VitaPad Extended

New approaches to editing pathway diagrams.

Matt Holford

Yale Center for Statistical Genomics and Proteomics

Outline

New areas of inquiry:

- Graphics enhancement
- XML persistence
- Database persistence
- Architecture enhancement

Graphics Enhancements

Incorporation of prefuse (Heer, et. al.):

- State-of-the-art toolkit for displaying complex interactive visual data
- Extensible design framework
- Dynamic control over layout algorithms
- Advanced interaction controls: zooming, fisheye scoping, animation

XML - The Problem

- We need an XML structure that represents both scientific and visual data in a single document and does so in a way that can be easily interpreted by other programs
- There is not currently a standard in universal currency for either pathways or for graph visualization

XML Standards for Pathways

- SBML, CellML
  - For simulation model exchange
  - Allow incorporation of formulas, stoichiometry, etc.
- PSI MI (MIF)
  - Limited to protein-protein interaction
- Transport formats (KGML, DIP, etc)
  - Inherently limited by their intent to import/export data from a specific database
  - KGML includes some graphical information

BioPAX

- Attempt to establish a universal standard for exchange of pathway information
- Large group of collaborators, including representatives from several major existing pathway databases
- Uses OWL (Ontological Web Language) an extension of XML with object-oriented functionality

BioPAX (Cont.)

- Multiple levels
  - Level 1 – Metabolic pathways
  - Level 2 – Adds molecular binding interactions
  - Level 3 – Signaling and regulation
  - Higher levels – Cell-level interactions and higher
- Still under development
  - Only level 1 complete
  - Level 2 targeted for June, 2005

XML Standards for Graphing

- Most pathway diagram programs use proprietary, often binary, persistence formats
- SVG (Scalable Vector Graphics) could be used to describe any graphics, but we would prefer something more specific to the task
- Handful of standards that predate XML still in public currency, e.g. Graphviz’s DOT and VCG’s GML, used by Cytoscape

XGMML

- Details
  - XML version of GML
  - Easily converted to and from GML
  - Not currently maintained or widely used (since 2001)
- Document structure
  - Simple yet flexible
  - Root element: graph
  - Children: Nodes and edges

XGMML (Cont.)

- Document Structure (Cont.)
  - Nodes and edges have a graphics element which contains elements to describe attributes such as colors, fonts and coordinates
  - All elements have a generic <att> element which can hold any information, including tags from another document structure
  - This allows us to easily incorporate any information that is not part of our generic network graph structure, e.g. scientific details, visual rendering information, user-defined extensions

The Solution

- We use XGMML as the basic document structure representing the network graph and most aspects of its appearance
- We use the <att> element for each node and edge to store scientific information about that node or edge in BioPAX

The Solution (Cont.)

- We make edge decorations an <att> of an edge and then use a graphics element to describe its appearance
- Any other information we wish to attach can be done so using the <att> element provided the application is able to handle to document format that is used

A Simplified Example

```
<graph id=“1” name=“Sample Graph”>
    <node id=“n1”>
        <graphics x=“100” y=“100” type=“Rectanlge” />
        <att type=“BioPAX”>
            <bp:smallMolecule rdf:ID=“sm001”>
                <bp:NAME>Pyruvate</bp:NAME>
            </bp:smallMolecule>
        </att>
    </node>
    <node id=“n2”>
        <graphics x=“200” y=“200” type=“Rectangle” />
        <att type=“BioPAX”>
            <bp:smallMolecule rdf:ID=“sm002”>
                <bp:NAME>L-Glutamate</bp:NAME>
            </bp:smalllMolecule>
        </att>
    </node>
```

```
    <edge source=“n1” target=“n2”>
        <graphics outline=“Red” arrow=“both” />
        <att type=“BioPAX”>
            <bp:pathwayStep rdfID=“ps001”>
                <bp:catalysis rdfID=“cs001” />
            </bp:pathwayStep>
        </att>
        <att type=“Decoration”>
            <decoration id=“d1”>
                <graphics fill=“Green” />
                <att type=BioPAX”>
                    <bp:protein rdf:ID=“p001” />
                </att>
            </decoration>
        </att>
    </edge>
</graph>
```

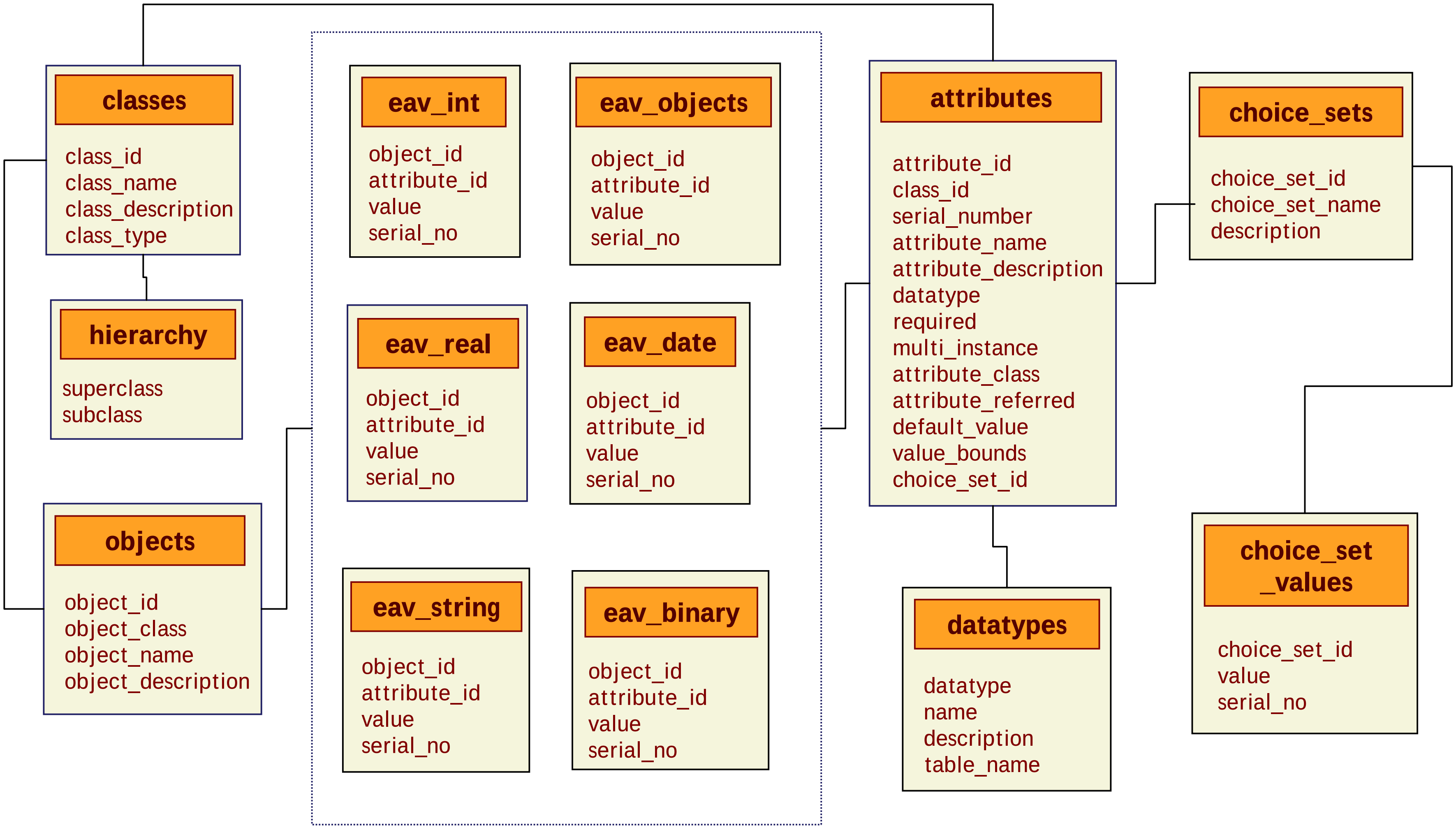
EAV/CR

- Database schema developed by Prakash Nadkarni, Luis Marenco and others at YCMI
- Highly abstract in structure; metadata driven
- Ideal for complex, heterogeneous data especially in areas of rapidly advancing knowledge

EAV/CR Schema

- Entity-Attribute-Value (EAV) model
  - Each entity (or class) has an arbitrary number of attributes, stored as rows in the database
  - Each attribute of an entity has a particular value
  - Domain-specific information not “hard-coded” into the table structure
- Classes and Relationships (CR)
  - Subclasses inherit attributes from superclasses

EAV/CR Schema Illustrated



classes

class_id
class_name
class_description
class_type

hierarchy

superclass
subclass

objects

object_id
object_class
object_name
object_description

eav_int

object_id
attribute_id
value
serial_no

eav_objects

object_id
attribute_id
value
serial_no

eav_real

object_id
attribute_id
value
serial_no

eav_date

object_id
attribute_id
value
serial_no

eav_string

object_id
attribute_id
value
serial_no

eav_binary

object_id
attribute_id
value
serial_no

attributes

attribute_id
class_id
serial_number
attribute_name
attribute_description
datatype
required
multi_instance
attribute_class
attribute_referred
default_value
value_bounds
choice_set_id

datatypes

datatype
name
description
table_name

choice_sets

choice_set_id
choice_set_name
description

choice_set _values

choice_set_id
value
serial_no

A Simple Example

Name: Dmitri

Age: 2 years

Weight: 12 lbs.

Color: Gray

Name: Gretchen

Age: 2 years

Weight: 9 lbs.

Color: Black

Traditional Method

Cats

Name	Age	Weight	Color
Dmitri	2	12	Gray/White
Gretchen	2	9	Black

EAV/CR Method

Classes

ID	Name
C1	Cat

Objects

ID	Name	Class
O1	Cat 1	C1
O2	Cat 2	C2

Eav_String

Object	Attribute	Value
O1	A1	Dmitri
O2	A1	Gretchen
O1	A3	Gray
O2	A3	Black

Attributes

ID	Class	Name	Datatype
A1	C1	Name	String
A2	C1	Age	Integer
A3	C1	Color	String
A4	C1	Weight	Integer

Eav_Int

Object	Attribute	Value
O1	A2	2
O2	A2	2
O1	A4	12
O2	A4	9

Adding New Data

Name: Dmitri

Age: 2 years

Weight: 12 lbs.

Color: Gray

Favorite Food: Mice

Name: Gretchen

Age: 2 years

Weight: 9 lbs.

Color: Black

Favorite Food: Ice Cream

Traditional Method

Cats

Name	Age	Weight	Color	Fav. Food
Dmitri	2	12	Gray/White	Mice
Gretchen	2	9	Black	Ice Cream

EAV/CR Method

Classes

ID	Name
C1	Cat

Objects

ID	Name	Class
O1	Cat 1	C1
O2	Cat 2	C2

Eav_String

Object	Attribute	Value
O1	A1	Dmitri
O2	A1	Gretchen
O1	A3	Gray
O2	A3	Black
O1	A5	Mice
O2	A5	Ice Cream

Attributes

ID	Class	Name	Datatype
A1	C1	Name	String
A2	C1	Age	Integer
A3	C1	Color	String
A4	C1	Weight	Integer
A5	C1	Fav. Food	String

Eav_Int

Object	Attribute	Value
O1	A2	2
O2	A2	2
O1	A4	12
O2	A4	9

EAV/CR and VitaPad

- Emphasis on flexibility
  - Pathways are highly subjective constructs
  - Constantly exposed to new data and new approaches to data
  - User goals will vary significantly
- Close match with XML and OWL
  - Transfer of information will be relatively easy because of the similarity in design and intent of EAV/CR and OWL

EAV/CR Programming Issues

- Need for a database engine
  - YCMI's library is in C#. This needed to be rewritten in Java.
  - Now we can take advantage of object-oriented design; YCMI code needed to be backwards-compatible with older VB-Script code
  - We use Hibernate, an Object Relational Mapping (ORM) tool to make transaction and query handling easier
- Need for UI tools
  - EAV/CR can be quite counter-intuitive to the unitiated
  - We are working on building user-friendly controls for browsing EAV/CR data into the VitaPad framework

Extending VitaPad

- Extensibility
  - We intend VitaPad to be responsive to changing scientific knowledge and user demands
  - Because these are unpredictable we must plan accordingly
- In software design, this is typically done in two ways
  - Plugins
  - Embedded scripting

Plugins and Scripting

- Some potential uses for plugins:
  - Support for a new file format
  - Interaction with another application
  - Incorporating a new way of displaying scientific data
- Some potential uses of scripting:
  - “Live” interaction with a running version of the application
  - Ability to create small custom tasks not originally part of program functionality

Jython

- Implementation of the Python language that runs on the Java Virtual Machine (JVM)
- Combines functionality of Python and Java libraries
- Python is widely used, especially in the biological community

A Theoretical Example

- The problem
  - We have datasets reflecting experimental conditions at 0, 12, 24, 48 and 72 hours
  - We want to reflect the change over time on the pathway graph
- The solution
  - We write a script that displays each set of values in a loop
  - We execute this script inside a running instance of VitaPad

Example (Cont.)

Pseudo-code

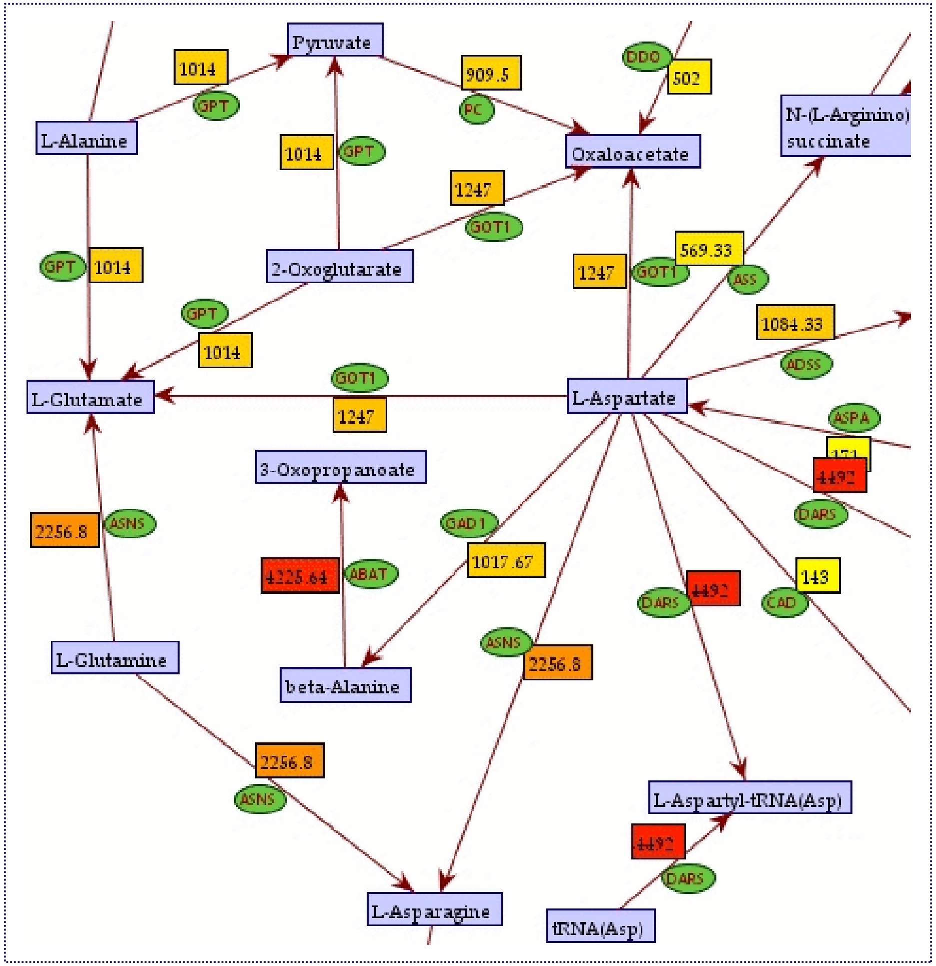
```
while (running)

    for (each experiment in list)
        dataMap = readExperiment()

        for(each key in dataMap.keys)
            dec = graph.getDec(dataMap.key)
            dec.setValue(dataMap.value)

        graph.repaint()

        sleep(15)
```


Pyruvate
1014
GPT
909.5
PC
DDO
502
L-Alanine
1014
GPT
Oxaloacetate
N-(L-Arginino)
succinate
1247
GOT1
GPT
1014
2-Oxoglutarate
1247
GOT1
569.33
ASS
GPT
1014
1084.33
ADSS
GOT1
L-Glutamate
1247
L-Aspartate
ASPA
171
4492
DARS
3-Oxopropanoate
2256.8
ASNS
GAD1
4225.64
ABAT
1017.67
DARS
4492
143
CAD
ASNS
2256.8
L-Glutamine
beta-Alanine
2256.8
ASNS
L-Aspartyl-tRNA(Asp)
4492
DARS
L-Asparagine
tRNA(Asp)