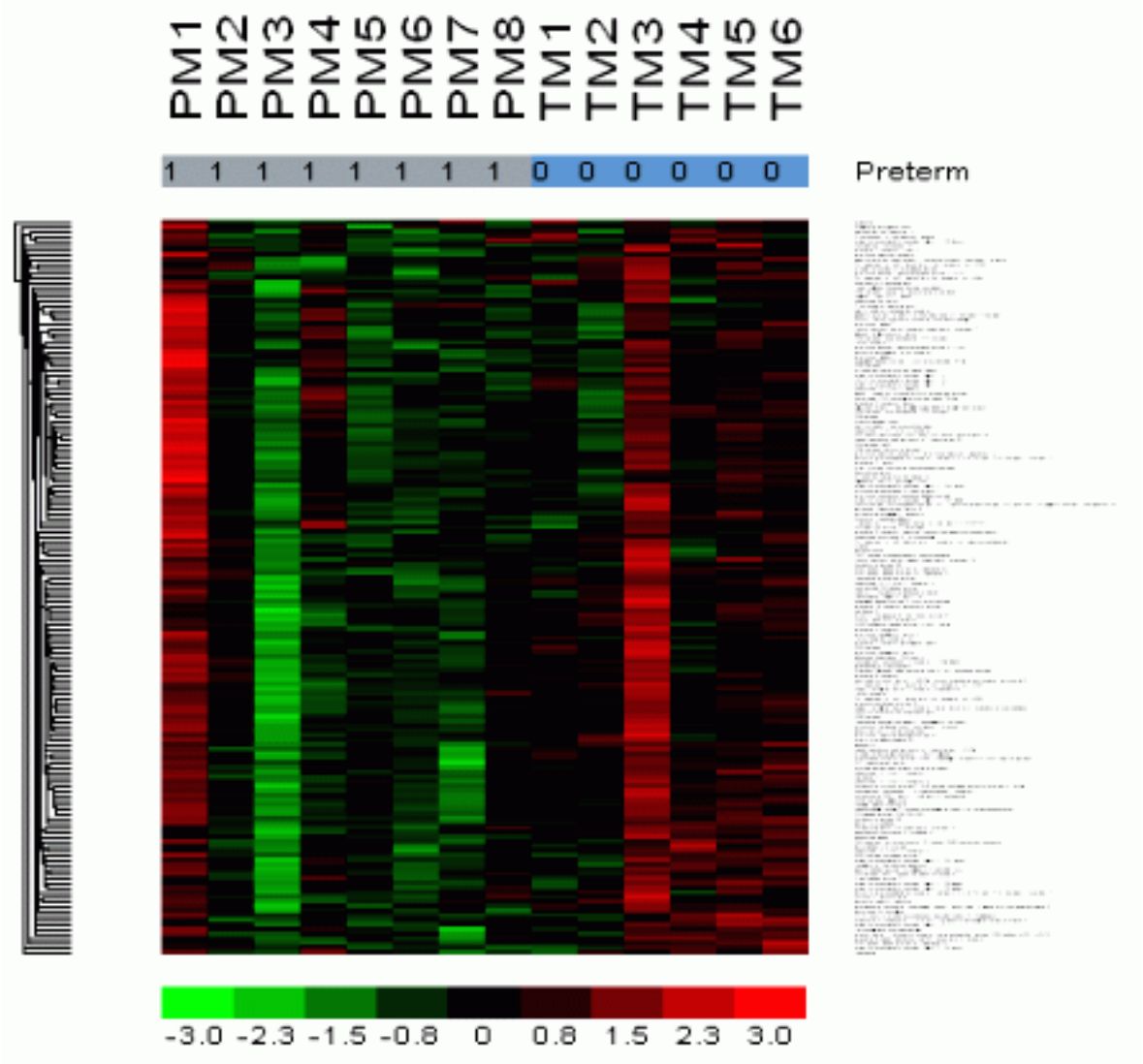
III. Semantic Web Decisions
   - A. What languages and tools?
   - B. Ontology design

The Problems

- The dramatic increase of bioinformatics data available in web-based systems and databases calls for novel processing methods.
- The high degree of complexity and heterogeneity of bioinformatics data and analysis requires integration methods.
- Information must be processed by a sequence of tools that often use different formats and data semantics.

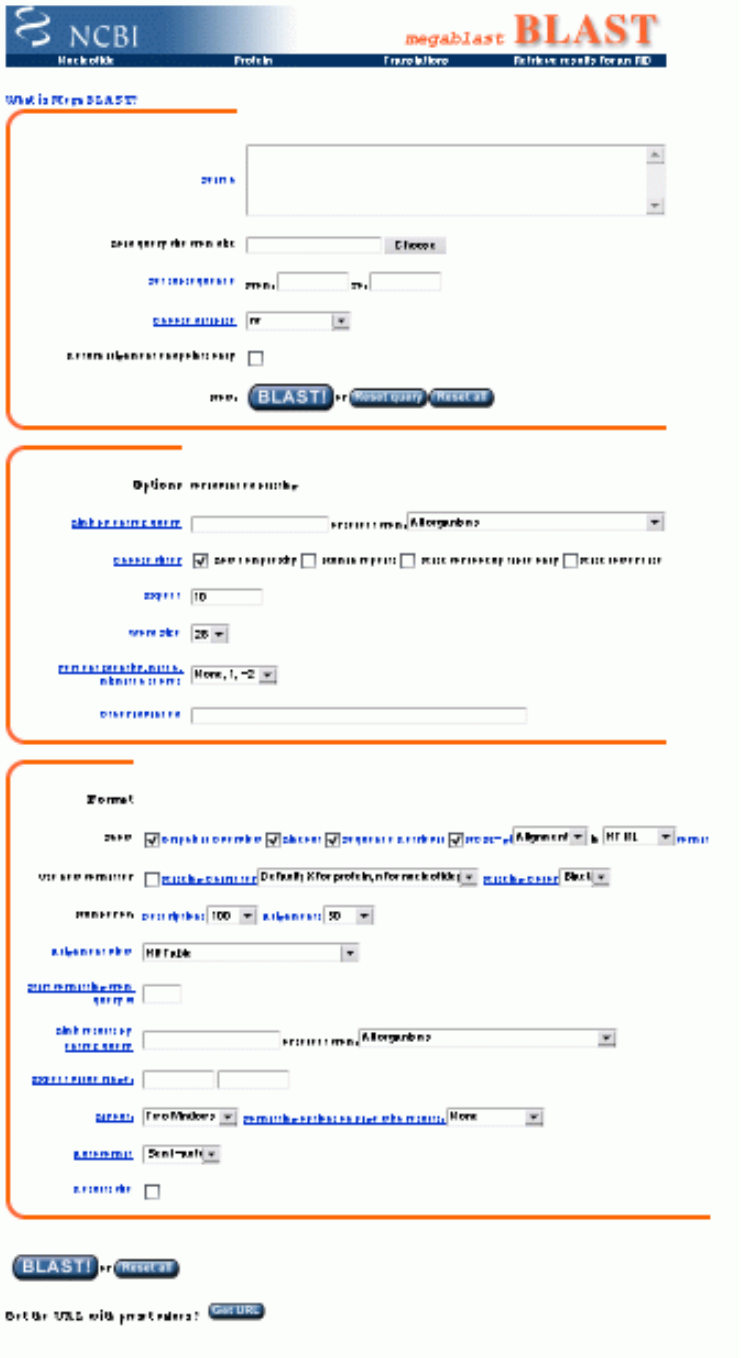


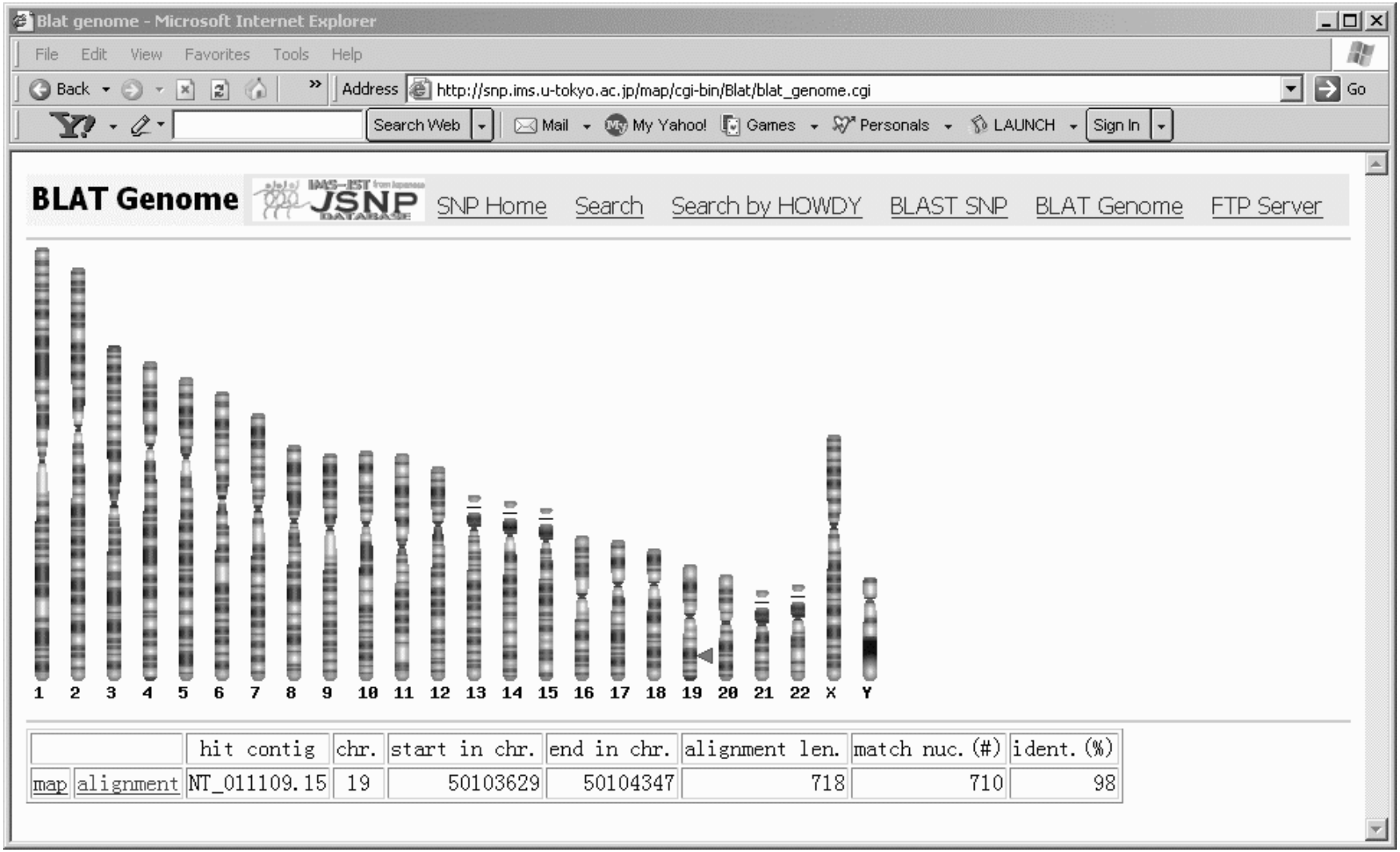
Example of a complex data format



Example of a complex data format

Example of a typical interface for a bioinformatics web service





Output window for a bioinformatics web service

Flat File Records

Consider the following records in flat file:

011500 18.66 0 0 62 46.271020111 25.220010
011500 26.93 0 1 63 68.951521001 32.651010
020100 33.95 1 0 65 92.532041101 18.930110
020100 17.38 0 0 67 50.351111100 42.160001

What do they mean?

Metadata

- The explanation of what data means is called metadata or “data about data.”
- For a flat file or database the metadata is called the schema.

```
NAME     LENGTH FORMAT            LABEL
instudy 6      MMDDYY      Date of randomization into
study
bmi      8      Num         Body Mass Index.
obesity 3      0=No 1=Yes Obesity (30.0 <= BMI)
ovrwt    8      0=No 1=Yes Overweight (25 <= BMI < 30)
Height   3      Num         Height (inches)
Wtkgs    8      Num         Weight (kilograms)
Weight   3      Num         Weight (pounds)
```

Record Structures

- A flat file is a collection of records.
- A record consists of fields.
- Each record in a flat file has the same number and kinds of fields as any other record in the same file.
- The schema of a flat file describes the structure (i.e., the kinds of fields) of each record.
- A schema is an example of an *ontology*.

Self-Describing Data

```
<Interview RandomizationDate="2000-01-15" BMI="18.66" Height="62"... />
<Interview RandomizationDate="2000-01-15" BMI="26.93" Height="63"... />
<Interview RandomizationDate="2000-02-01" BMI="33.95" Height="65"... />
<Interview RandomizationDate="2000-02-01" BMI="17.38" Height="67"... />
```

```
<ATTLIST Interview
      RandomizationDate  CDATA #REQUIRED
      BMI                CDATA #IMPLIED
      Height             CDATA #REQUIRED
>
```

The eXtensible Markup Language

- XML is a format for representing data.
- XML goes beyond flat files by allowing elements to contain other elements, forming a hierarchy.

XML	Flat Files
Element	Record
Attribute	Field
DTD	Schema

```
<bioml>
  <organism name="Homo sapiens (human)">
    <chromosome name="Chromosome 11" number="11">
      <locus name="HUMINS locus">
        <reference name="Sequence databases">
          <db_entry name="Genbank sequence" entry="v00565"
                    format="GENBANK"/>
          <db_entry name="EMBL sequence" format="EMBL" entry="V00565"/>
        </reference>
        <gene name="Insulin gene">
          <dna name="Complete HUMINS sequence" start="1" end="4992">
      1 ctcgaggggc ctagacattg ccctccagag agagcaccca acaccctcca ggcttgaccg
              ...
          </dna>
          <ddomain name="flanking domain" start="1" end="2185"/>
          <ddomain name="polymorphic domain" start="1340" end="1823"/>
          <ddomain name="Signal peptide" start="2424" end="2495"/>
          ...
          <exon name="Exon 1" start="2186" end="2227"/>
          <intron name="Intron 1" start="2228" end="2406"/>
         ...
        </gene>
      </locus>
    </chromosome>
  </organism>
</bioml>
```

XML Element Hierarchy

```
<bioml>
  <organism name="Homo sapiens (human)">
    <chromosome name="Chromosome 11" number="11">
      <locus name="HUMINS locus">
        <reference name="Sequence databases">
          <db_entry name="Genbank sequence" entry="v00565" format="GENBANK"/>
          <db_entry name="EMBL sequence" format="EMBL" entry="V00565"/>
        </reference>
        <gene name="Insulin gene">
          <dna name="Complete HUMINS sequence" start="1" end="4992">
          1 ctcgaggggc ctagacattg ccctccagag agagcaccca acaccctcca ggcttgaccg
          ...
          </dna>
          <ddomain name="flanking domain" start="1" end="2185"/>
          <ddomain name="polymorphic domain" start="1340" end="1823"/>
          <ddomain name="Signal peptide" start="2424" end="2495"/>
          <exon name="Exon 1" start="2186" end="2227"/>
          <intron name="Intron 1" start="2228" end="2406"/>
        </gene>
      </locus>
      <locus>
      ...
      </locus>
    </chromosome>
  </organism>
</bioml>
```

Specifying XML Hierarchies

A DTD can specify the kinds of element that can be contained in an element.

```
<ELEMENT locus (reference|gene)*>
<ELEMENT reference (db_entry)*>
<ELEMENT gene (dna,ddomain*,(exon|intron)*)>
```

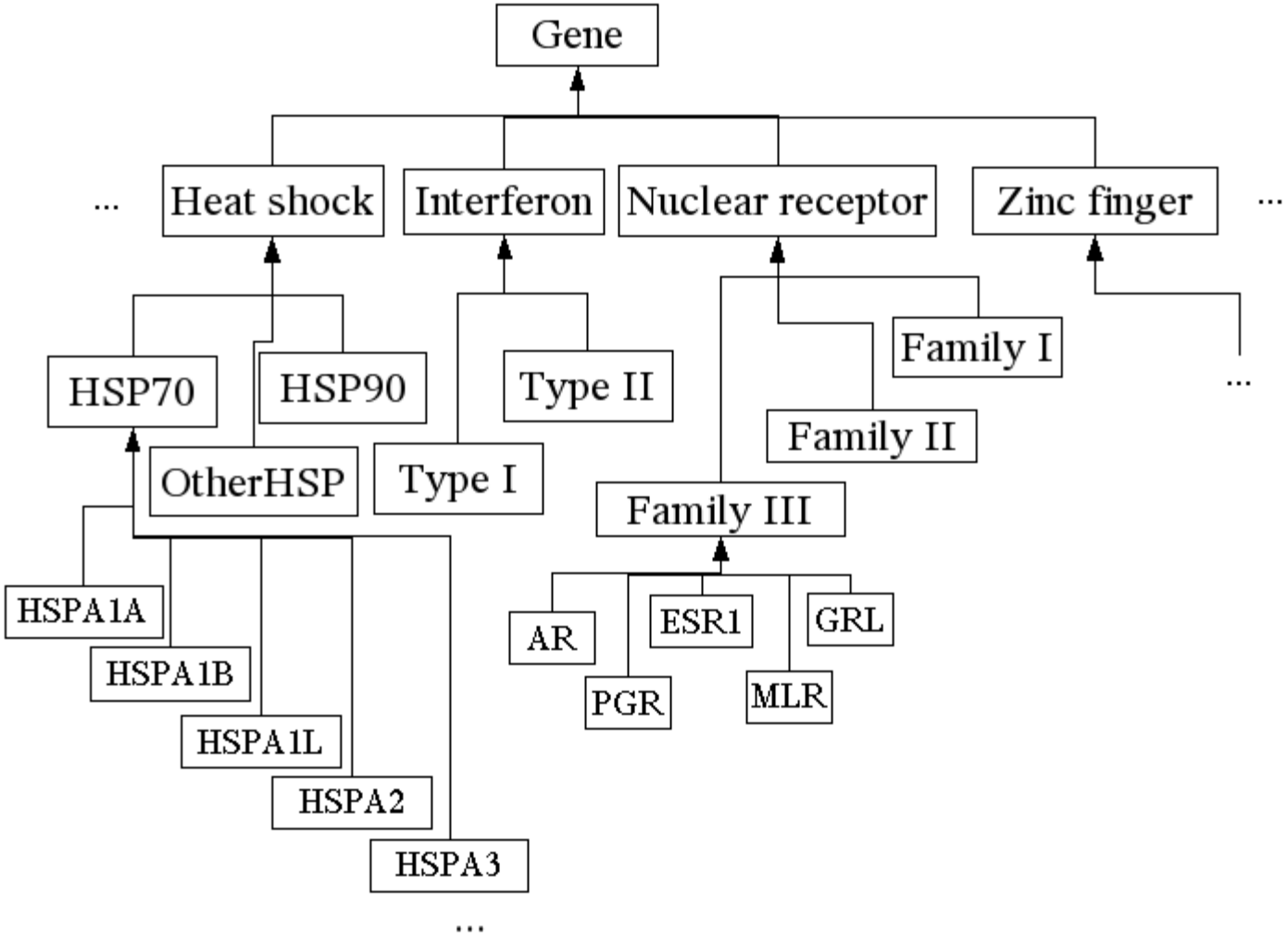
A locus element can contain any number of reference and gene elements.
A reference element can contain any number of db_entry elements.
A gene element must contain a dna element, followed by any number of ddomain elements, followed by any number of exon and intron elements.

Hierarchical Organization

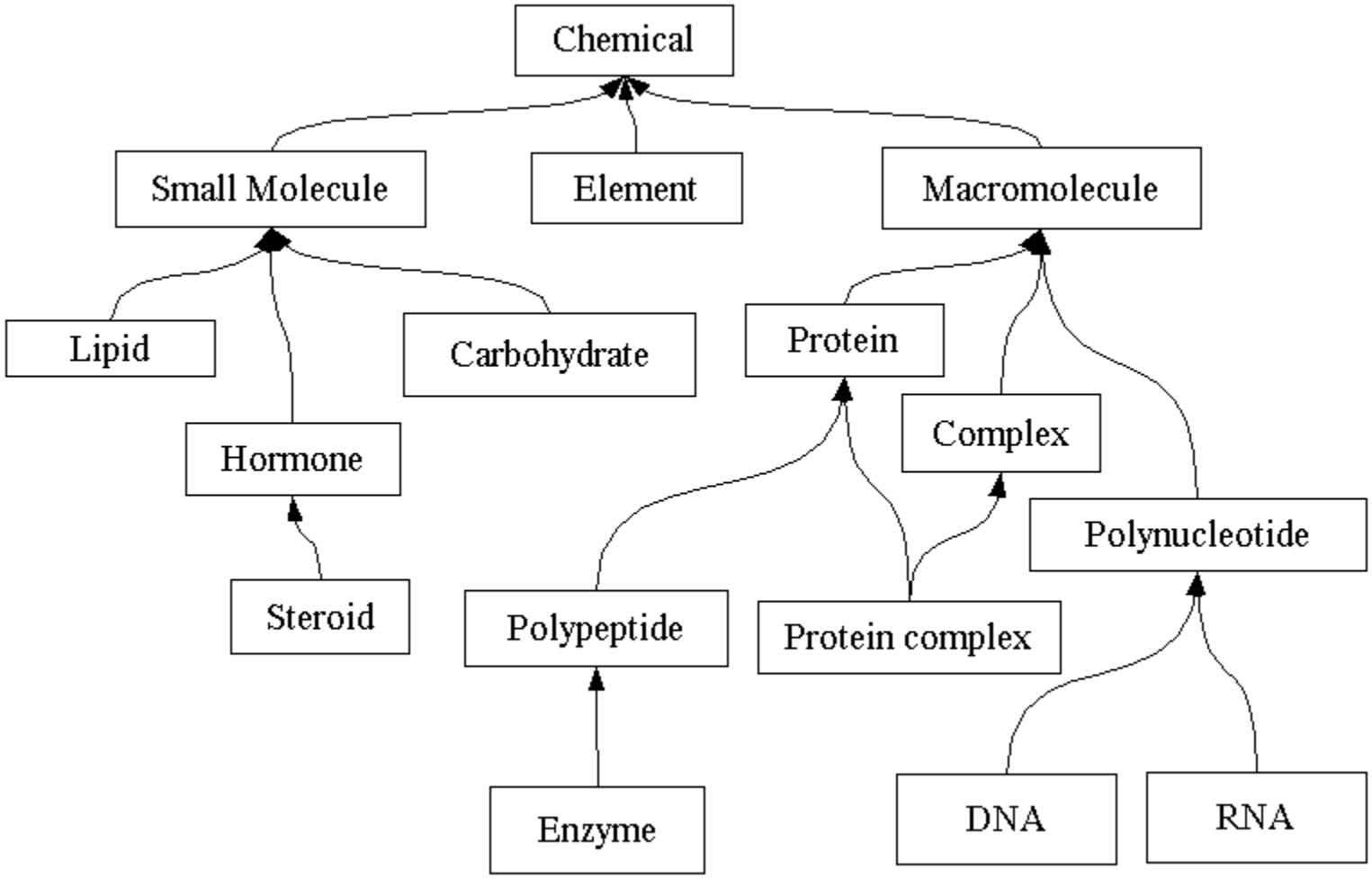
- XML elements are hierarchical: each element can contain other elements, that in turn can contain other elements, and so on.
- The relationship between an element and a contained element (*child element*), is implicit.
- In the example, a child element could be:
  - Physically contained (ddomain, exon, intron,…)
  - Stored in (db_entry)
  - Sequence of (dna)

The Meaning of a Hierarchy

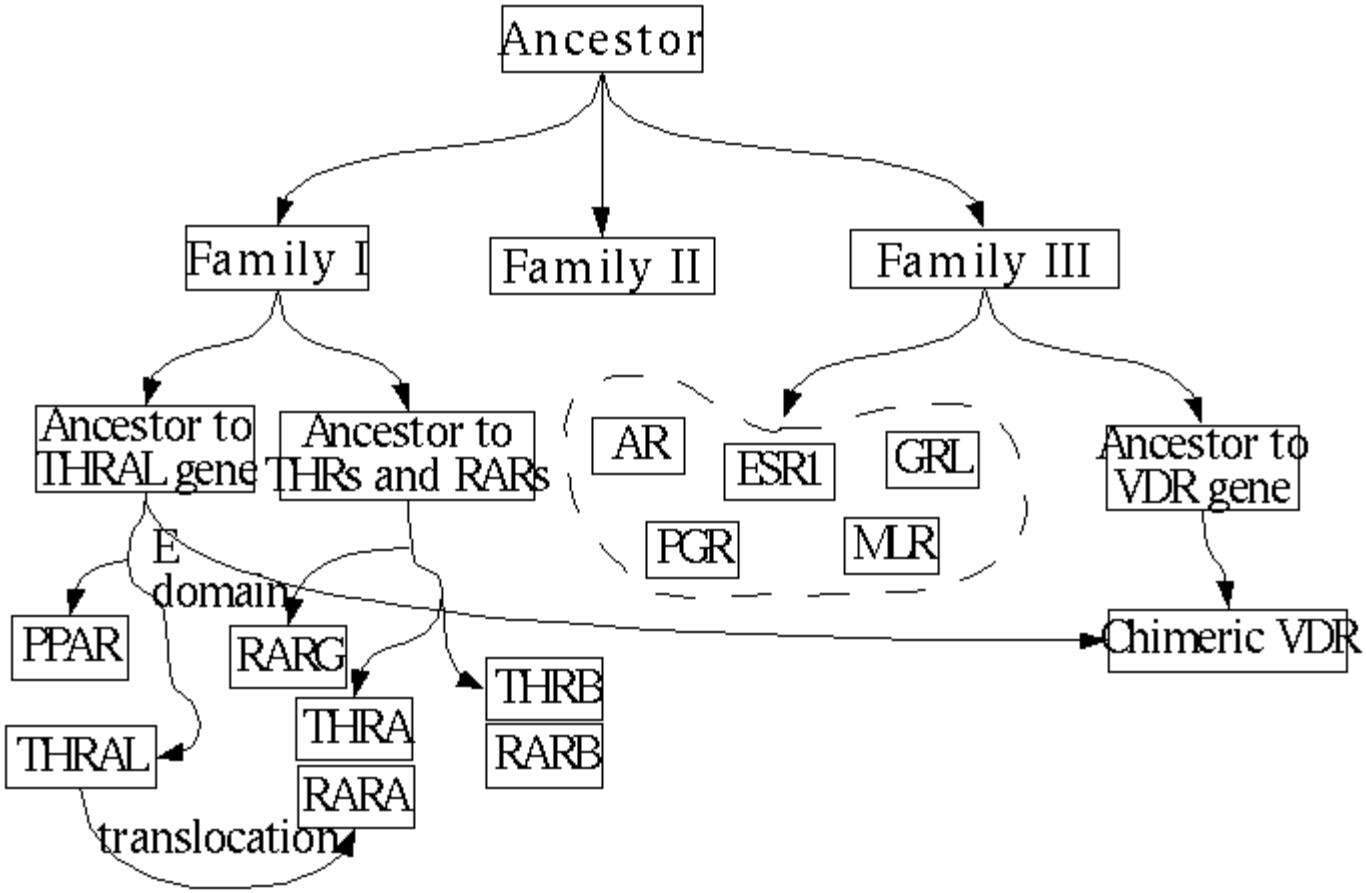
- Hierarchies can be based on many principles: subclass (subset), instance (member), or more complex relationships.
- Hierarchies to be based on several principles at the same time.
- XML hierarchies cannot represent these more general forms of hierarchy.


Gene
...
Heat shock
Interferon
Nuclear receptor
Zinc finger
...
HSP70
HSP90
Type II
Family I
...
Family II
OtherHSP
Type I
Family III
HSPA1A
HSPA1B
HSPA1L
HSPA2
HSPA3
...
AR
ESR1
GRL
PGR
MLR

Taxonomy



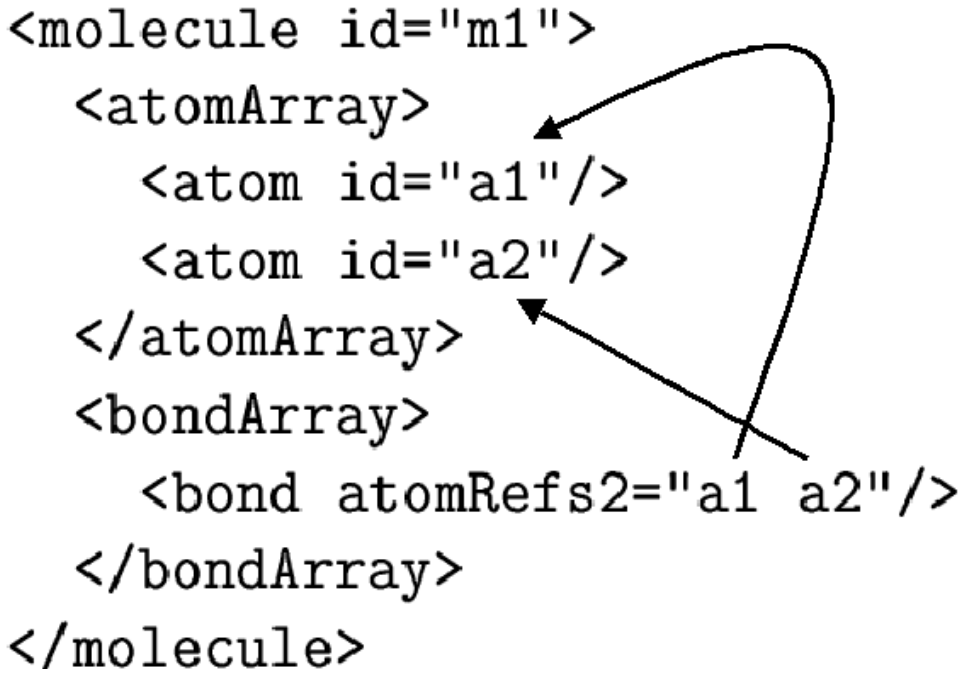
Subclass Hierarchy



Mixed Hierarchy

Non-Hierarchical Relationships

- Hierarchical relationships are represented by one element *contained* inside another one.
- Non-hierarchical relationships are represented using *reference attributes,* such as the two arrows in the diagram.
- Containment and reference are very different in XML.



Attribute Types

- Attributes generally contain a specific kind of data such as numbers, dates and codes.
- XML does not include any capability for specifying kinds of data like these.
- XML Schema (XSD) allows one to specify data structures and data types.
- The syntax for XSD differs from that for DTDs, but it is easy to convert from DTD to XSD using the dtd2xsd.pl Perl script.

XSD Basic Types

string Arbitrary text without embedded elements.

decimal A decimal number of any length and precision.

integer An integer of any length. This is a special case of decimal. There are many special cases of integer, such as positiveInteger and nonNegativeInteger.

date A Gregorian calendar date.

time An instant of time during the day, for example, 10:00.

dateTime A date and a time instance during that date.

duration A duration of time.

gYear A Gregorian year.

gYearMonth A Gregorian year and month in that year.

boolean Either true or false.

anyURI A web resource.

Attribute Types

One can introduce additional data types in three ways:

- **Restriction.** Restrict another data type using:
  - Upper and lower bounds
  - Patterns
  - Enumeration (e.g., standard codes)
- **Union.** Combine the values of several datatypes. Useful for adding special cases.
- **List.** A sequence of values.

The DNA Data Type

```
<xsd:simpleType name="DNAbase">
   <xsd:restriction
base="xsd:string">
      <xsd:pattern value="[ACGT]"/>
   </xsd:restriction>
 </xsd:simpleType>
 <simpleType name="DNASequence">
   <list itemType="DNABase"/>
 </simpleType>
```

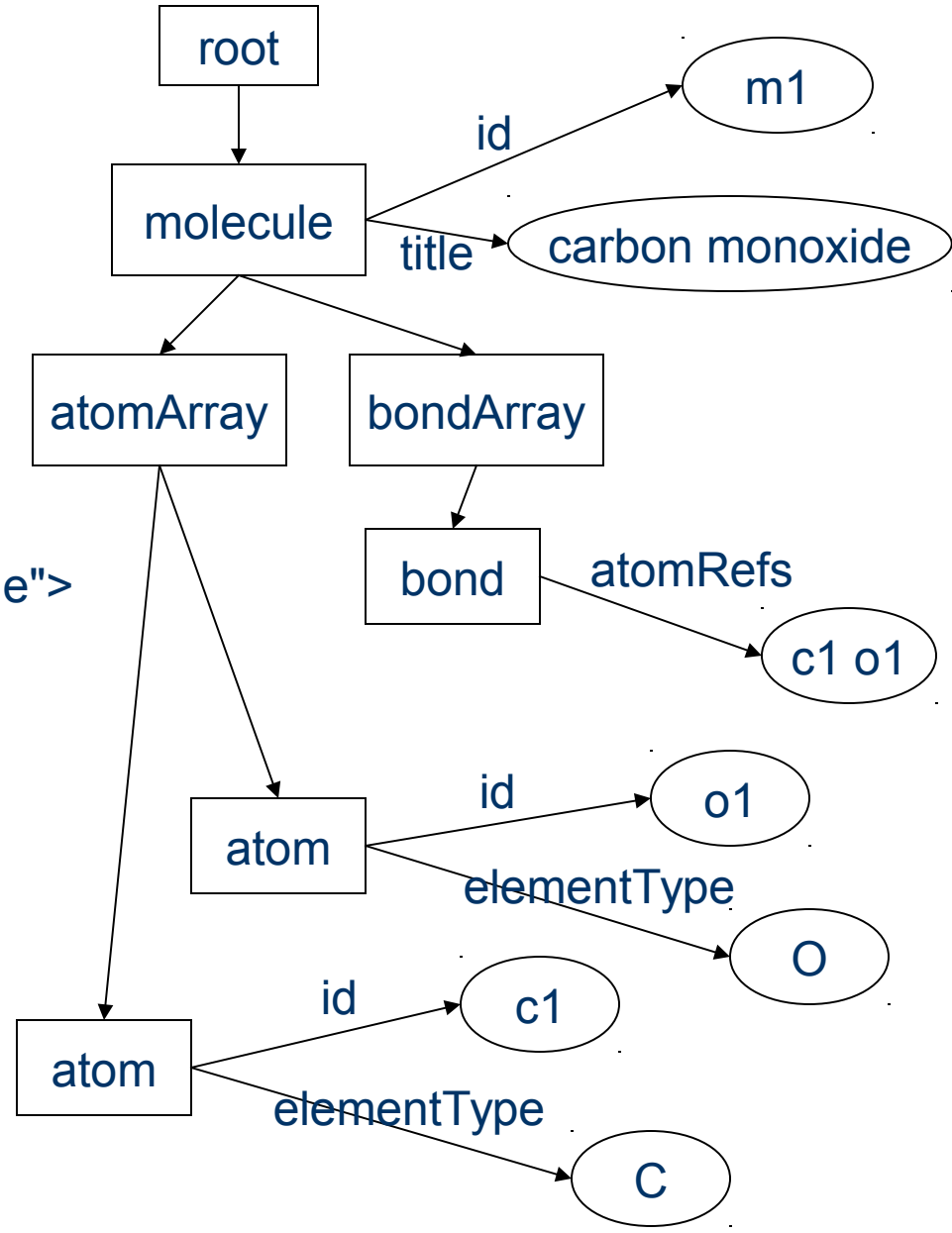
A single DNA base is specified by restricting the string data type. A sequence is specified as a list of bases.

Formal Semantics

- Semantics is primarily concerned with *sameness*. It determines that two entities are the same in spite of appearing to be different.
- Number semantics: 5.1, 5.10 and 05.1 are all the same number.
- DNA sequence semantics: cctggacct is the same as CCTGGACCT.
- XML document semantics is defined by infosets.

XML infoset for carbon monoxide

```
<molecule id="m1" title="carbon monoxide">
  <atomArray>
    <atom id="c1" elementType="C"/>
    <atom id="o1" elementType="O"/>
  </atomArray>
  <bondArray>
     <bond atomRefs="c1 o1"/>
  </bondArray>
</molecule>
```



XML Semantics

- The infoset contains two kinds of relationship:
  - Unlabeled hierarchical relationship link
  - Labeled attribute link
- The order of attributes does not matter. The infoset is the same no matter how they are arranged.
- The order of hierarchical links does matter. The infoset is different if the elements are in a different order.

Rule-Based Systems

- Rule-based programming is a distinct style from the more common procedural programming style.
- Rule engines logically infer facts from other facts, and so are a form of automated reasoning system.
- There are many other kinds of reasoning system such as theorem provers, constraint solvers, and business rule systems.

Kinds of Rule Engine

- Both forward- and backward-chaining rule engines require a set of rules and an initial knowledge base of facts.
- Forward-chaining rule engines apply rules which cause more facts to be asserted until no more rules apply. One can then query the knowledge base. The best known example is Jess.
- Backward-chaining rule engines begin with a query and attempt to satisfy it, proceeding backward from the query to the knowledge base. Prolog is the best known example of this style of rule engine.

RuleML

- The standard language for XML based rules.
- RuleML is supported by over 40 rule engines. See www.ruleml.org/#Participants-Systems.
- A rule has two parts:
  - The antecedent or *body* of the rule.
  - The consequent or *head* of the rule.
- When the antecedent is satisfied, the consequent is invoked (fired).
- The assertion of a new fact by the consequent is called *logical inference*.

PAK proteins serve as targets for the small GTP binding proteins Cdc42 and Rac.

```
<Implies>
  <head>
    <Atom>
      <opr><Rel>targets</Rel></opr>
      <Var>protein</Var>
      <Var>target</Var>
    </Atom>
  </head>
  <body>
    <And>
      <Atom>
        <opr><Rel>type</Rel></opr>
        <Var>target</Var>
        <Var>PAK1</Var>
      </Atom>
      <Or>
        <Atom>
          <opr><Rel>type</Rel></opr>
          <Var>protein</Var>
          <Var>Cdc42</Var>
        </Atom>
        <Atom>
          <opr><Rel>type</Rel></opr>
          <Var>protein</Var>
          <Var>Rac</Var>
        </Atom>
      </Or>
    </And>
  </body>
</Implies>
```

The Resource Description Framework

- RDF is a language for representing information about resources in the web.
- While RDF is expressed in XML, it has different semantics.
- Many tools exist for RDF, but it does not yet have the same level of support as XML.

XSD vs. RDF

- XML semantics based on infosets
- Easy to convert from DTD to XSD
- Support for data structures and types
- Element order is part of the semantics

- Different semantics based on RDF graphs
- Cannot easily convert from DTD to RDF
- Uses only XSD basic data types
- Ordering must be explicitly specified using a collection construct

XML vs. RDF Terminology

XML	RDF
Element Type	Class
Element Instance	Resource
Data attribute	DatatypeProperty
Reference attribute	ObjectProperty
Containment	Property

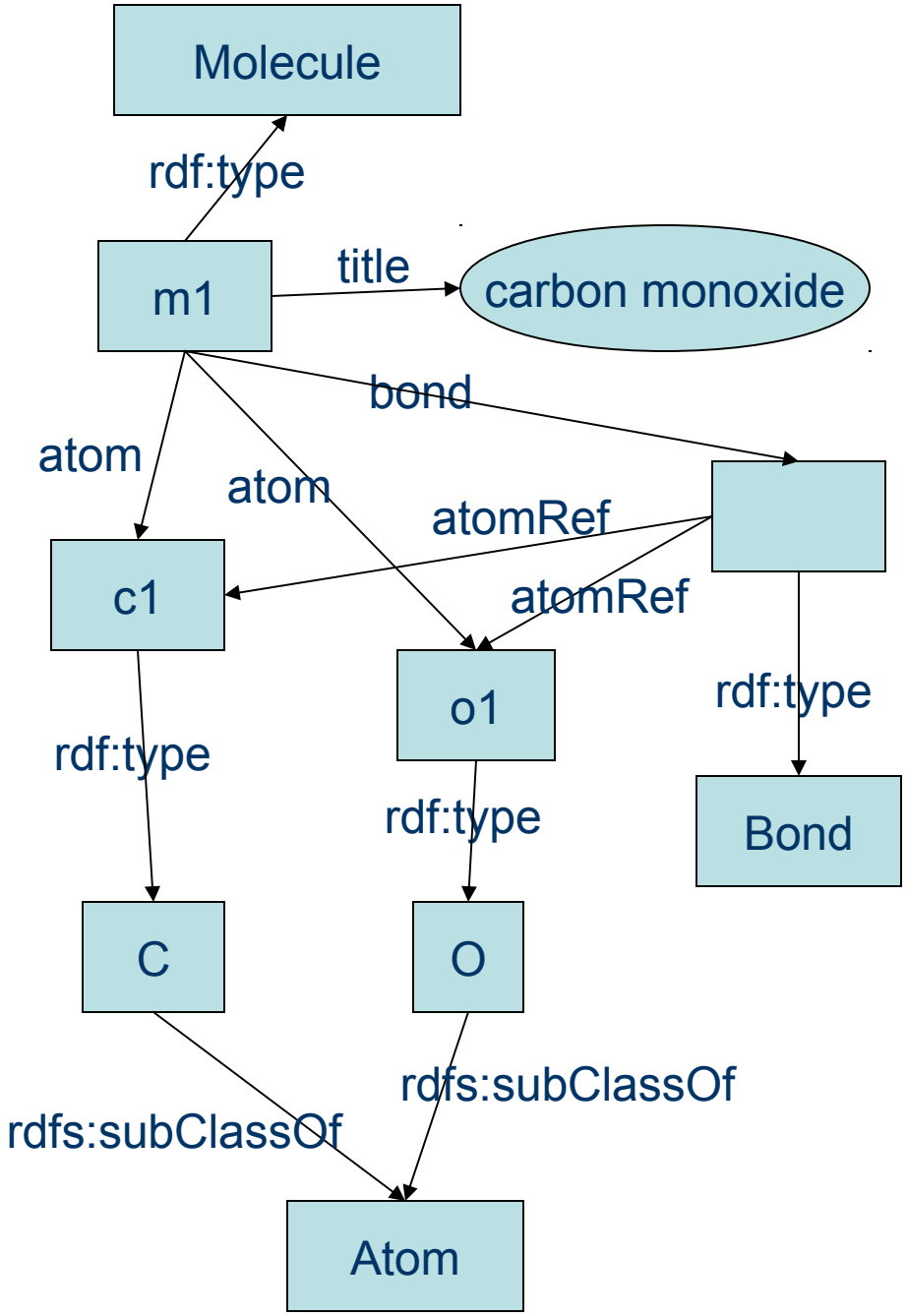
RDF Semantics

- All relationships are explicit and labeled with a property resource.
- The distinction in XML between attribute and containment is dropped, but the containment relationship must be labeled on a separate level. This is called *striping.*

```
...
    <locus name="HUMINS locus">
      <contains>
        <gene name="Insulin gene">
          <isStoredIn>
            <db_entry name="Genbank sequence" entry="v00565"
                      format="GENBANK"/>
            <db_entry name="EMBL sequence" format="EMBL"
                      entry="V00565"/>
          </isStoredIn>
          <isCitedBy>
            <db_entry name="Insulin gene sequence" format="MEDLINE"
                      entry="80120725"/>
            <db_entry name="Insulin mRNA sequence" format="MEDLINE"
                      entry="80236313"/>
            <db_entry name="Localization to Chromosome 11" format="MEDLINE"
                      entry="93364428"/>
          </isCitedBy>
          <hasSequence>
            <dna name="Complete HUMINS sequence" start="1" end="4992">
              1 ctcgaggggc ctagacattg ccctccagag agagcaccca acaccctcca ggcttgaccg
              ...
            </dna>
          </hasSequence>
        </gene>
      </contains>
    </locus>
...
```

RDF graph for carbon monoxide

```
<Molecule rdf:id="m1"
      title="carbon monoxide">
 <atom>
  <C rdf:id="c1"/>
  <O rdf:id="o1"/>
 </atom>
 <bond>
  <Bond>
   <atomRef rdf:resource="c1"/>
   <atomRef rdf:resource="o1"/>
  </Bond
 </bond>
</Molecule>
```



RDF Triples

- RDF graphs consist of edges called *triples* because they have three components: subject, predicate and object.
- The semantics of RDF is determined by the set of triples that are explicitly asserted or inferred.
- In the chemical example, some of the triples are:
  - (m1, rdf:type, cml:Molecule)
  - (m1, cml:title, “carbon monoxide”)
  - (m1, cml:atom, c1)
  - (m1, cml:atom, o1)
- Notice that properties are many-to-many relationships.

Notes on RDF Semantics

- There is no easy way to convert from XML to RDF because RDF makes explicit many relationships that are implicit in XML.
- In the chemical example, the element types are classes in RDF but have no special meaning to XML.
- The fact that n1 is an atom can be inferred from the fact that N is a subclass of Atom.
- The ordering of atoms in a molecule is significant in XML but not in RDF. RDF is therefore closer to the correct semantics.

RDF Rules

- Subclass rule. If a resource r has type A which is a subclass of B, then r has type B.
- Subproperty rule. Analogous to the subclass rule but for properties.
- Domain rule. If a property p has a domain D and s is the subject of a triple with property p, then s has type D.
- Range rule. If a property p has a range R and o is the object of a triple with property p, then o has type R.

RDF Rules

- While RDF has built-in rules, it has no mechanism for adding new rules.
- RuleML is the rule language for RDF.
- Many of the rule engines that support RuleML also support RDF. See www.ruleml.org.

The Web Ontology Language

- OWL is based on RDF and has three increasingly general levels: OWL Lite, OWL-DL, and OWL Full.
- OWL adds many new features to RDF:
  - Functional properties
  - Inverse functional properties (database keys)
  - Local domain and range constraints
  - General cardinality constraints
  - Inverse properties
  - Symmetric and transitive properties

Class Constructors

- OWL classes can be constructed from other classes in a variety of ways:
  - Intersection (Boolean AND)
  - Union (Boolean OR)
  - Complement (Boolean NOT)
  - Restriction
- Class construction is the basis for *description logic.*

Description Logic Example

- Concepts are generally defined in terms of other concepts. For example:

  The iridocorneal endothelial syndrome (ICE) is a disease characterized by corneal endothelium proliferation and migration, iris atrophy, corneal oedema and/or pigmentary iris nevi.

- ICE-Syndrome class is the intersection of:
  - The set of all diseases
  - The set of things that have at least one of the four symptoms

```
<owl:Class rdf:ID="ICE-Syndrome">
  <owl:intersectionOf parseType="Collection">
    <owl:Class rdf:about="#Disease"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#has-symptom"/>
      <owl:someValuesFrom>
        <owl:Class rdf:ID="ICE-Symptoms">
          <owl:oneOf parseType="Collection">
            <Symptom name="corneal endothelium proliferation and migration"/>
            <Symptom name="iris atrophy"/>
            <Symptom name="corneal oedema"/>
            <Symptom name="pigmentary iris nevi"/>
          </owl:oneOf>
        </owl:Class>
      </owl:someValuesFrom>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Example of Description Logic

OWL Semantics

- An OWL ontology defines a theory of the world. States of the world that are consistent with the theory are called *interpretations* of the theory.
- A fact that is true in every model is said to be *entailed* by the theory. Logical inference in OWL is defined by entailment.
- Entailment can be counter-intuitive, especially when it entails that two resources are the same.

OWL Semantics

- OWL semantics is defined by entailment, not by constraints as in databases.
- Another way to understand this distinction is that OWL assumes an open world, while databases assume a closed world.
- The next two slides show some examples of the distinction between these two.

Consider this definition:

A locus is a place on a chromosome where a gene is located.

The fact that a locus is on a chromosome leads to this OWL specification:

```
<rdfs:Class rdf:ID="Locus">
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty>
    <owl:ObjectProperty rdf:ID="locatedOn">
     <rdfs:range rdf:resource="Chromosome"/>
    </owl:ObjectProperty>
   </owl:onProperty>
   <owl:cardinality rdf:datatype="xsd:integer">1</owl:cardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
</rdfs:Class>
```

This says that a locus is located on exactly one chromosome.
Now suppose that a locus is accidentally placed on two chromosomes:

```
<Locus rdf:ID="HUMINS">
 <locatedOn rdf:resource="Chromosome11"/>
 <locatedOn rdf:resource="Chromo11"/>
</Locus>
```

Then these two chromosomes must be the same:

```
<Chromosome rdf:about="Chromosome11">
 <owl:sameAs rdf:resource="Chromo11"/>
</Chromosome>
```

Most other systems would have signaled a constraint violation.

Now suppose that a locus is not placed on any chromosome.
Then the locus is placed on a blank (anonymous) chromosome:

```
<Locus rdf:ID="HUMINS">
 <locatedOn>
  <Chromosome/>
 </locatedOn>
</Locus>
```

Most other systems would have signaled a constraint violation.

Open World vs. Closed World

- The advantage of the open world assumption is that it is more compatible with the web where one need not know all of the facts, and new facts are continually being added.
- The disadvantage is that operations (such as queries) are much more computationally complex.

Computational Complexity

- The various languages are progressively more complex.
- Operations (such as queries) in XML and RDF require polynomial time in the worst case.
- OWL Lite operations are much more difficult, requiring exponential time in the worst case.
- OWL-DL is even more difficult than OWL Lite. One can only show that an operation can be completed in a finite amount of time.
- OWL Full is the most difficult of all. An operation need not finish at all.

Phase Transitions

- In spite of these negative results, OWL is quite reasonable in practice.
- The reason for this phenomenon is that the hard cases are not randomly distributed, but rather concentrated in a small region of the problem space.
- The transition from problems that are easy to ones that are hard is known as a *phase transition.*

Outline

I. Semantic Web Languages
   A. Hierarchies and relationships
   B. Basic XML semantics
   C. Data semantics
   D. RDF semantics
   E. OWL semantics

II. Semantic Web Usage
   A. Ontology based information retrieval
   B. Transformation languages and tools
   C. Semantic Web services
   D. Bayesian Web: Combining logic and probability

III. Semantic Web Decisions
   A. What languages and tools?
   B. Ontology design

Ontologies for Information Retrieval

- Source of terminology
- RDF graph matching
- Queries based on formal logic

Using Ontologies for Formulating Queries

- Ontologies are an important source of terminology that can be used to formulate queries.
- Biological and medical ontologies can be so large and complex that specialized browsing and retrieval tools are necessary.
- Several browsers are now available for the UMLS: MeSH, Know-ME, Apelon DTS, SKIP, etc.
- One can use ontologies as a means of query modification when a query does not return satisfactory results.

RDF Graph Matching

- Graph matching is analogous to sequence matching, such as in BLAST.
- Translating natural language text to an RDF graph that captures meaning remains an unsolved problem, but reasonably good tools are available.
- Systems that use RDF graph matching are available. Such a system allows one to query a corpus such as PubMed using natural language.

RefSeq Genome DB

Search Unstructured Comments in Gene Annotations

Enter your query here:	What regulates the adhesiveness of integrins at the plasma membrane of lymphocytes, and is responsible for association of PSCDs with membranes?
Must **include**:	
Must **exclude:**	
Max. size of results:	20
	Run Query List Saved Queries

RefSeq Genome DB

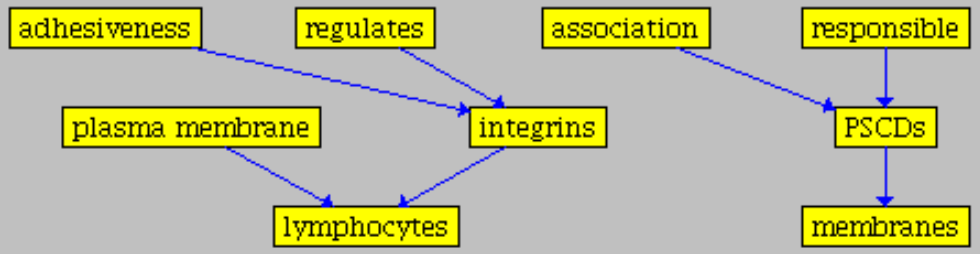
Search Unstructured Comments in Gene Annotations

The query returned 20 documents:

Main Query:

What regulates the adhesiveness of integrins at the plasma membrane of lymphocytes, and is responsible for association of PSCDs with membranes?

returned 20 documents



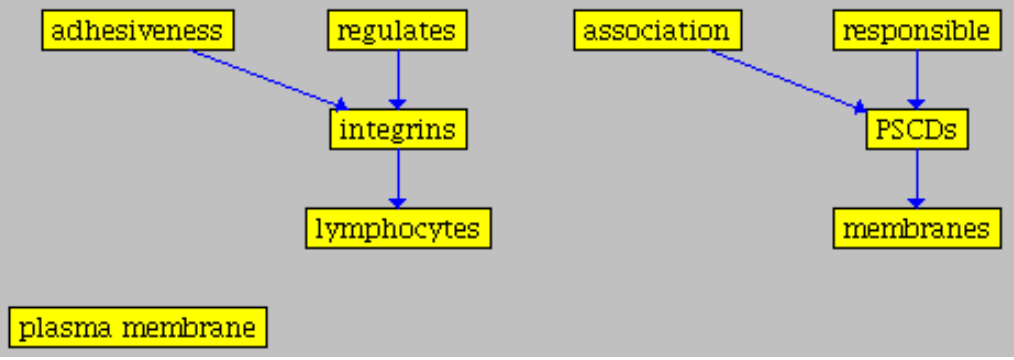
Must include:

Must exclude:

Max. size of results: 20

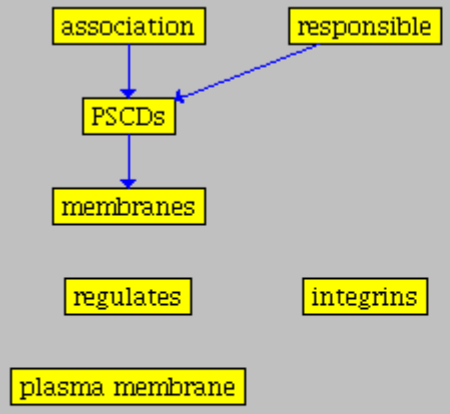
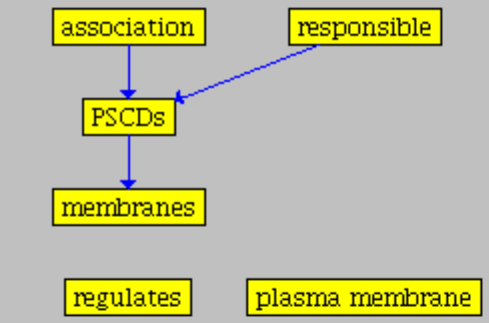
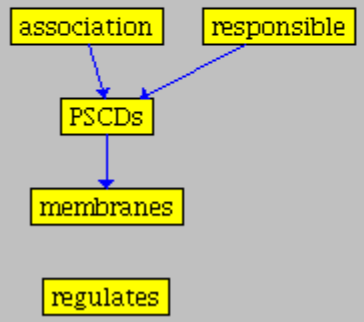
Run Query | Save Query | List Saved Queries

The documents which best matched your query are:



Homo sapiens pleckstrin homology, Sec7 and coiled/coil domains 1(cytohesin 1) (PSCD1), transcript variant 1, mRNA.

Homo sapiens pleckstrin homology, Sec7 and coiled/coil domains 1(cytohesin 1) (PSCD1), transcript variant 2, mRNA.

		Homo sapiens pleckstrin homology, Sec7 and coiled/coil domains 2 (cytohesin-2) (PSCD2), transcript variant 2, mRNA. Homo sapiens pleckstrin homology, Sec7 and coiled/coil domains 2 (cytohesin-2) (PSCD2), transcript variant 1, mRNA.
		Homo sapiens pleckstrin homology, Sec7 and coiled/coil domains 3 (PSCD3), mRNA.
		Homo sapiens pleckstrin homology, Sec7 and coiled/coil domains 4 (PSCD4), mRNA.

Proposed Web Query Languages

Ontology language	Query Language	Remarks
XML DTD and XSD	XQuery	Combines document navigation with an SQL-like query language
RDF	RDQL	Similar to SQL, specialized to the case of a 3-column table
OWL	OWL-QL	Requires a description logic theorem prover

XML Navigation Using XPath

- XPath is a language for navigating the hierarchical structure of an XML document.
- Navigation uses paths that are similar to the ones used to find files in a directory hierarchy.
- Navigation consists of steps, each of which specifies how to go from one node to the next. One can specify the direction in which to go (axis), the type of node desired (node test), and the particular node or nodes when there are several of the same type (selection).

XPath Features

- An axis can specify directions such as: down one level (child), down any number of levels (descendant), up one level (parent), up any number of levels (ancestor), and the top of the hierarchy (root).
- Node tests include: elements, attributes (distinguished using an at-sign) and text.
- One can select nodes using a variety of criteria which can be combined using Boolean operators.

Querying XML Using XQuery

- XQuery is the standard query language for processing XML documents.
- Every XPath expression is a valid query.
- A general query is made of four kinds of clause:
  - A **for** clause scans the result of an XPath expression, one node at a time.
  - A **where** clause selects which of the nodes scanned by the for clauses are to be used.
  - A **return** clause specifies the output of the query.
  - A **let** clause sets a variable to an intermediate result.

In the PubMed database, find all citations dealing with the therapeutic use of glutethimide. More precisely, find the citations that have"glutethimide" as a major topic descriptor, qualified by "therapeutic use."

```
for $citation in document("pubmed.xml")//MedlineCitation
where exists
 (for $heading in $citation//MeshHeading
  where $heading/DescriptorName/@MajorTopicYN="Y"
  and $heading/DescriptorName="Glutethimide"
  and $heading/QualifierName="therapeutic use"
  return $heading)
return $citation
```

Example of an PubMed query using XQuery

Transformation Languages and Tools

- Any programming language can be used for transformation. **Perl** is especially well suited for transformation to and from XML.
- The **XSLT** language is a rule-based language specifically designed for transformation from XML to XML.
- A series of databases and tools can be linked together in a data flow. The **myGrid** project has developed a workbench for specifying such data flows which has a large library of transformation modules.

Changing the Point of View

- Transformation is the means by which information in one format and for one purpose is adapted to another format for another purpose.
- Information transformation is also called repackaging or repurposing.
- A transformation step is performed using one of three main approaches:
  - Event-based parsing
  - Tree-based processing
  - Rule-based transformation

Processing XML Elements

- One way to process XML documents is to parse the document one element at a time. This is called the handlers style.
- In the handlers style, one specifies procedures that are invoked by the parser. Most commonly one specifies procedures to be invoked at the start of each element, for the text content of the element, and at the end of the element.
- A common way to design procedures is for the parameters to be in pairs: a parameter name and a parameter value. To make this easier to read, one should separate the parameter name from the parameter value with the -> symbol.
- The handlers style for parsing XML documents is efficient and fast but is only appropriate when the processing to be done is relatively simple.

The Document Object Model

- The whole document style of XML processing reads the entire document into a single Perl data structure.
- DOM methods are used to extract information from an XML document.
- The entities that occur in an XML document are represented by DOM nodes.
- DOM lists are used for holding a collection of DOM nodes.

Producing XML

- To convert non-XML data to the XML format, one can use the same techniques that apply to any kind of processing of text data. The XML document is just another kind of output format.
- The Perl Template Toolkit simplifies the production of XML documents by using a WYSIWIG style.
- The Perl Template Toolkit has its own language for iteration and selecting an item of a hash or array. The Template Toolkit language is much simpler than Perl because it has fewer features.

XSLT Templates

- An XSLT program consists of templates.
- A template either matches a specific kind of element or attribute or it uses a wild card to match many kinds of elements and attributes.
- A template performs an action on the matching elements and attributes.
- After transforming the matching element or attribute, a template can apply other templates to continue the transformation.

Programming in XSLT

- A transformation action occurs in a context: the element or attribute being transformed.
- The context is normally chosen in the same order in which the elements or attributes appear in the document, but which can be changed by using a sort command.
- The context is changed by using either an apply-templates (rule-based) command or a for-each (traditional iteration) command.

Navigation and Computation

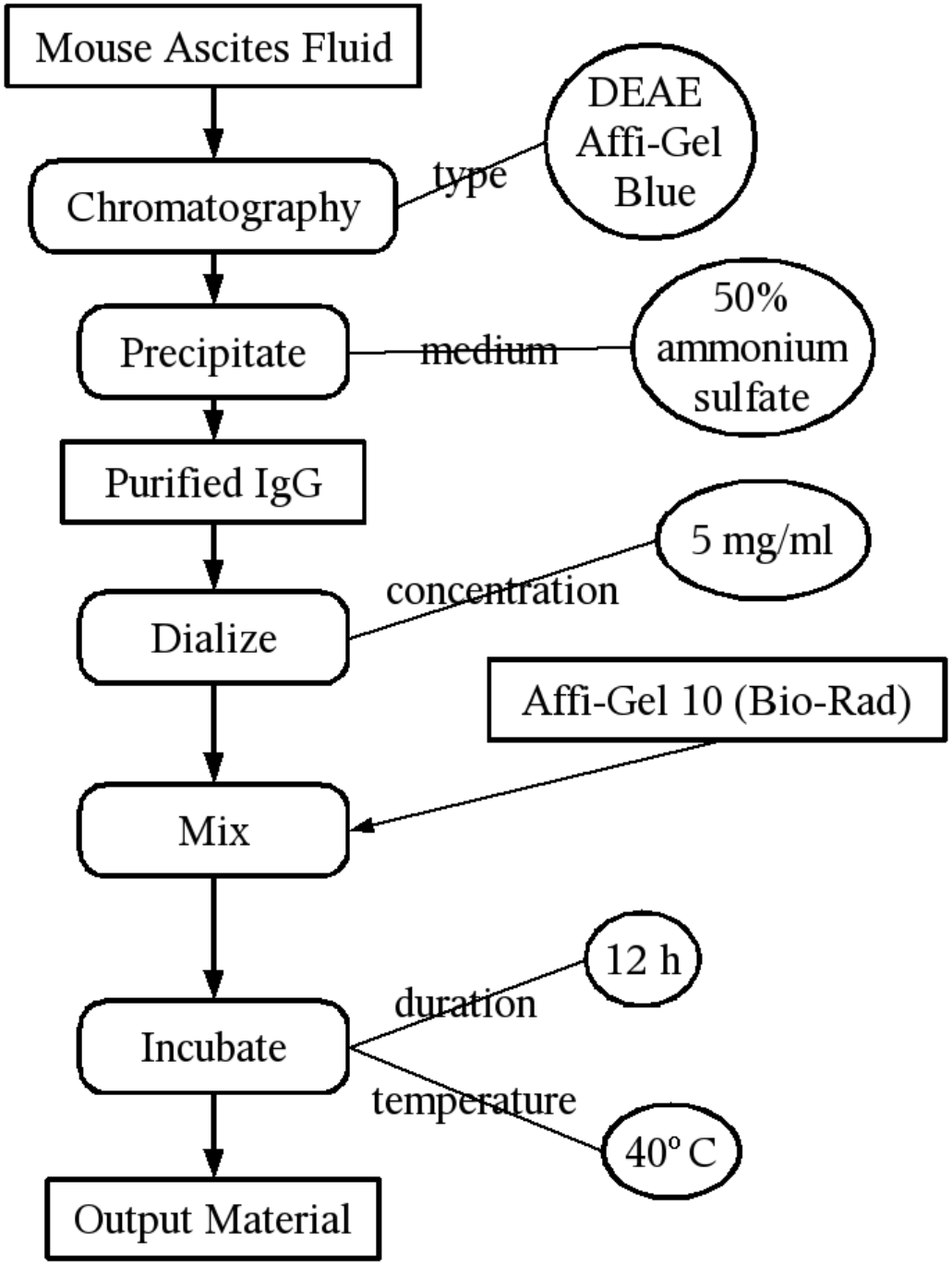
- XSLT navigation is the process of traveling from one element or attribute to another one in the document.
- Navigation is specified using XPath.
- Computations are specified using operators, such as `+, -, *`

Other Features of XSLT

- Conditionals are used for special cases.
- Precise Formatting
- Multiple Source Documents
- Procedural Programming

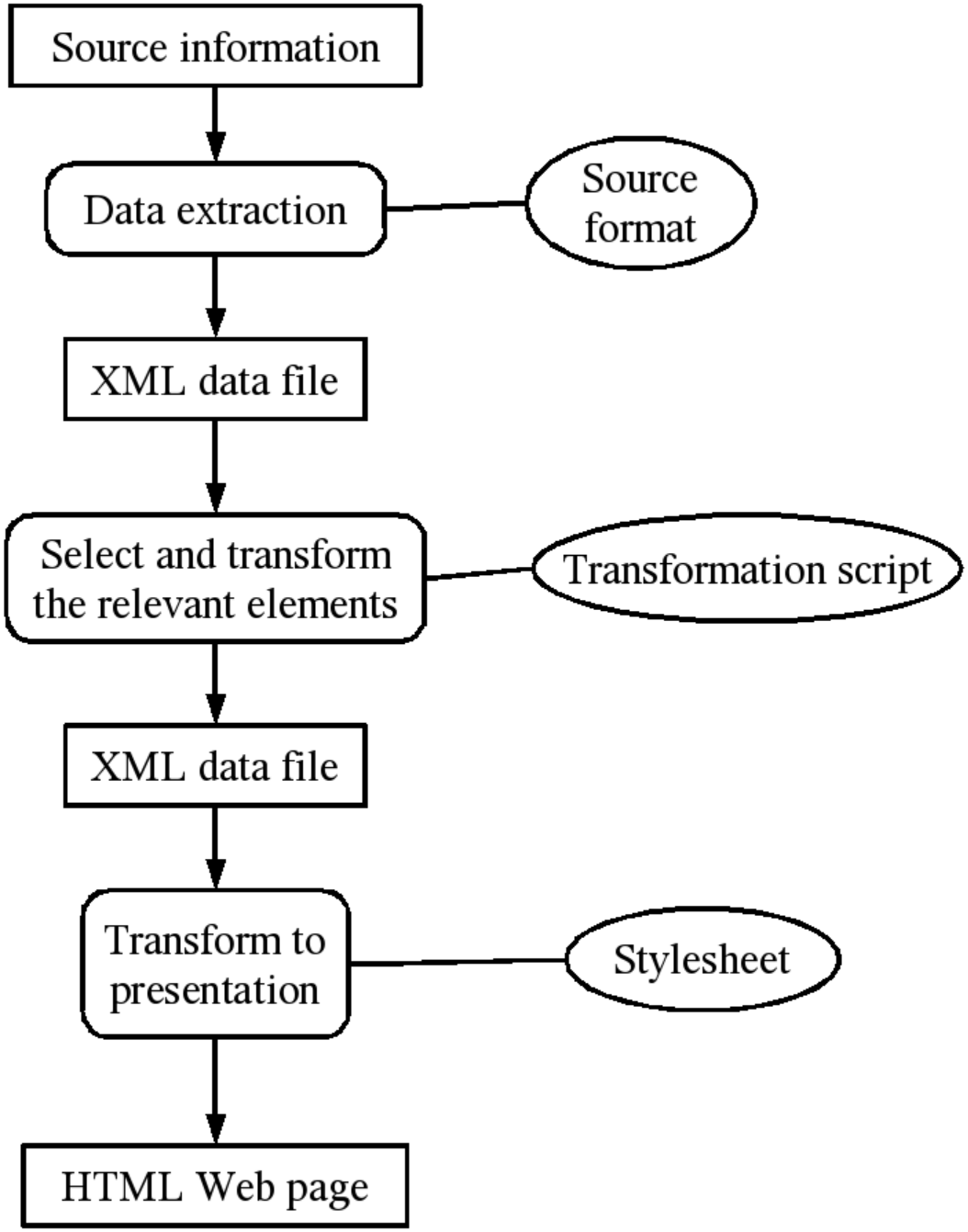
Experimental and Statistical Methods as Transformations

- Biology experiments and statistical analyses are transformation processes.
  - A biology experiment transforms biological and chemical materials into quantitative measurements.
  - A statistical analysis transforms survey information into statistical measurements.
- Information transformation is performed in a series of steps to reduce the overall effort.


Mouse Ascites Fluid
Chromatography
type
DEAE Affi-Gel Blue
Precipitate
medium
50% ammonium sulfate
Purified IgG
Dialize
concentration
5 mg/ml
Affi-Gel 10 (Bio-Rad)
Mix
Incubate
duration
12 h
temperature
40° C
Output Material

Presentation of Information

- Transformation is an effective means for controlling how data are presented.
- Information transformation is performed in a series of steps to reduce the overall effort and to separate concerns.
- Different individuals and groups of individuals are concerned with each step of the transformation process.

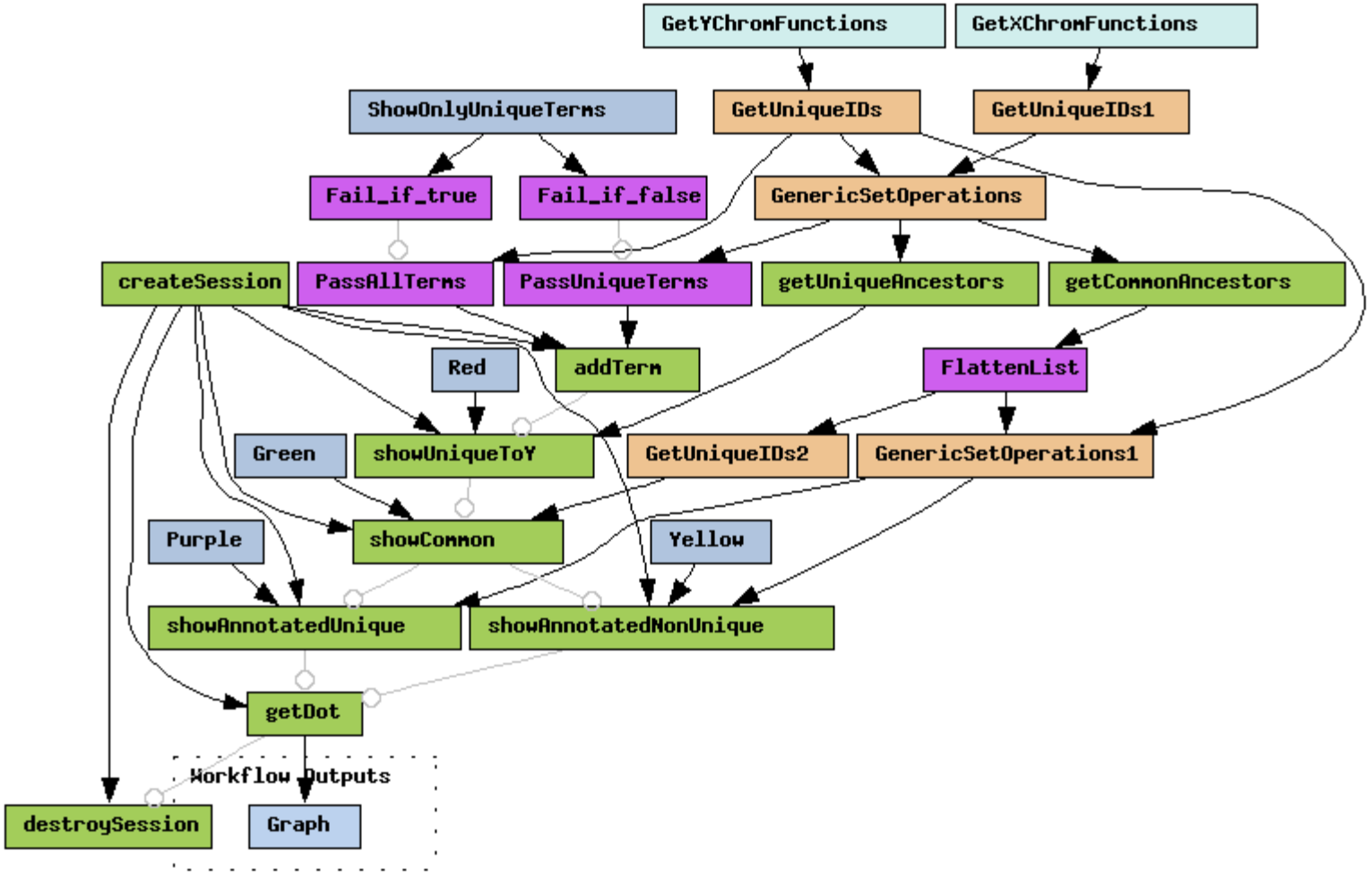

Source information
Data extraction
Source format
XML data file
Select and transform the relevant elements
Transformation script
XML data file
Transform to presentation
Stylesheet
HTML Web page

Automating Transformations

- Reconciling differing terminology has many names depending on the particular context where it is done, such as: ontology mediation, schema integration, data warehousing, virtual data integration, query discovery, and schema matching.
- Automated ontology mediation systems attempt to reduce manual effort, but they rarely provide a net gain.
- Most automated ontology mediation systems are still research prototypes.

The myGrid Project

- Taverna workbench supports the scientific process for in silico experiments.
  - Management
  - Sharing and reusing results
  - Recording their provenance and the methods used to generate results
- Workflows link together third party and local resources using database queries and web service protocols.



MyGrid Workflow

Web Services

- In the traditional programming model, all processing is done locally.
- Web services allow one to use programs that run on other machines.
- Tools have been developed that greatly reduce the effort of developing and offering web services.

Traditional Programming

- Traditional approach to application development
  - Write the program in a language such as Perl.
  - Compile the program.
  - Place the program and auxiliary files in a location where they can be found and downloaded.
- Using the application requires these steps:
  - Find the URL of the program.
  - Download the compiled program and auxiliary files.
  - Run it on the command line (or by clicking on an icon), specifying options as needed.

Web Services Approach

- Web service development has some new steps:
  - Write the program in some language like Perl.
  - Compile the program.
  - Describe the program using the WSDL.
  - Provide the application as a web service using a SOAP tool.
- Using the application proceeds as follows:
  - Find the URL of the web service.
  - Run on the command line using a WSIF tool.

Standards for Web Services

- Web Service Definition Language (WSDL)
  - Defines the web service
  - Written by the developer
- Simple Object Access Protocol (SOAP)
  - Format for running a service and receiving results
  - SOAP tools hide the underlying format and protocol from the developer and client.
- Universal Description, Discovery and Integration (UDDI)
  - Mechanism for advertising web services

Example of running blastp

- For example, one can download WU BLAST and run blastp with a command line like this:

```
blastp nr myquery.aa  v=10 b=100 filter=seg e=1e-10 nogaps
```

- WU BLAST is available as a web service at http://www.ebi.ac.uk/ws/WSWUBlast.wsdl
- It can be invoked using Apache WSIF like this:

```
java clients.DynamicInvoker http://www.ebi.ac.uk/ws/WSWUBlast.wsdl
  blastp nr myquery.aa  v=10 b=100 filter=seg e=1e-10 nogaps
```

Web Service Tools

- SOAP tools convert a local application to a web service:
  - SOAP::Lite is a Perl module available at CPAN.
  - Apache Axis is a Java based SOAP tool.
- SOAP tools also allow one to invoke a web service.
  - SOAP::Lite includes web service invocation.
  - Apache WSIF (part of the Axis project) is a Java based tool for web service invocation.

Semantic Web for Web Services

- Service Discovery
  - UDDI uses informal natural language descriptions.
  - The descriptions should be specified using ontologies.
- Service Definition
  - WSDL uses XSD for defining services, options and parameters.
  - RDF or OWL can be used to express the services using terminology in an ontology.

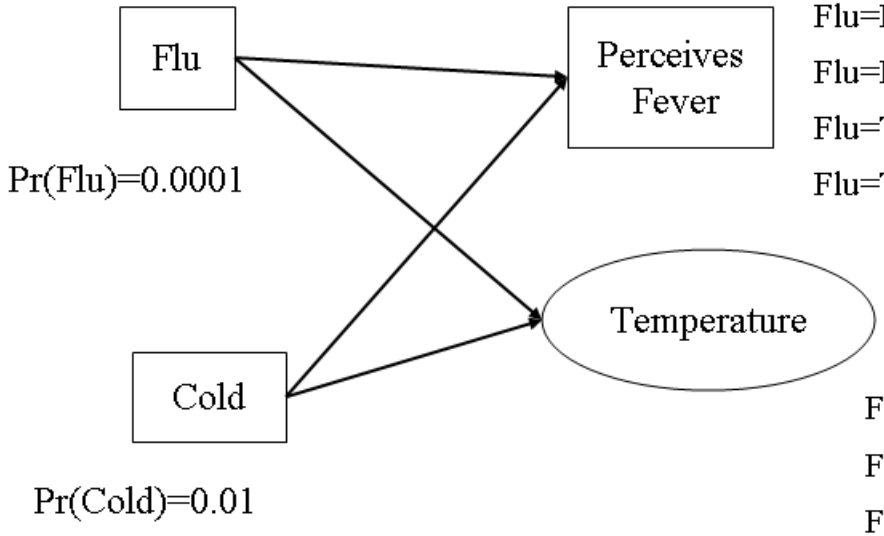
The Semantic Web and Uncertainty

- There are many sources of uncertainty, such as measurements, unmodeled variables, and subjectivity.
- The Semantic Web is based on formal logic for which one can only assert facts that are unambiguously certain.
- The Bayesian Web is a proposal to add reasoning about certainty to the Semantic Web.
- The basis for the Bayesian Web is the concept of a Bayesian network.

The Bayesian Network Formalism

- A BN is a graphical mechanism for specifying joint probability distributions (JPDs).
- The nodes of a BN are random variables.
- The edges of a BN represent stochastic dependencies.
- The graph of a BN must not have any directed cycles.
- Each node of a BN has an associated CPD.
- The JPD is the product of the CPDs.

Bayesian Network Specification

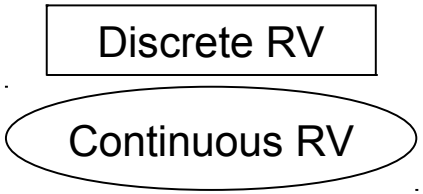


	Perceives Fever	
	TRUE	FALSE
Flu=F & Cold=F	0.01	0.99
Flu=F & Cold=T	0.10	0.90
Flu=T & Cold=F	0.90	0.10
Flu=T & Cold=T	0.95	0.05

	Temperature	
	Mean	Std Dev
Flu=F & Cold=F	37.0	0.5
Flu=F & Cold=T	37.5	1.0
Flu=T & Cold=F	39.0	2.0
Flu=T & Cold=T	39.5	2.5

CPDs:

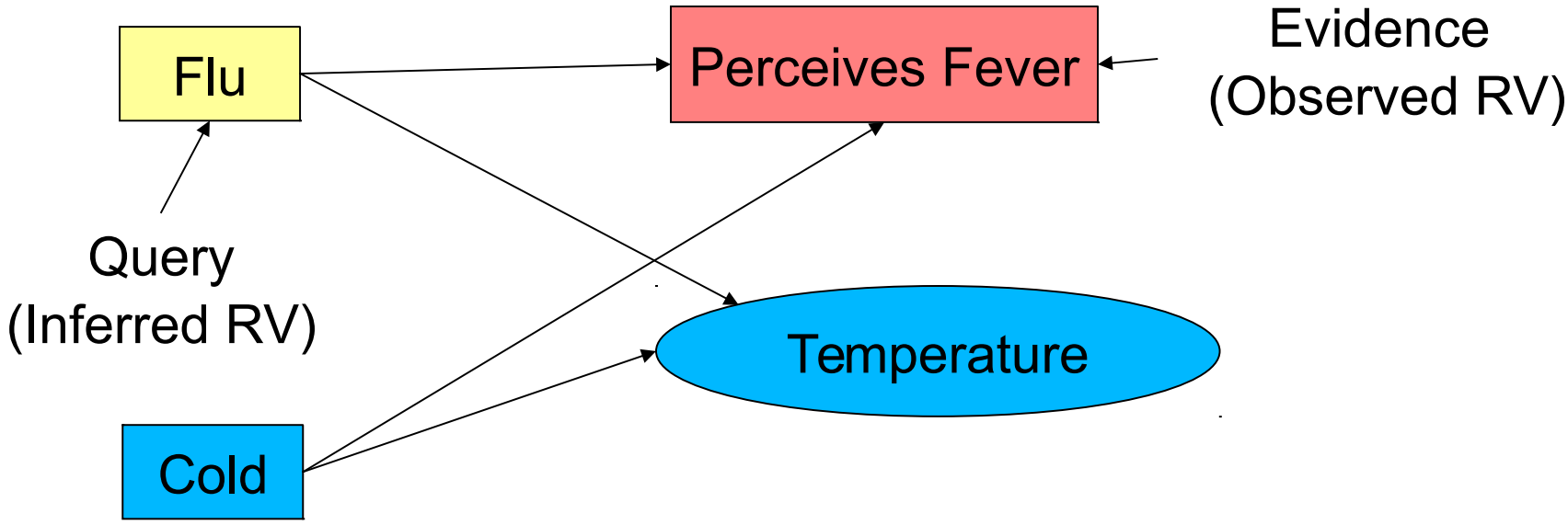
1. Perceives Fever given Flu and/or Cold.
2. Temperature given Flu and/or Cold.
3. Probability of Flu (unconditional).
4. Probability of Cold (unconditional).



Stochastic Inference

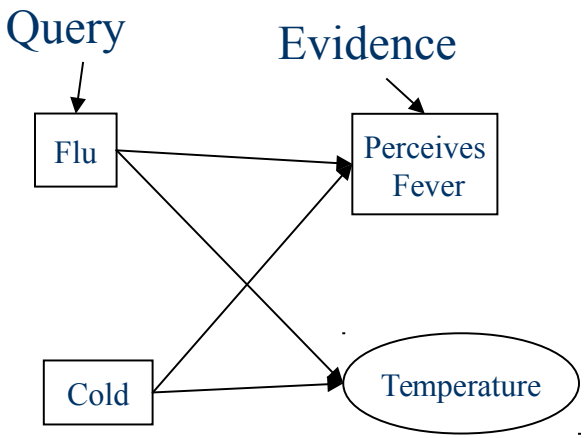
- Stochastic inference.
  - The main use of BNs.
  - Analogous to the process of logical inference and querying performed by rule engines.
  - Based on Bayes' law.
- Evidence
  - Can be either hard observations with no uncertainty or uncertain observations specified by a probability distribution.
  - Can be given for any nodes, and any nodes can be queried.
- Nodes can be continuous random variables, but inference in this case is more complicated.
- BNs can be augmented with other kinds of nodes, and used for making decisions based on stochastic inference.

Bayesian Network Inference

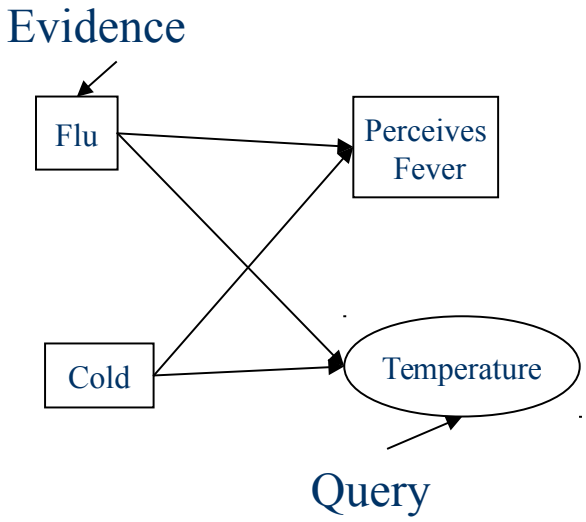


Inference is performed by observing some RVs (evidence) and computing the distribution of the RVs of interest (query).
The evidence can be a value or a probability distribution.
The BN combines the evidence probability distributions even when there are probabilistic dependencies.

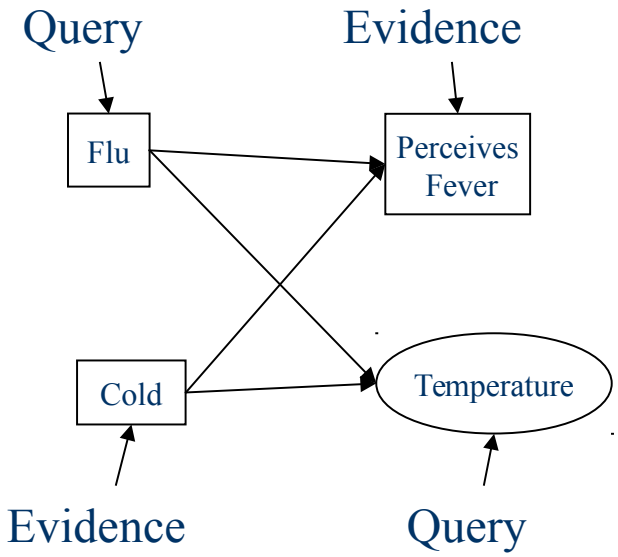
Bayesian Network Inference



Diagnostic Inference



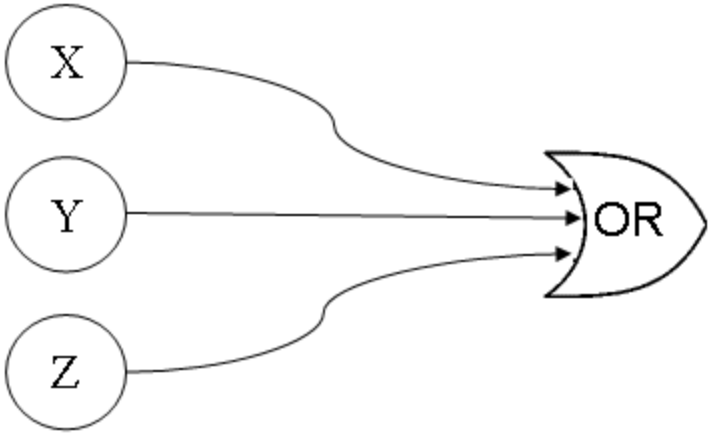
Causal Inference



Mixed Inference

BN Design Patterns

- One methodology for designing BNs is to use design patterns or idioms.
- Many BN design patterns have been identified, but most are only informally specified.



$$\Pr(W=\text{true}|X=\text{true},Y=\text{true},Z=\text{true}) = 1-q_x q_y q_z$$
$$\Pr(W=\text{true}|X=\text{true},Y=\text{false},Z=\text{true}) = 1-q_x q_z$$
$$\Pr(W=\text{true}|X=\text{true},Y=\text{true},Z=\text{false}) = 1-q_x q_y$$
$$\vdots \qquad \vdots$$
$$\Pr(W=\text{true}|X=\text{false},Y=\text{false},Z=\text{false}) = 0$$

The noisy OR-gate design pattern

Bayesian Web facilities

- Common interchange format
- Ability to refer to common variables (diseases, drugs, ...)
- Context specification
- Authentication and trust
- Open hierarchy of probability distribution types
- Component based construction of BNs
- BN inference engines
- Meta-analysis services

Bayesian Web Capabilities

- Use a BN developed by another group as easily as navigating from one Web page to another.
- Perform stochastic inference using information from one source and a BN from another.
- Combine BNs from the same or different sources.
- Reconcile and validate BNs.

Outline

I. Semantic Web Languages
   A. Hierarchies and relationships
   B. Basic XML semantics
   C. Data semantics
   D. RDF semantics
   E. OWL semantics

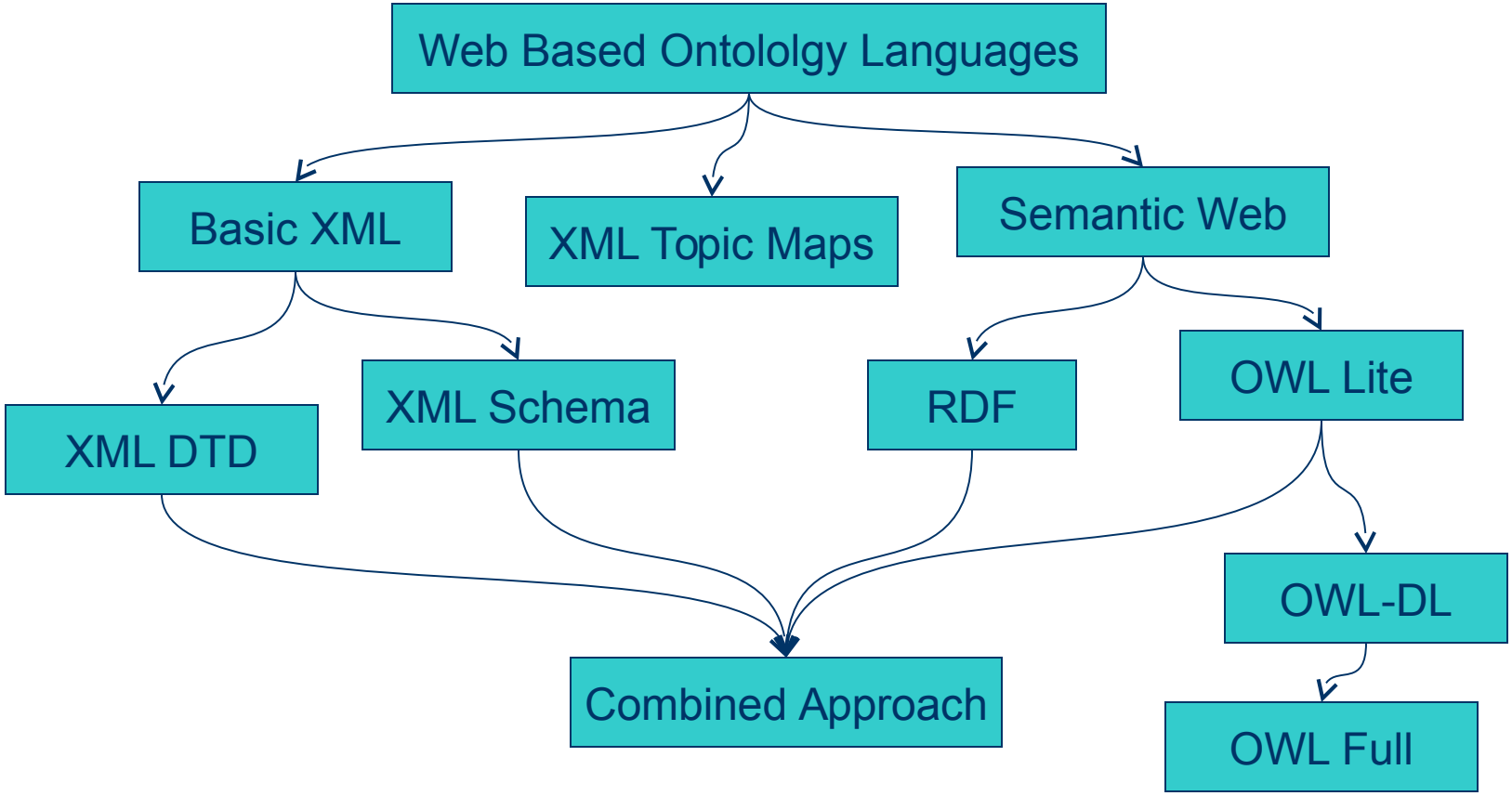
II. Semantic Web Usage
   A. Ontology based information retrieval
   B. Transformation languages and tools
   C. Semantic Web services
   D. Bayesian Web: Combining logic and probability

III. Semantic Web Decisions
   A. What languages and tools?
   B. Ontology design

Making Choices

- What language?
- What tools?
- Ontology design
- Service design
- These choices are interrelated.

Classification of Ontology Languages



Ontology Development Tools

- No explicit ontology or tool
- Automatic generation of the ontology from examples
- XML editor
- RDF editor
- OWL editor
- CASE tool adapted for ontology development

Building Bioinformatics Ontologies

- Before developing an ontology, one should understand its purpose.
- The purpose of the ontology should answer the following questions:
  - Why is it being developed?
  - What will be covered?
  - Who will use it?
  - How long will it be used?
  - How will it be used?

Acquiring Domain Knowledge

- Ontologies are based on domain knowledge.
- The main sources of domain knowledge for ontology development:
  - Statement of purpose of the ontology
  - Glossaries and dictionaries
  - Usage examples

Reusing Existing Ontologies

- Reusing existing ontologies can save time and improve quality.
- However, reusing an existing ontology is not always appropriate. One must balance the risks against the advantages.
- There are three ways to reuse an ontology:
  - Copy the ontology
  - Include the ontology
  - Import the ontology

Designing the Concept Hierarchy

- XML hierarchies are concerned with the structure of the document.
- RDF and OWL hierarchies are concerned with the subclass relationships.
- Concept hierarchies can be developed in several ways:
  - From the most general to the most specific (top-down)
  - From the most specific to the most general (bottom-up)
  - Starting at an intermediate, basic level (middle-out)

Hierarchy Design Techniques

- Uniform Hierarchy. Maintain a uniform structure throughout the hierarchy.
- Classes vs. Instances. Carefully distinguish instances from classes.
- Ontological Commitment. Keep the hierarchy as simple as possible, elaborating concepts only when necessary.
- Strict Taxonomies. Specify whether or not the hierarchy is strict (nonoverlapping).

Designing the Properties

- Classes vs. Property Values
- Domain and Range Constraints
- Cardinality Constraints
- Properties can be classified in several ways:
  - Attribute vs. relationship
  - Property values are data or resources
  - ntrinsic vs. extrinsic

Property Design Techniques

- Subclassification and property values can sometimes be used interchangeably. Choosing between the two design possibilities can be difficult.
- One should specify the domain and range of every property. They should be neither too general nor too specific.
- Cardinality constraints are important for ensuring the integrity of the knowledge base.
- Depending on the ontology language, one can specify other constraints, but these are less important.

Other Design Issues

- Namespaces
  - Select an appropriate domain.
  - Partition the ontology.
- Rule Language
  - RuleML, SWRL or some other language
- Rule Engine or Theorem Prover
  - Forward chaining or backward chaining
  - Description logic or general theorem prover
- Coping with computation complexity

Validating and Modifying the Ontology

- Ontology validation consists of the following activities:
  - Verify the fulfillment of the purpose.
  - Check that all usage examples are expressible.
  - Create examples that are consistent with the ontology, and determine whether they are meaningful.
  - Check that the ontology is formally consistent.
- Ontologies evolve over time due to changing requirements and circumstances.

To Learn More

For more information, see K. Baclawski and T. Niu, *Ontologies for Bioinformatics*, MIT Press, to appear in late August, 2005.

The website for this course and the book is ontobio.org.