version

2.0

BILKENT UNIVERSITY

Bilkent Center for Bioinformatics

PATİKA*web*

User's Guide

BILKENT UNIVERSITY - CENTER FOR BIOINFORMATICS

PATIKA*web* User's Guide

Bilkent University • Center for Bioinformatics
Phone (312) 290 1401 • Fax (312) 266 4047

Table of Contents

Preface

TECHNICAL SPECIFICATIONS

PATIKA*web* requires one of the following Internet browsers:

- Microsoft Internet Explorer 5.5 or more,
- Mozilla Firefox 1.0 or more (not tested thoroughly),
- Apple Safari 1.0 or more (not tested thoroughly).

along with JDK 1.5 or higher to run. It does not impose any limitations on the size of the PATIKA graphs; they can contain as many biological objects or interactions as the available memory permits.

LAUNCHING THE WEB EDITION OF PATIKA

You do not need to install any application-specific software to run PATIKA*web*. Once you have agreed to the licensing terms, you can point your Internet browser to the following URL, and you're ready to go:

http://web.patika.org

ABOUT THE DOCUMENTATION

PATIKA*web* User's Guide is part of a document set published by the PATIKA group in Bilkent Center for Bioinformatics. Also refer to "A Quick Start Guide to Using PATIKA*web*", "PATIKA*web* Brochure" and "PATIKA*web* Key Features", all available in the Product Web Site.

Throughout this document, you may occasionally see parts highlighted in red. These parts are still to-do.

CONTACTING US

We believe user input is essential to successful product development. Please contact us with any questions or comments and visit our web site.

Product Web site	http://www.cs.bilkent.edu.tr/~patikaweb
Project Web site	http://www.patika.org
Email	patika@cs.bilkent.edu.tr
Mailing Address	Bilkent Center for Bioinformatics, Bilkent University, Ankara 06800, Turkey
Telephone and Fax	+90 (312) 290 1401 and +90 (312) 266 4047

Chapter 1

1 Overview

This chapter introduces the PATIKA Project and the tool PATIKA*web*, which is the Web edition of PATIKA that aims modeling and analysis of biological pathways through advanced visualization techniques.

1.1 Motivation and Goals

Availability of the sequences of entire genomes shifts the scientific curiosity toward the identification of function of the genomes in large scale as in genome studies. Currently, data produced about cellular processes at molecular level has been accumulating with an accelerating rate as a result of proteomics studies. In this regard, it is essential to develop tools for storing, integrating, accessing, and analyzing this data effectively. The PATIKA Project (www.patika.org) aims to do exactly this through following studies:

- Define an ontology for a comprehensive representation of cellular pathways.
- Develop software tools and construct an associated database using this ontology and providing an effective environment for pathway data integration, storage, access, visualization and analysis.
- Design methods for automatic population and annotation of the pathway database.
- Design methods for inferring pathway activity using temporal data such as gene expression data.
- Develop techniques for effective visualization of pathway and gene expression data.

PATIKA*web* is one of the tools resulting from these studies. PATIKA*web* provides a Web service for retrieving and analyzing biological pathways in the PATIKA database, which currently contains data integrated from popular public pathway databases such as Reactome. It features a user-friendly interface, dynamic visualization and automated layout, advanced graph-theoretic queries for extracting biologically important phenomena and exporting facilities to various exchange formats as well as static images.

1.2 Overall Architecture

PATIKA*web* has a distributed architecture, where the server is composed of a database component and an application server (Figure 1-1).

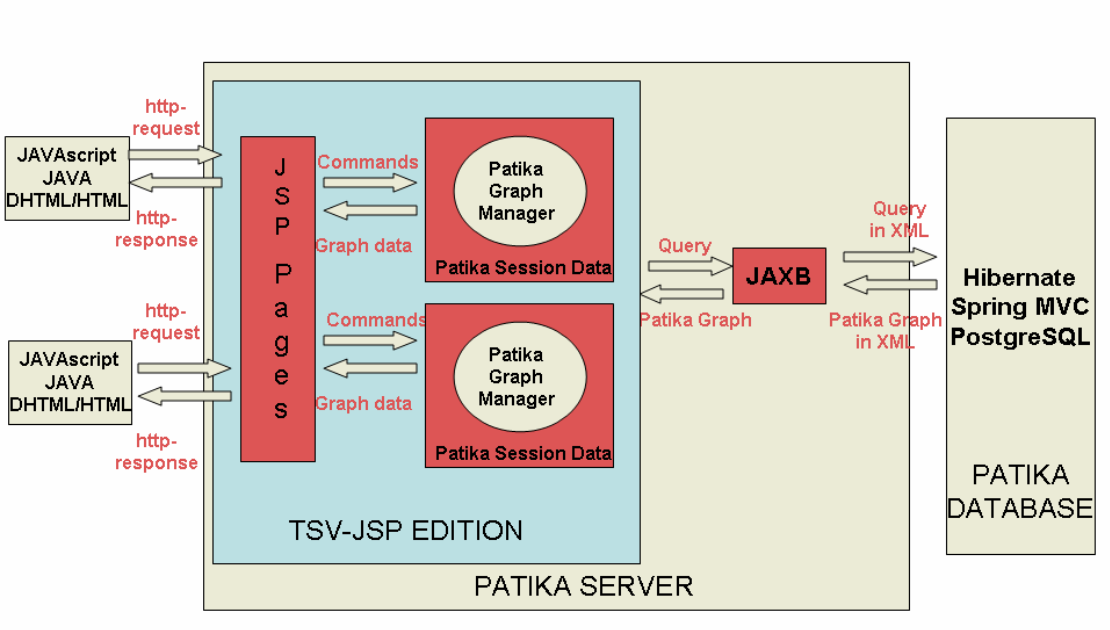


Figure 1-1 An overview of the PATIKA*web* system architecture

1.3 Basics of Ontology

PATIKA ontology is a qualitative ontology based on biochemical paradigm, and extends it to introduce biological annotations and abstractions. Every first class object in the PATIKA ontology is a PATIKA object, which stores common information, such as name, description, data sources and GO terms. PATIKA names comply with external naming conventions such as HUGO Nomenclature whenever possible.

Bioentities and States

In biological literature an entity typically maps to several different molecules in the cell. For example p53 protein has several different forms such as native, phosphorylated, and MDM2-bound forms, each representing different information contexts. In Patika Ontology p53 is a *bioentity,* a grouping of similar molecules, where each member of this group is a *simple state*. Note that a bioentity is an abstract concept, where a simple state is a reactant within the biochemical paradigm.

There are also *compound states*, a grouping of Patika objects, which exhibits a state-like behavior. Compound states are further classified into *complexes* and *abstract states*. Complexes represent non-covalent molecular complexes, whereas abstract states group elements based on homologies and level of detail.

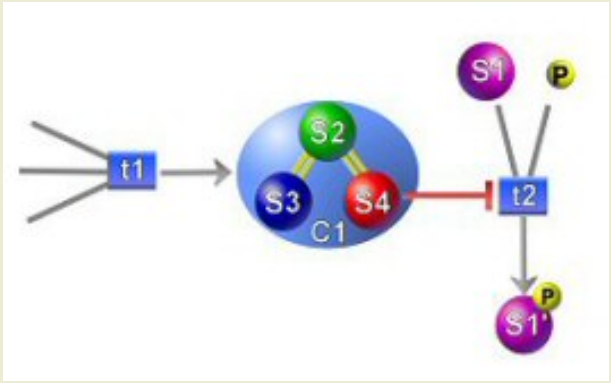


Figure 1-2 A portion of a pathway containing a molecular complex of three states

Transitions

Transitions represent events in the cell, typically reactions but also transportation of molecules, or environmental changes. A transition has a set of *products* and *substrates* and obeys a certain *stochiometry*. Also a transition might have *activators* and/or *inhibitors*. Similar to states, abstract transitions allow representing black box pathways and groups of similar reactions.

Interactions

Relations between bioentities, states and transitions are described using *interactions*. which can be directed or undirected. Interactions are divided into two based on their level of detail.

Mechanistic interactions define relations between states and transitions at the chemical level of detail. Bioentity interactions, on the other hand, are defined between two bioentities and cover cases where the mechanistic detail is unknown, such as yeast two hybrid data, or automatically parsed literature relations. They represent incomplete information, and always map to one or more mechanistic level interaction, although latter one might not be identified yet.

Compartments

A significant number of transitions transport molecules between cellular *compartments*. Transitions that a state can participate in are strictly related to its compartment; thus a change in the compartment means a change in the state's information context. We choose to incorporate the state's compartment in the model.

Abstractions

Network of molecular interactions derived from current biological data is incomplete and complicated. Complete network of cellular events is clearly beyond human perception. Different levels of *abstractions* are necessary to make effective analysis of cellular processes and dealing with complexity better.

1.4 Database Contents

PATIKA*web* provides access to data from a number of popular pathway databases, and interfaces with major databases and ontologies (Chapter 7).

1.5 Key Features

PATIKA*web* features a number of novelties as a pathway model constructor with a state-of-the-art visualization component:

- **Modeling & visualization at two different levels:** Biological phenomenon can be split into and represented at both the entity level and the mechanistic (state-transition) level (Chapter 2 on PATIKA Pathway Models and Views).
- **Topology manipulation operations:** Operations such as `delete` and `find neighbors` (Section 3.3 on Manipulating Graph Topology) and advanced querying facilities such upstream and downstream queries (Chapter 4 on Querying the PATIKA Database) help the user manipulate their model as desired.
- **Local persistence:** Current models can be persisted locally to be restored and used at a later time (See 2.3 on Save / Load Pathway Models and Views).
- **Import/export facilities:** Models from other databases or models constructed by users may be imported into PATIKA using BioPAX or PATIKA (.pmdl) notation. Similarly models in PATIKA may be exported into BioPAX or SBML format.

- **Standart graph editing facilities:** Current view can be manipulated through standard editing operations such as `zoom`, `scroll`, `fit-in-canvas`, `fit-cell-in-canvas`, and `overview window` as well as applying basic geometric changes to the drawings (Chapter 3 on View Manipulation).

- **Pathway object inspector and crosslinks to external databases:** Each pathway model object can be inspected through a properties dialog (Section 3.1 on Overview). This dialog possesses cross links to external database sources as well.

- **Advanced visualization techniques:** PATIKA pathway model views can be manipulated through advanced visualization techniques such as nesting and specialized static/incremental layout algorithms (Section 3.2 on Advanced Editing).

- **Microarray data analysis component:** Microarray data in tab-delimited popular formats may be loaded and mapped on top of constructed pathway models, visualized and analyzed using standard clustering methods.

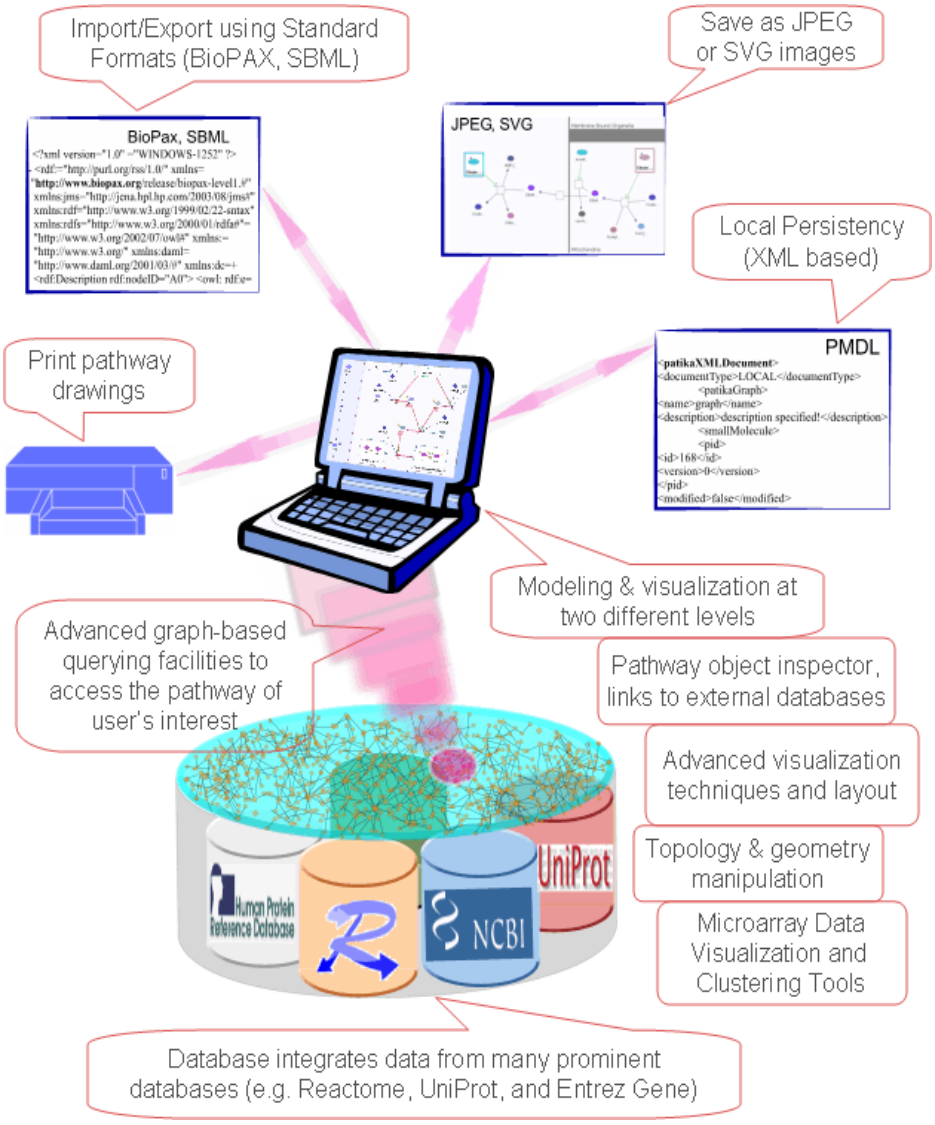


Figure 1-3 PATIKA*web* key features

1.6 Getting Started

Simply direct your browser to http://web.patika.org; you will be prompted with a license agreement for your first attempt. Once you accept the agreement, you will be faced with the page in Figure 1-4. The page simply presents certain statistics about the current contents of the PATIKA database along with a menu and an associated toolbar on top. You may load an existing pathway model from your local disk ("File | Load Model") or upload a sample one from the PATIKA server ("File | Sample Models"). Alternatively you may start from scratch and construct the pathway model of your interest by a query ("View | Query Dialog"; refer to Chapter 4 on Querying the PATIKA Database).

The drawing area for display pathway model views is called the *drawing canvas* or simply *canvas*.

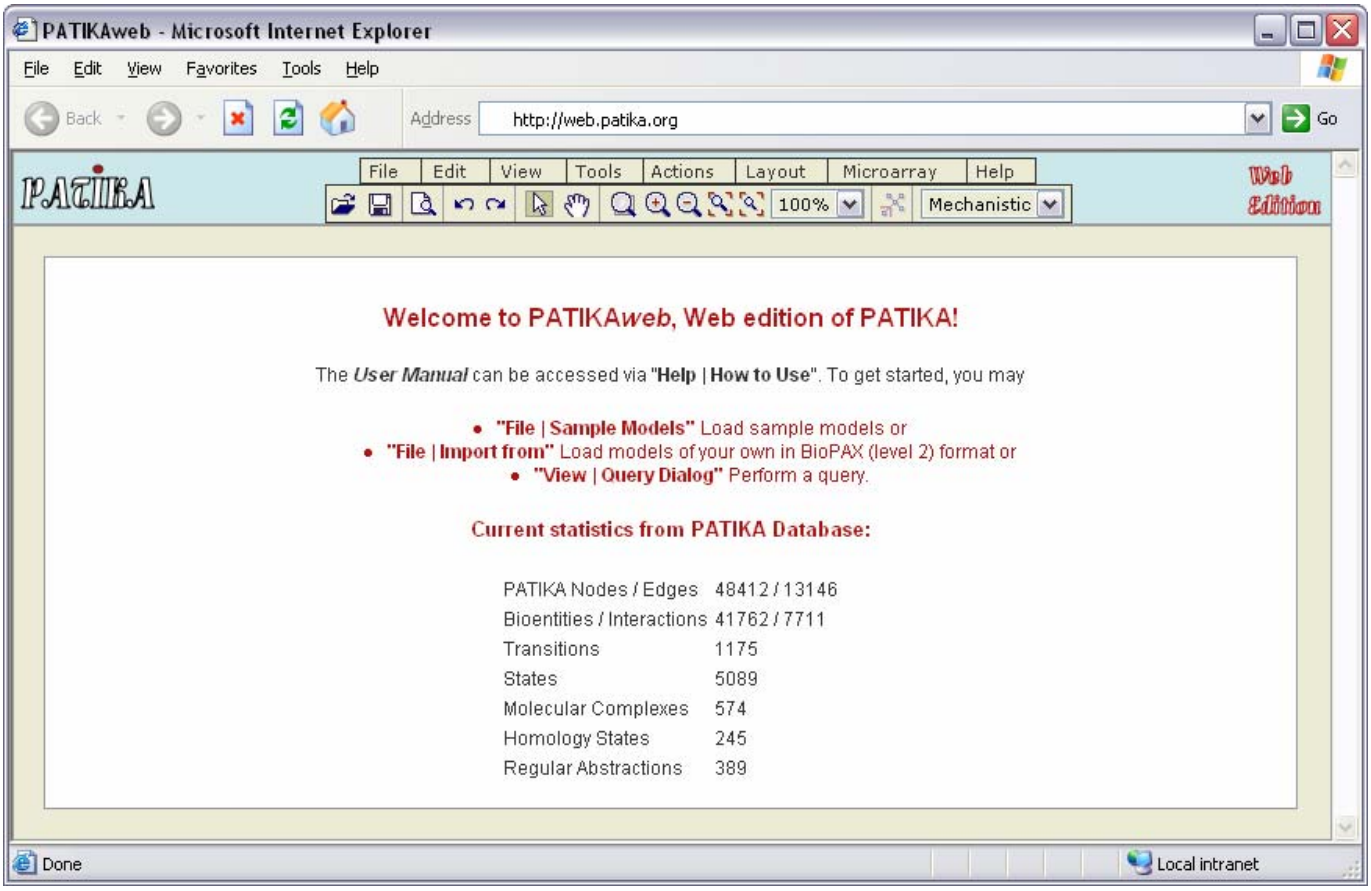


Figure 1-4 Initial PATIKA*web* page

! Please make sure that your browser does not block pop-up windows as some types of facilities such as querying and object inspection are provided through pop-up windows.

Chapter 2

2 PATIKA Pathway Models and Views

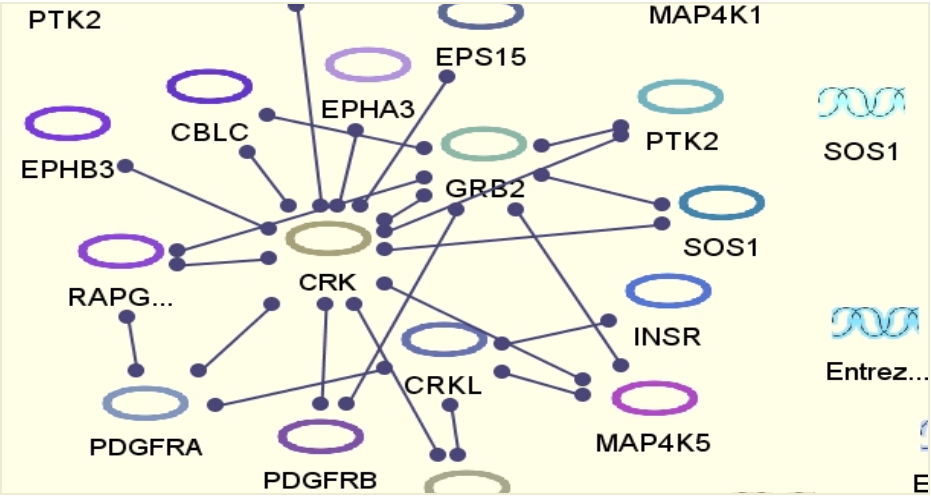
This chapter discusses the way PATIKA defines a *pathway model* based on the PATIKA ontology and how such models are presented to the users in two different *views*.

2.1 Modeling at Two Different Levels

Pathways are modeled at two different levels in PATIKA: biological entity level and mechanistic (state-transition) level; and for each level we provide a separate view.

2.2 Model Graph vs. View Graphs

A PATIKA pathway model (PPM) is made up of a *model graph* or simply a *subject* (PS) and two *view graphs* or simply *views*. Each view corresponds to the associated model's bioentity (PBV) and mechanistic level views (PMV). Figure 2-1 illustrates this with an example.



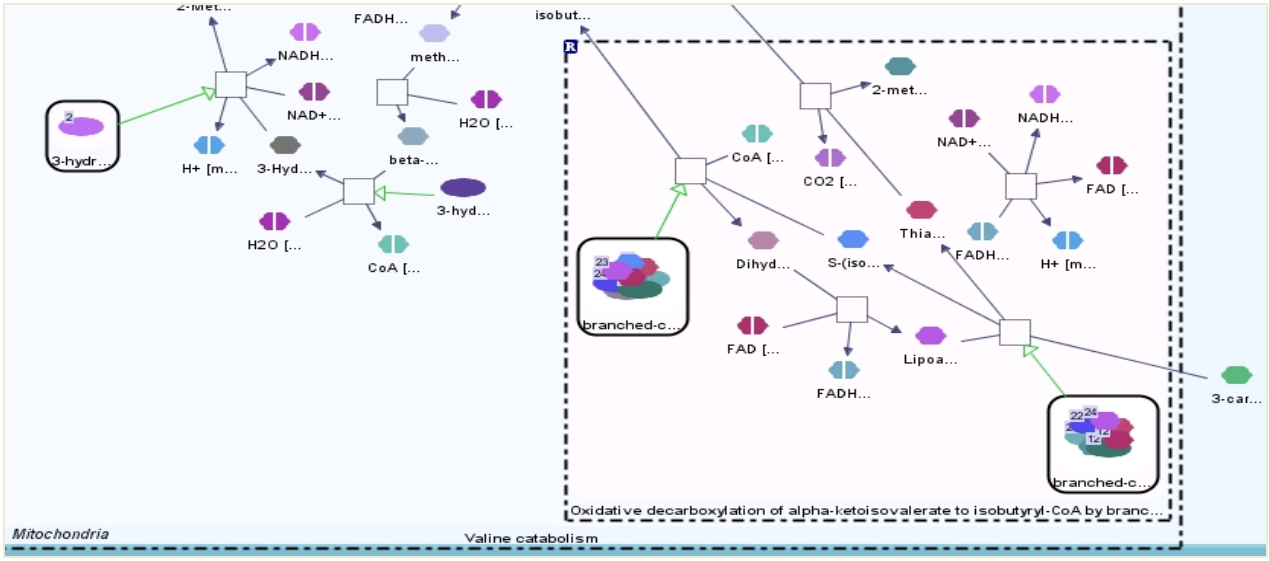


Figure 2-1 Pathway models with partial bioentity (top) and mechanistic views (bottom)

When the currently selected view is empty, the Welcome page is displayed.

2.3 Save / Load Pathway Models and Views

One can persist the currently built model locally on disk using "File | Save Model". This will save the current pathway model along with its subject and associated views. Note that default extension for PATIKA pathway models is ".pmdl".

You can load a previously saved model using "File | Load Model" (Figure 2-2). This operation will discard the current pathway model replacing it with the one from disk.

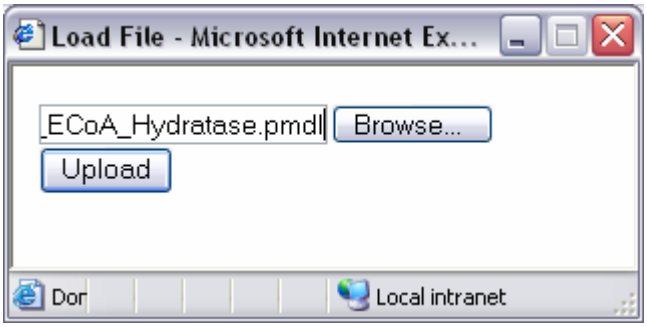


Figure 2-2 Load Model Dialog

Import from BioPAX

You may upload pathway models in BioPAX level 2 format as well, using "File | Import from". This will convert the BioPAX model into PATIKA model, which you may manipulate and analyze as you like.

Export to BioPAX and SBML

Optionally you may like to export a PATIKA pathway model into another format such as the standard exchange formats like BioPAX (level 1) and SBML. You may do so by using "File | Export to".

Saving a View as an Image

One can also save a pathway model view as an image using "File | Save as Image". JPG and SVG formats are supported by PATIKA*web*. Note that in order to fully capture a pathway model you might want to produce an image for each of the two associated views.

In addition, you may print a view by first clicking "File | Printable View" and then using your browser's print functionality.

Chapter 3

3 View Manipulation

This chapter describes how to manipulate PATIKA pathway model views. Remember that each pathway model has exactly two views: a bioentity view (PBV) showing interactions such as protein-protein interactions and a mechanistic view (PMV) showing transitions such as membrane transportation of a molecule.

3.1 Basic Graph Editing

PATIKA*web* provides state-of-the-art visualization tools for manipulating pathway model views. *Undo* and *redo* support is available for all editing operations.

One may change the current editing tool from the toolbar located right under the main menu (Figure 3-1).

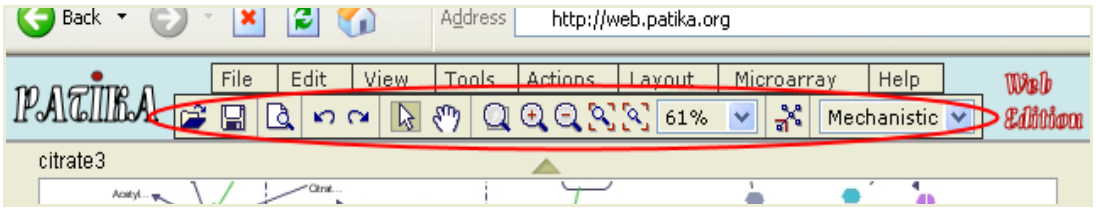


Figure 3-1 PATIKA*web* toolbar

Selection

The default tool in PATIKA*web* is the *Select Tool*. Selection starts with a left-click on the canvas. If the mouse is on a PATIKA pathway model object at the time of the click, then the hit object is selected; otherwise a rectangle is started to be used for a marquee selection. If you left-click on a PATIKA node and drag it, then the hit node is moved to the location at which the mouse is released, and the new selection list is set to the dragged node. Should the clicked node be one of the previous multiple-selection, then a group move is performed.

! However, notice that for mechanistic views, should the release location of a node be outside the compartment of the node, it is relocated at a position closest to the desired respecting the compartment constraint.

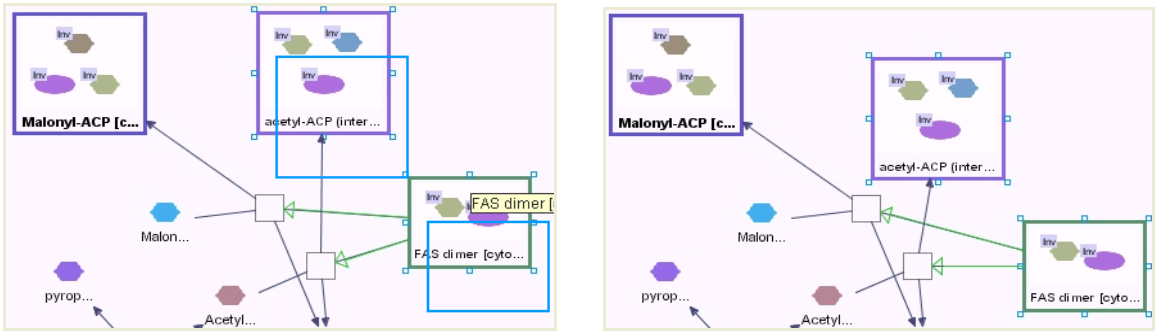


Figure 3-2 During (left) and after (right) a drag of two complex states

The selection tool may also be used to resize compartments by simply left-click on one of its separators and dragging to the desired location. However, the compartment size will not be allowed to be reduced if the new separator location results in any PATIKA node violating its compartment constraint.

Panning

A convenient way to scroll a drawing is using the panning tool. When you left-click on location A on the canvas and drag it to a new location B, then the objects in the drawing are shifted by the distance between A and B; in other words, the object at location A ends up in location B after the operation (Figure 3-3).

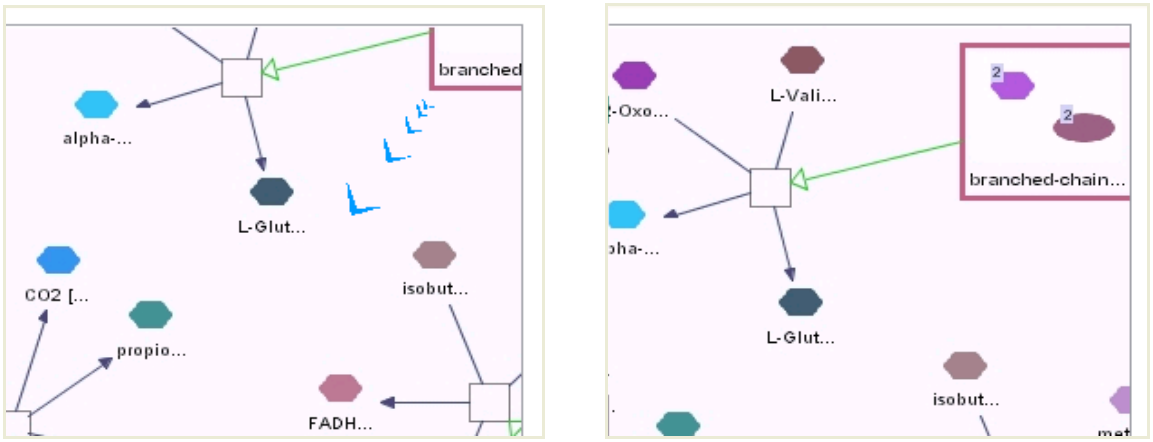
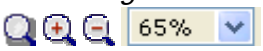


Figure 3-3 During (left) and after (right) panning to shift the current view towards lower left as indicated with arrows drawn by the panning tool

Zooming



Tools for marquee zoom, zoom in, zoom out, zoom to a predetermined level may be respectively used to change the zoom level.

Scrolling

Scroll left, up, down, and right tools may be respectively used to scroll the current drawing by 20 percent of the associated canvas dimension (height or width).

Fit-in-canvas and Fit-cell-in-canvas

For mechanistic views these two tools may be used to set the zoom level such that the bounding rectangle of the drawing or the bounds of the cell tightly fit the canvas, respectively (Figure 3-4).

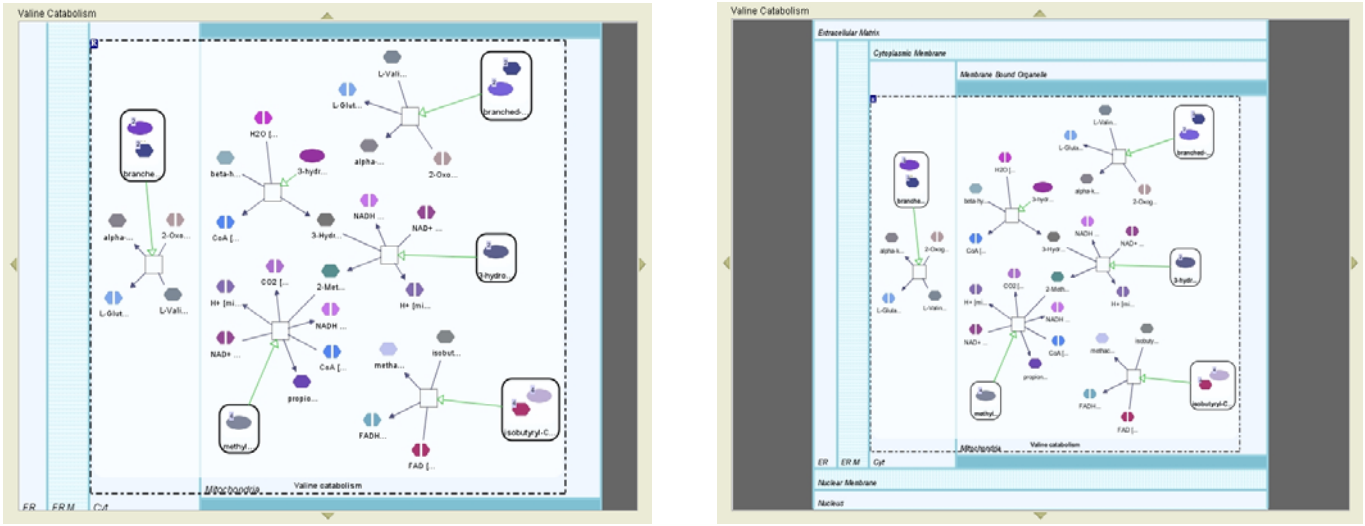
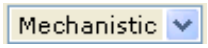


Figure 3-4 Examples of fit-in-canvas (left) and fit-cell-in-canvas (right)

Layout

Pathway drawings can be automatically arranged for easier visualization through the use of our specialized layout algorithms (see Section 3.2 for details).

View Type



The current view may be changed using the combo box in the toolbar between Mechanistic and Bioentity views.

Overview Window

The overview window (Figure 3-5) located on the upper right of the window may be used to have an overview of the entire drawing and quickly navigate to the area the user wants to focus on. The blue marquee rectangle in the overview window indicates the part of the drawing currently viewable in the canvas. This window may also be used for quick navigation over the drawing by simply dragging it (i.e. panning) or resizing it (i.e. zooming).

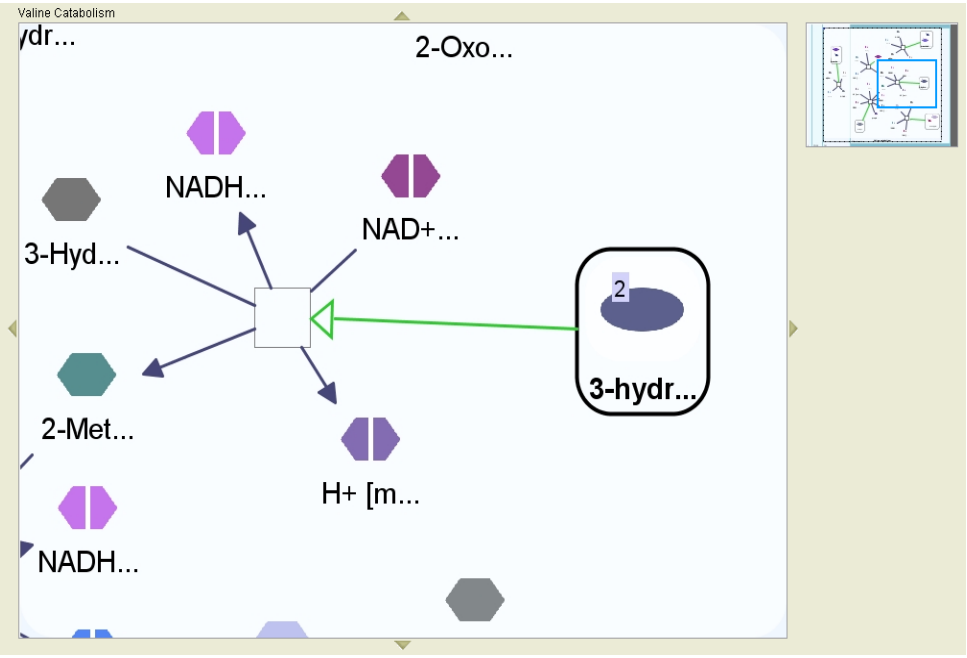


Figure 3-5 Overview window is located at the upper-right of your window.

Object Inspectors

You may right-click on a PATIKA object and select "Properties" to pop-up its inspector window (Figure 3-6). This window lists the properties of the associated object in a table, including any references (hyperlinks) to external databases. Notice that you may view the properties of as many objects as you like as each inspector opens up in a new window.

The choices for "Data Source" are as follows:

- Experimental
- Inferred
- Imported
- Other

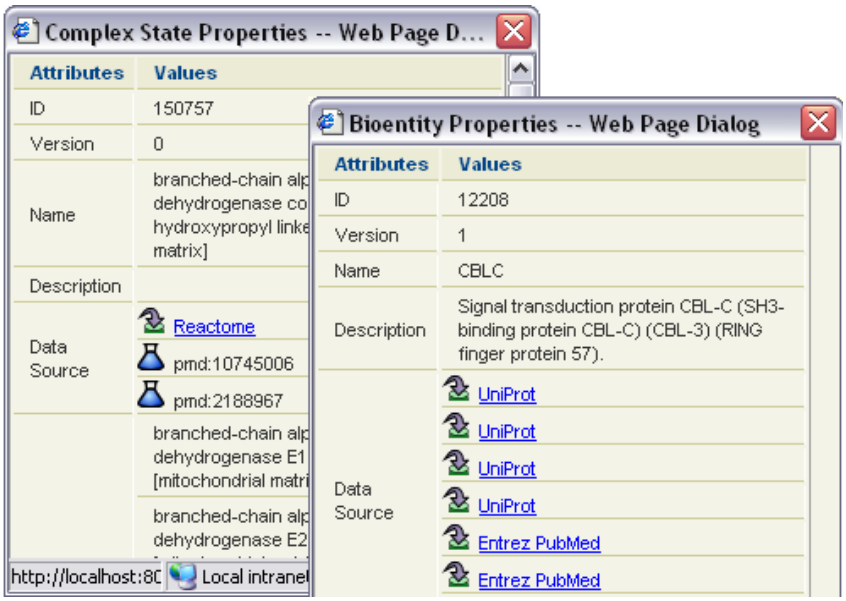


Figure 3-6 Inspector windows of a bioentity and a simple state; in this example both object's data source is the Reactome database. The user may follow the provided link to the Reactome Web page for details of this object.

3.2 Advanced Editing

PATIKA*web* provides various ways to edit the drawing at hand.

Nesting through Expand/Collapse

Various types of compound nodes (abstractions and complexes) explained earlier can be expanded or collapsed to manage complexity of a pathway drawing (Figure 3-7).

! Multi-compartment spanning compound nodes are not allowed to collapse.

Not every abstraction can be visualized through nesting; when not possible such abstractions are color coded through *holos* (see the sub-model in Figure 3-7 in which states and transitions in the lower left of the drawing belonging to an abstraction are colored green).

Also note that when a compound node is collapsed, any inter-graph edge going into this compound node will be replaced by a representative edge, called *meta-edge*, to draw attention into the no-longer-visible interaction. Meta edges are created and destroyed as needed and are not taken into account by many operations including queries!

! Expand operation is not allowed when there is not enough room for the expanded node. This might occur due to compartment resize or movement of the collapsed node near a compartment's boundaries.

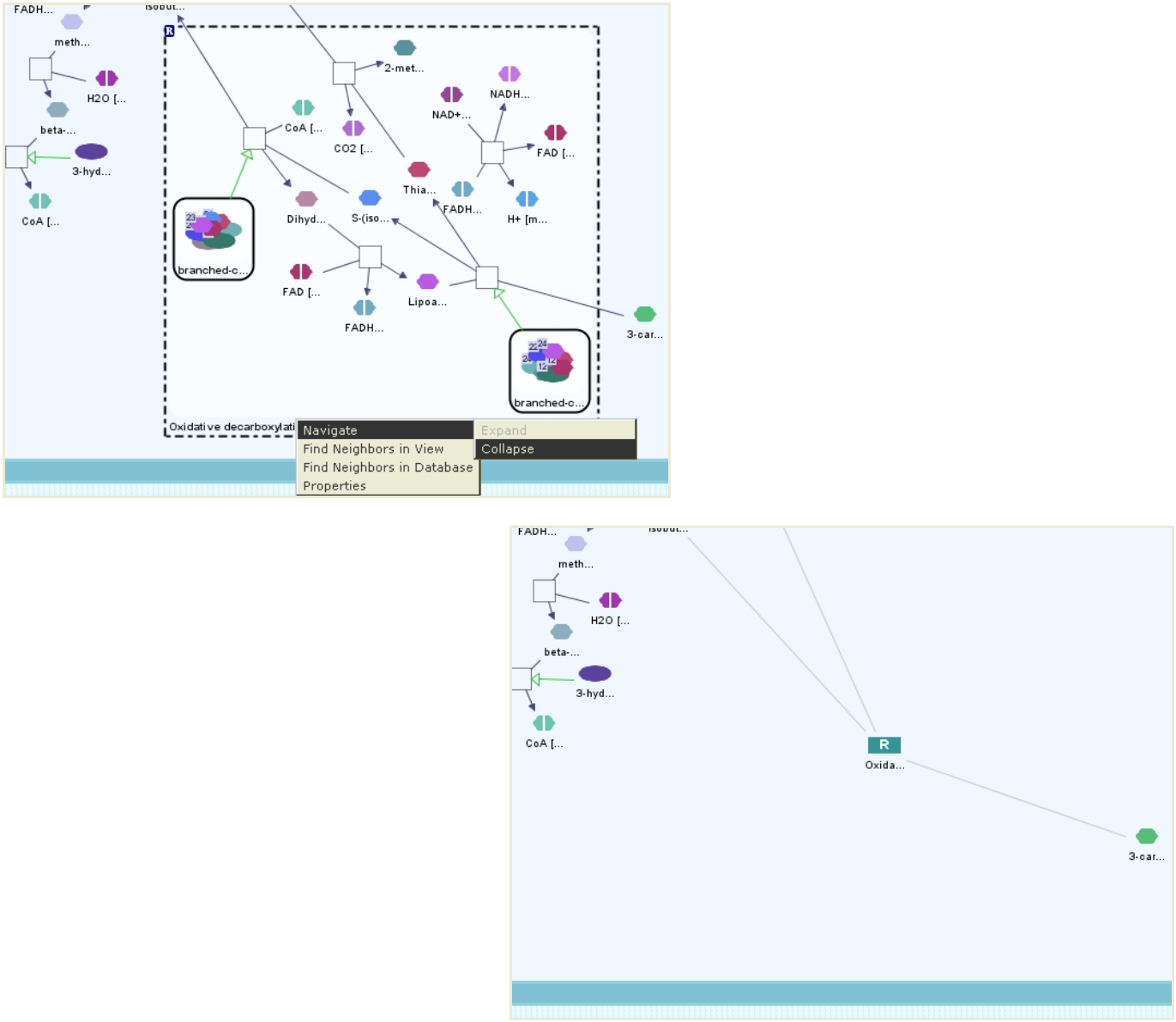


Figure 3-7 Before and after a regular abstraction (Oxidative decarboxylation of …) is collapsed; notice the meta-edges formed due to inter-graph edges being no longer visible.

Layout

Regular and incremental layout algorithms developed in-house [2] may be performed to rearrange the pathway drawings. Incremental layout respects the current geometry of the drawing while rearranging it by keeping the objects' relative positions as much as possible.

Regular layout is applied on operations (e.g. a query result displayed in a new window) that form a new view from scratch whereas incremental layout is performed on operations (e.g. a neighborhood search that merges the new neighbors into the drawing) that modify the current drawing.

Layout properties may be adjusted through the Layout Properties Dialog under Layout in the main menu (Figure 3-8).

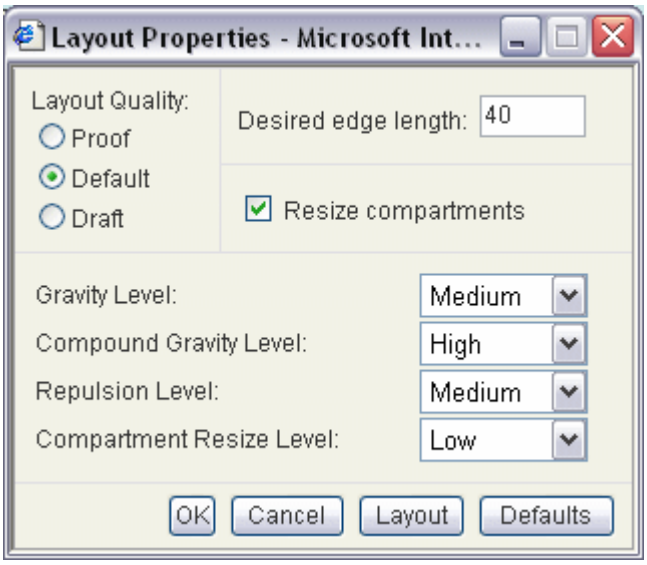


Figure 3-8 Layout Properties Dialog through which layout options can be set

LAYOUT QUALITY

This parameter adjusts the convergence criteria; the higher the quality is set, the more the number of iterations that are performed, resulting in a possibly nicer drawing at the cost of more execution time.

DESIRED EDGE LENGTH

This parameter sets the desired edge length of layout. The higher this parameter is, the longer the edges will be when layout is performed.

RESIZE COMPARTMENTS & COMPARTMENT RESIZE LEVEL

This parameter applies to layout of mechanistic pathway views only and specifies whether the compartments are allowed to be resized by the layout operation. If it is set to true, layout tries to adjust each compartment's size optimally to fit its contents. However, the user is allowed to determine the amount of resize allowed using the compartment resize level.

GRAVITY LEVEL

This parameter specifies the level by which disconnected pathway objects should be pulled together towards a "virtual center of gravity". This center is taken to be the middle of the bounding rectangle for compartments and compound nodes (i.e. abstractions and complex molecules).

COMPOUND GRAVITY LEVEL

This parameter helps user adjust the level by which disconnected pathway objects should be pulled together towards the center of compound nodes (does not affect the biological objects outside compound nodes).

REPULSION LEVEL

This parameter specifies the level by which pathway objects are allowed to come close. The lower this parameter is, the closer objects are allowed to come.

3.3 Manipulating Graph Topology

The topology of a pathway view may be modified through several operations.

Delete Selected

The selected graph object(s) may be deleted from a model (both from the model graph and the associated view graph) by simply choosing "Delete Selected from Model" from the graph popup menu or from the "Edit" menu. Notice that some unselected objects might get deleted as a result of this operation as well, to keep the model consistent with the database. For instance, when a member of a complex molecule is deleted, so is the complex molecule. Or deletion of a product edge will result in deletion of the associated transition as well.

Crop Selection

The user might like to create a new view (and model) containing only the currently selected objects using "Edit | Crop to Selected".

Find Neighbors

You may discover the immediate neighbors of a PATIKA node by choosing "Find Neighbors in Database" from the node's popup menu. In case the PATIKA node is a bioentity, bioentities that interact with this bioentity are determined and brought into the bioentity view if not already there. In case the node is a mechanistic one (i.e. a state), all states that are one transition away from this one are picked. In either case, the newly formed drawing is incrementally laid out and the neighbors are highlighted.

In case you are only interested in neighbors in the current model, perform "Find Neighbors in View".

Merging Query Results

As we will discuss later on, the results of a newly performed query may be used to form a new model or may be merged into the current view. Merging is performed at both the bioentity and mechanistic level.

3.4 Highlighting

Highlighting is performed after certain operations (e.g. a query) to stress certain parts of the model views. However, it may be removed by choosing "Remove Highlights" from the graph popup menu or from the "Actions" menu. You may also highlight a selected set of biological objects using "Actions | Highlight Selected".

Chapter

4

4 Querying the PATIKA Database

PATIKA*web* provides an advanced, graph-based querying facility for retrieving the data of user's interest from the PATIKA database. The querying component may also be used to query the current local model such as loaded or imported models.

4.1 Overview

Querying component of PATIKA supports both SQL-like queries and an array of graph-theoretic queries for finding shortest paths, feedback loops, positive/negative paths, common targets and regulators, or "interesting subgraphs" based on user's genes of interest. Once retrieved from the database, the query results may be merged to the user's current view and highlighted to provide an incremental user-friendly retrieval and analysis interface. Alternatively query results may form a new pathway model from scratch. Constructed models can be saved in XML, exported to standard formats such as BioPAX and SBML or converted to static images.

The query interface of PATIKA*web* has been implemented as an applet (Figure 4-1 and Figure 4-2) and can be invoked by "View | Query Dialog".

Figure 4-1 The user is asked to agree to run the PATIKA Query Applet.

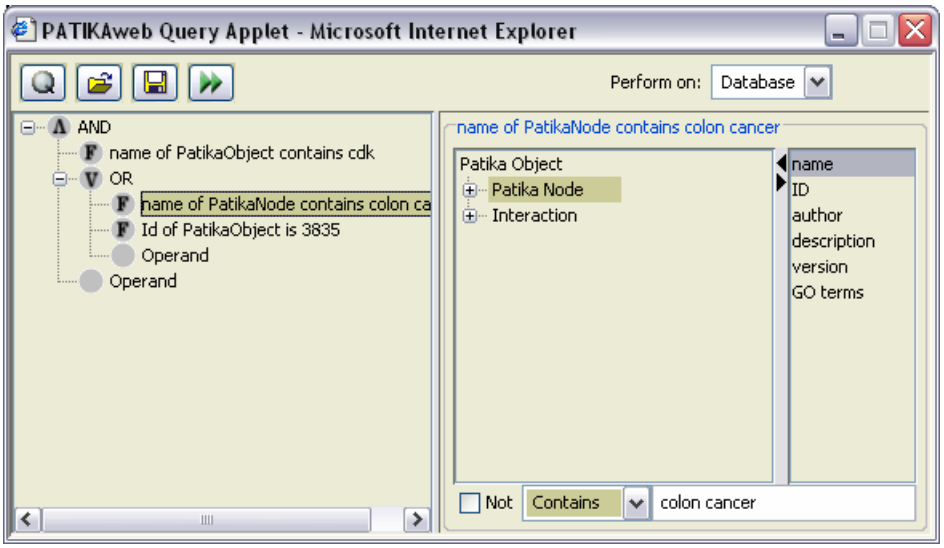


Figure 4-2 The Query Dialog consists of a toolbar (top) and panels for query forest (left) and parameters (right)

The initial query node of the query forest must be constructed by using the "New Query" tool in the toolbar. Its sub-queries or parameters may be constructed by the pop-up menu of a query node (Figure 4-3).

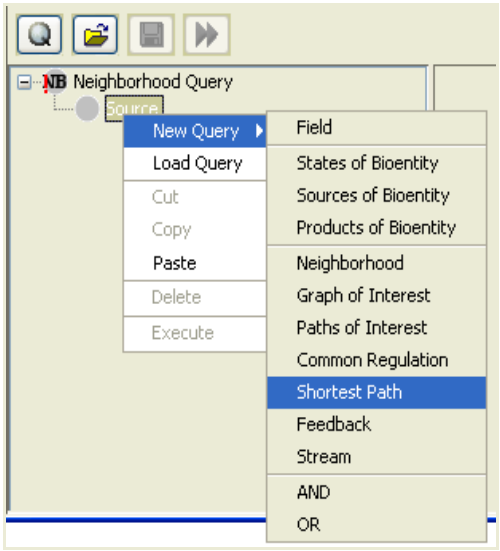


Figure 4-3 The source of a Neighborhood query is set through the pop-up menu of its "source" query node.

The user may choose whether the query is to run on the database or the view (i.e. local/current model) (Figure 4-4).

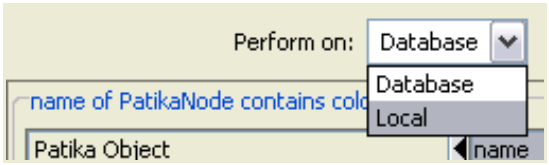


Figure 4-4 The user can specify whether the query is to execute on the database or the view.

When the user clicks on the "Execute Query" button, the query tree rooted at the currently selected query node of the query forest is executed (Figure 4-5).

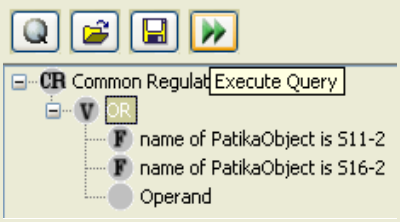


Figure 4-5 The query rooted at the OR query is executed when the user presses associated button.

Query Results

After the execution of a query initiated from the Query Dialog finishes, the returning result (i.e. pathway model) is summarized by the Query Result Dialog (Figure 4-6).

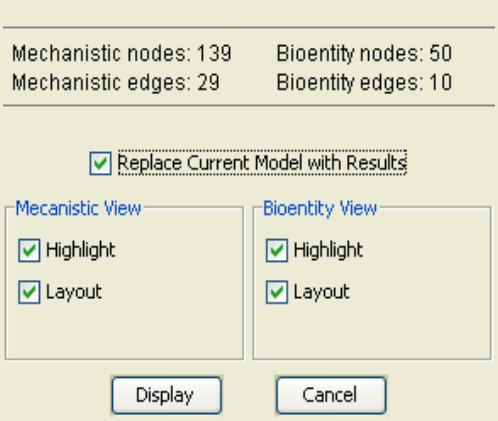


Figure 4-6 Sample Query Result Dialog

A number of statistics about the result is displayed in this dialog:

- Mechanistic nodes/edges (number of nodes/edges at the mechanistic level) ,
- Bioentity nodes/edges (number of bioentities/interactions at the bioentity level).

If the user does checks the "Replace Current Model with Results" option, then the previous pathway model and its views are discarded and the pathway model of the query results is displayed. If this option is unchecked, resulting model is merged into the existing one, possibly modifying both views.

The user may opt to highlight the new objects (pathway objects that were not part of the previous model but belongs to the model of the query result) in the views displayed. Normally a separate color is used for different roles in the resulting model. For instance in a neighborhood query, the source nodes are highlighted with a distinct color, whereas neighbors are highlighted with a different common color. The colors are pulled out of a fix color set sequentially. The legend for the highlight colors of the last query may be reached through "View | Query Highlight Legend". A sample highlight legend is shown in Figure 4-7.

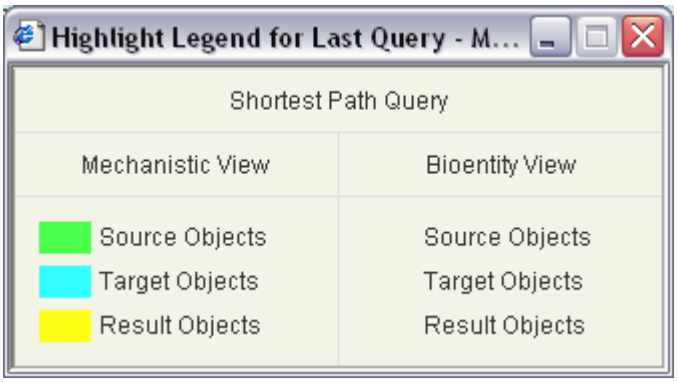


Figure 4-7 Sample Query Highlight Legend Dialog, where the last query is a shortest path query, and source, target and result (shortest paths) objects are highlighted with distinct colors (green, cyan and yellow, respectively).

If the user checks "Layout", then the associated view is also laid out.

Persisting Queries

You may persist the constructed queries locally on disk for later use clicking on the save tool in the Query Dialog toolbar. The default extension for XML based PATIKA queries is ".pql".

! Note that it's the query tree rooted at the selected query object that is saved on disk, not the entire query forest or the entire associated query tree.

To reload a previously saved query, click on the load tool in the Query Dialog toolbar, and point to the proper ".pql" file. This will not destroy the current query forest but add a new query tree at the very end.

4.2 Basic Queries

Field Query

The simplest query type that can be performed in PATIKA*web* through the query applet is the field query. Following fields of objects can be queried: *name*, *PATIKA ID*, *Author ID*, *description*, *version* and *GO terms*.

In most queries including the field query, you can specify and constrain the search to a PATIKA object type using the PATIKA object tree (Figure 4-8). Please refer to Chapter 6 on PATIKA Ontology for a better understanding of the PATIKA object tree.

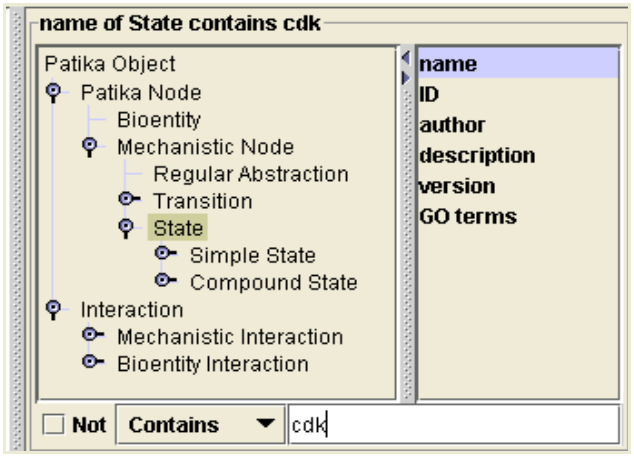


Figure 4-8 You may constrain your query to a PATIKA object of a specific type.

States of a Bioentity

This query is used to find all states of a specified set of bioentities in the current model or the database.

For instance, all available states of protein "tp53" can be obtained through this query as shown in Figure 4-9.

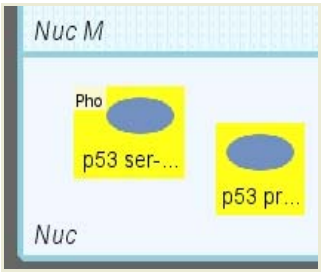


Figure 4-9 All states of protein "tp53" (yellow) as obtained by a "States of a Bioentity" query

Sources of a Bioentity

This query may be used to find all source bioentities of a specified set of bioentities in the current model or the database.

As shown in Figure 4-10, DNA "tp53" is found as the only source bioentity of protein "tp53" in PATIKA database.

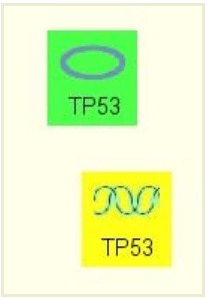


Figure 4-10 Sources (yellow) of protein "tp53" (green) as obtained by a "Sources of a Bioentity" query

Products of a Bioentity

This query may be used to find all product bioentities of a specified set of bioentities in the current model or the database.

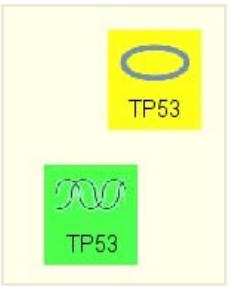


Figure 4-11 Products (yellow) of DNA "tp53" (green) as obtained by a "Products of a Bioentity" query

4.3 Logical Queries

Queries can be combined using logical operators. In addition, most queries are defined recursively taking the result of another query as input (Figure 4-12).

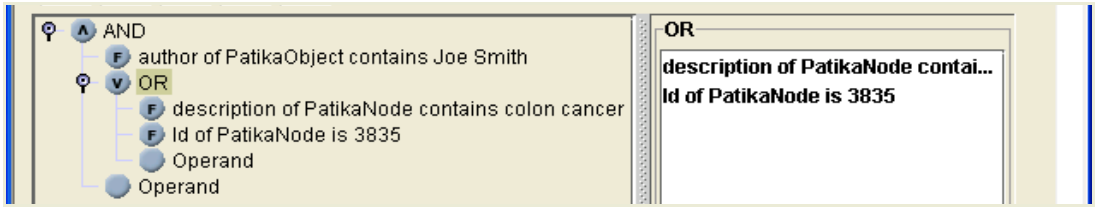


Figure 4-12 Queries can be combined through logical operators: Search for all PATIKA nodes whose description contains "colon cancer" or ID equals 3835 authored by Joe Smith.

4.4 Advanced Queries

More advanced graph algorithms that involve traversal starting with a set of source and/or target objects are explained in this section.

Advanced Query Options

Below are some common traversal options valid for all advanced queries. These options can be accessed and set through the button "Options" on the right hand side of each query panel (Figure 4-13).

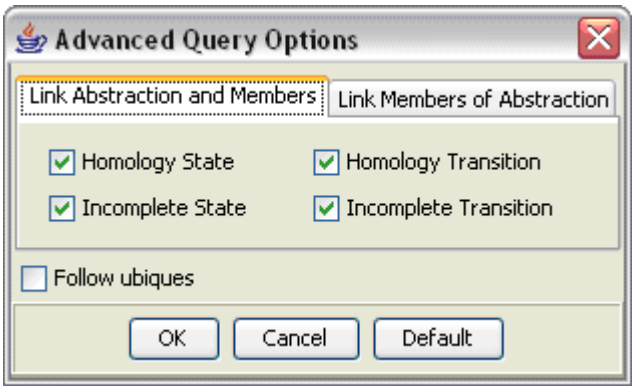


Figure 4-13 Advanced Query Options Dialog is used to set traversal options for ubiques and abstractions.

TRAVERSAL OVER UBIQUES

Traversal over ubiquitous molecules (see Section 6.3) may be avoided by un-checking "Follow Ubiques" flag in Options Dialog. Since such molecules are involved in potentially hundreds if not thousands of reactions at mechanistic level, one might prefer not to link two reactions whose only common actors are these kinds of molecules.

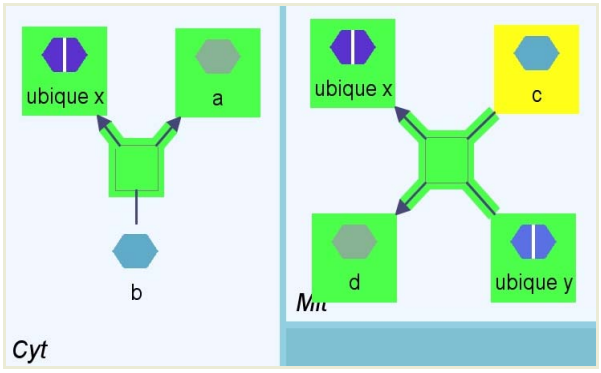


Figure 4-14 Whether or not "a" is in the 4-neighborhood (green) of "c" (yellow) depends on whether traversal over ubiques ("ubique x" in this case) is allowed. Obviously, in this case it is allowed.

TRAVERSAL OVER ABSTRACTIONS

These options decide how the traversal should continue, when it reaches an abstraction or a member of an abstraction. For instance, reaching a member of a complex molecule should often be interpreted as reaching **all** members of this complex; thus the traversal should be able to continue from another member.

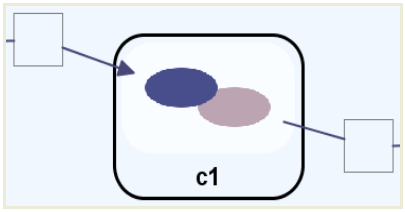


Figure 4-15 Traversal reaching complex "c1" from the transition on the left will continue to transition on the right if and only if "Link Members of Complex" option is true.

Neighborhood Query

Neighborhood query (NB) results in a set of molecules that are at most a specified distance from the source set (Figure 4-16).

- ***limit*** specifies the maximum distance of a molecule from a source molecule to be considered as part of the result of this query.

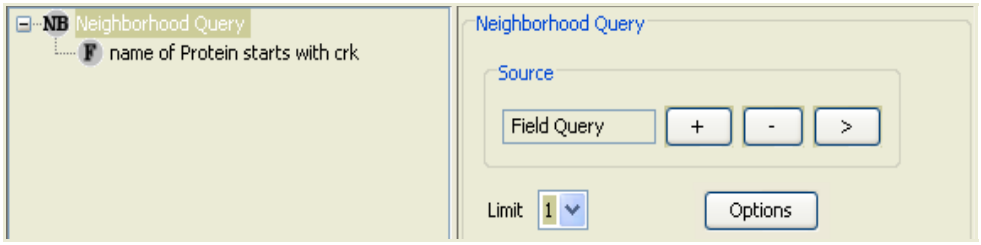


Figure 4-16 Sample Neighborhood Query Dialog for 1-neighborhood of protein bioentities whose names start with "crk"

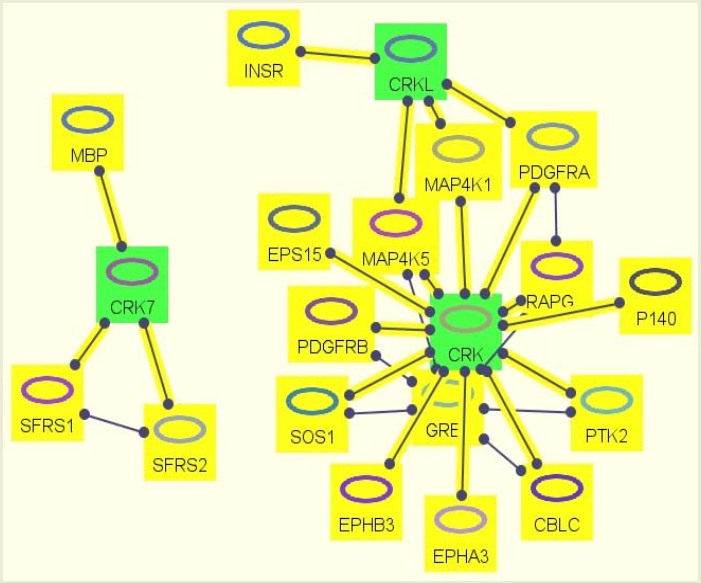


Figure 4-17 1-neighborhood of protein bioentities whose names start with "crk" (result of the query in Figure 4-16)

There are shortcuts for the neighborhood query in the popup menu of nodes in the currently displayed view. When you select "Find Neighbors in View" or "Find Neighbors in Database" item in the popup, a neighborhood query, respectively on the current view or the database, is performed where the source is the field query with the ID of the hit object, and limit is 1.

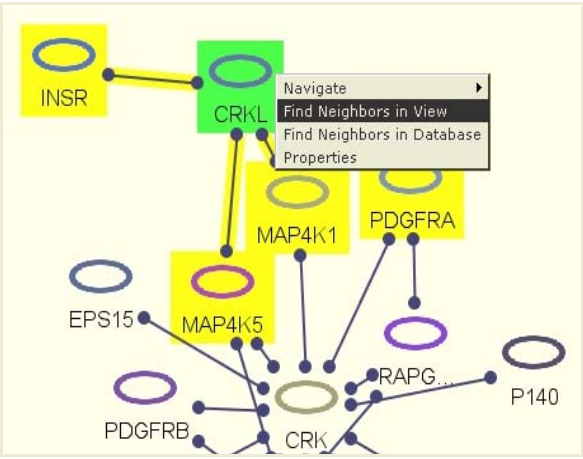


Figure 4-18 Immediate neighbors (yellow) of protein "CRKL" (green) may be queried using its popup menu

Graph/Paths of Interest

Graph-of-Interest (GoI) query aims at completing the "missing links" (and molecules on these links) among a set of molecules of interest that is no longer than a specified limit (Figure 4-19).

- ***limit*** specifies the maximum length of a missing link from a molecule in the specified source set to a molecule in the same set to be considered as part of the result of this query.

Paths-of-Interest (PoI) query, on the other hand, does the same thing *from* a specified set of source molecules *to* a specified set of target molecules (Figure 4-21).

- *limit* specifies the maximum length of a missing link between a molecule in the source set and a molecule in target set to be considered as part of the result of this query.

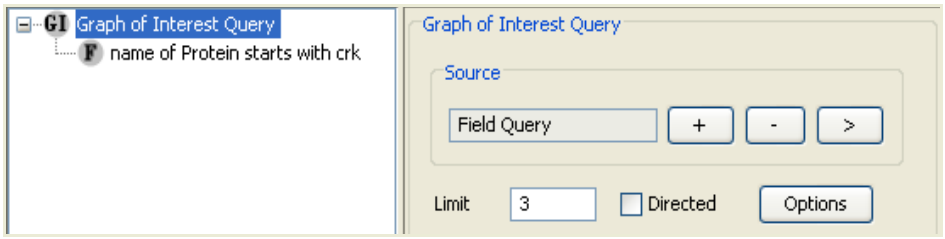


Figure 4-19 Sample GoI Query Dialog

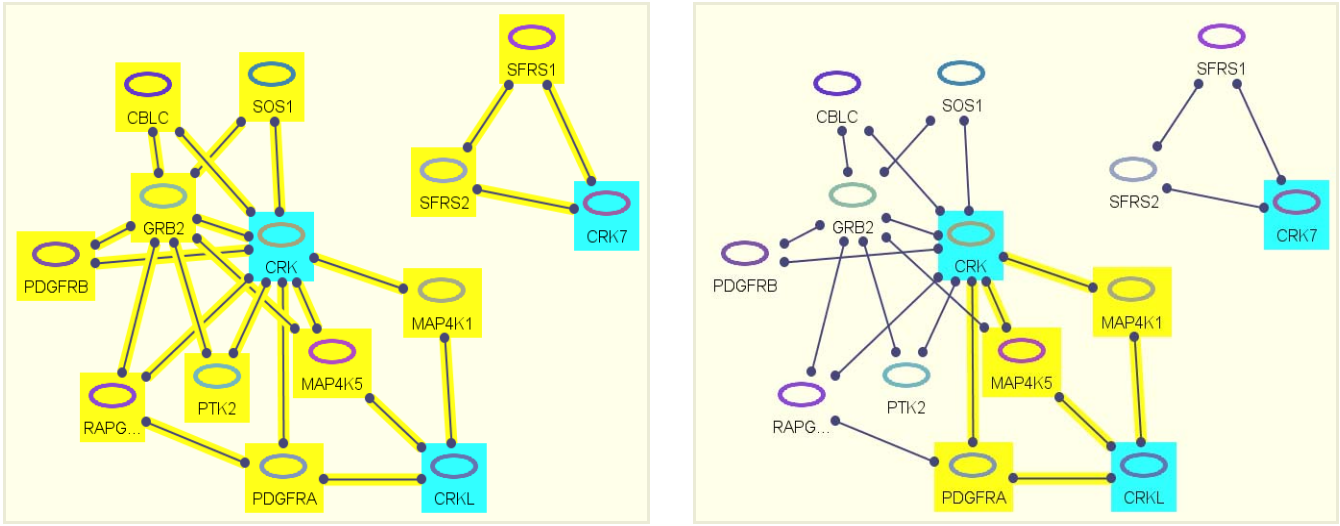


Figure 4-20 Result of the query in Figure 4-19: GoI (yellow) of protein bioentities whose names start with "crk" (cyan) where *limit* is 3 (left) and where *limit* is 2 (right). Notice that each protein on the GoI (yellow) is on at least one path between two source nodes (cyan) whose length is at most *limit*.

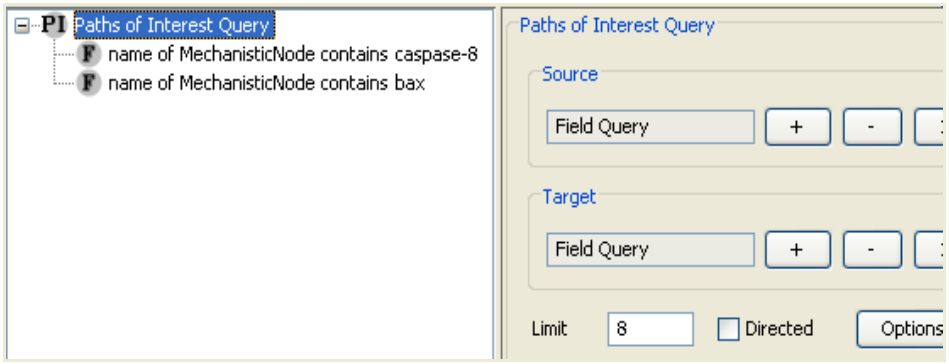


Figure 4-21 Sample PoI Query Dialog

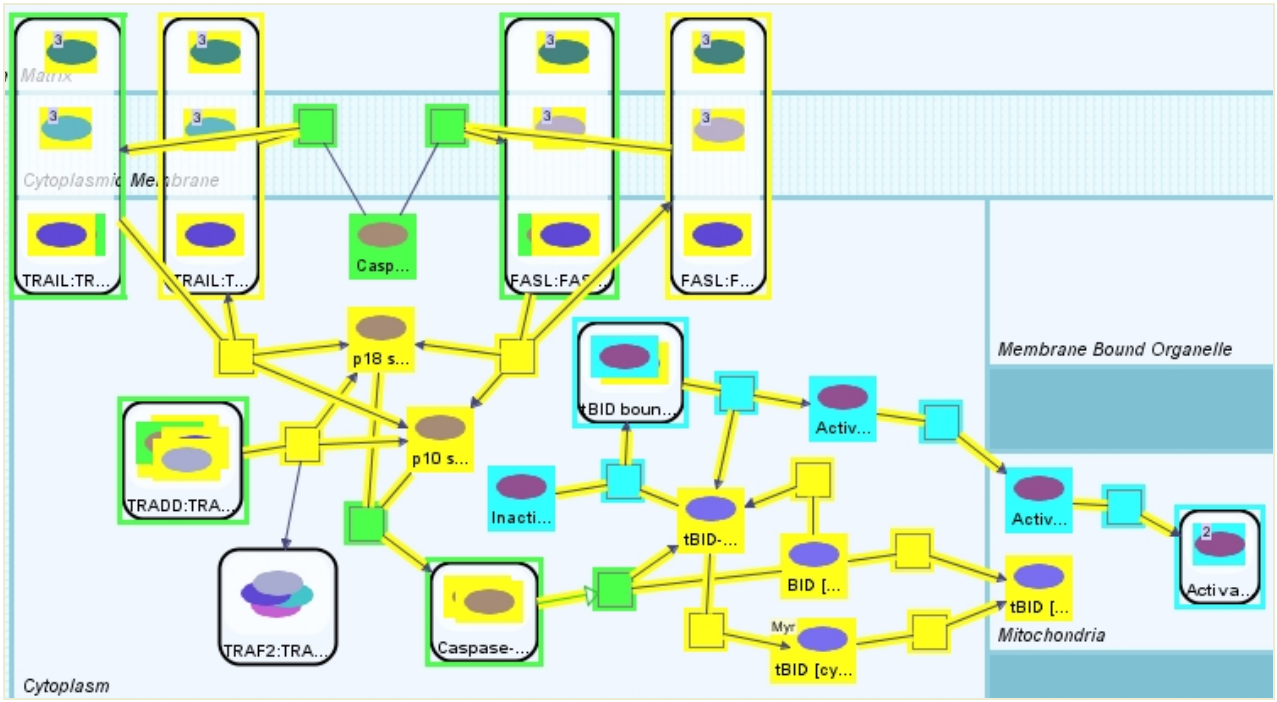


Figure 4-22 Result of the query in Figure 4-21: PoI (yellow) between mechanistic nodes whose names contain "caspase-8" (green) and those whose names contain "bax" (cyan)

Common Target / Regulator

If node A is the starting node of a directed path that ends up in node B, then node B is said to be a *target* of node A; similarly, node A is said to be a *regulator* of node B. In this context, common target of a source molecule set S is the set of molecules that are targets of all molecules in S. Similarly, the common regulator of a target molecule set S is the set of molecules that are regulators of all molecules in S.

This query results in a set of molecules that are common targets (regulators) of all the molecules in the source (target) set (Figure 4-23).

- ***limit*** specifies the maximum distance of a target (regulator) molecule from a source (target) molecule to be considered as part of the result of this query.
- ***direction*** specifies whether we are interested in finding targets or regulators.
- ***include regulation paths*** specifies whether the actual paths from the given molecule set to the targets (or from the regulators to the given molecules) are to be included as part of the result.

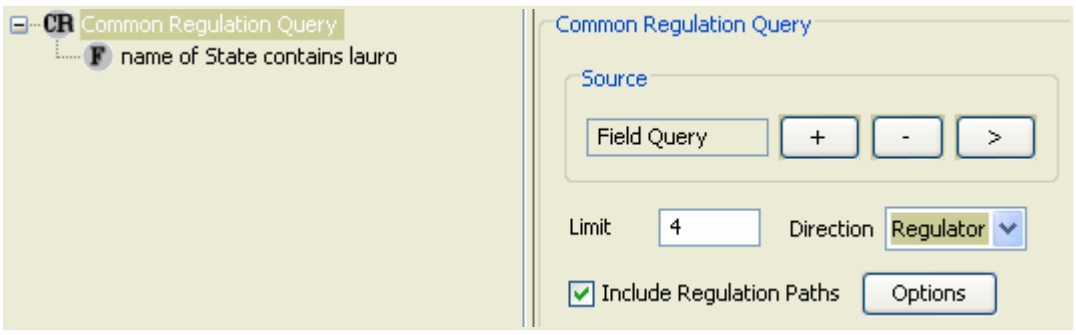


Figure 4-23 Sample Common Regulator Query Dialog

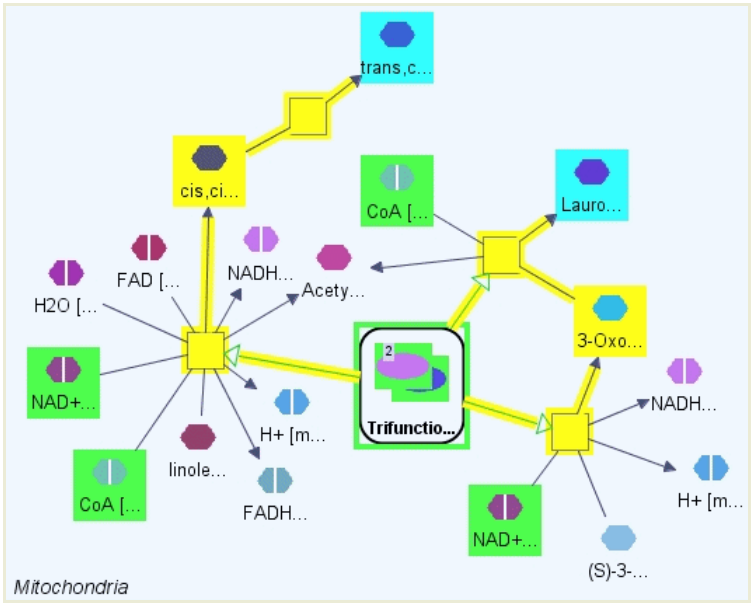


Figure 4-24 Result of the sample common regulator query in Figure 4-23, where we find common regulators (green) of the simple states whose names contain "lauro" (cyan); paths leading from common regulators to the targets are shown in yellow.

Shortest Paths

This query may be used to find the shortest paths (that is, pathways of shortest length) between two sets of pathway objects (i.e., source and target sets) (Figure 4-25). The user may specify source and target object sets by other PATIKA queries (e.g., field queries).

- ***limit*** specifies the maximum length of a path to be considered as a result of this query.
- ***directed*** is used to specify whether the edge (interaction) direction (e.g., product edge of a mechanistic view, going from the associated transition to the product state) is to be taken into account or not.
- ***further distance*** allows selection of paths that are longer than the shortest path(s) up to a certain length.

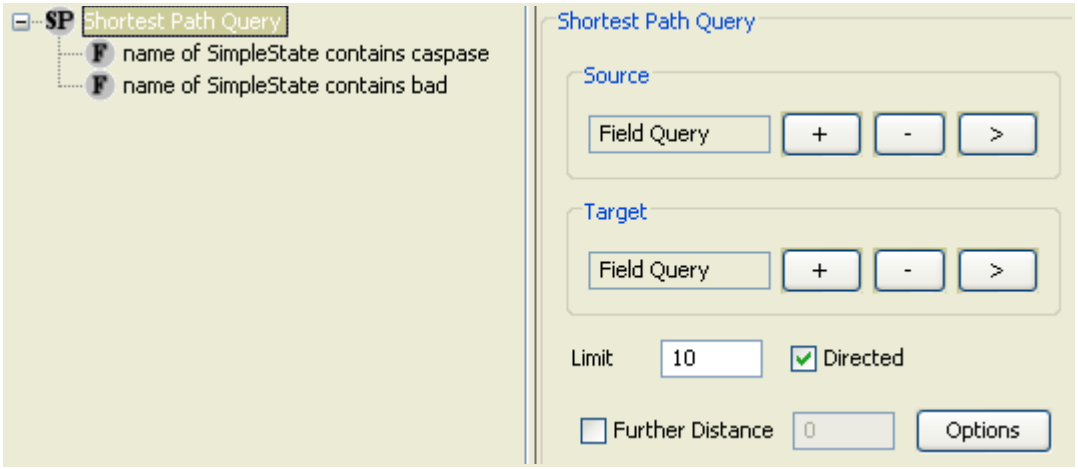


Figure 4-25 Sample Shortest Paths Query Dialog

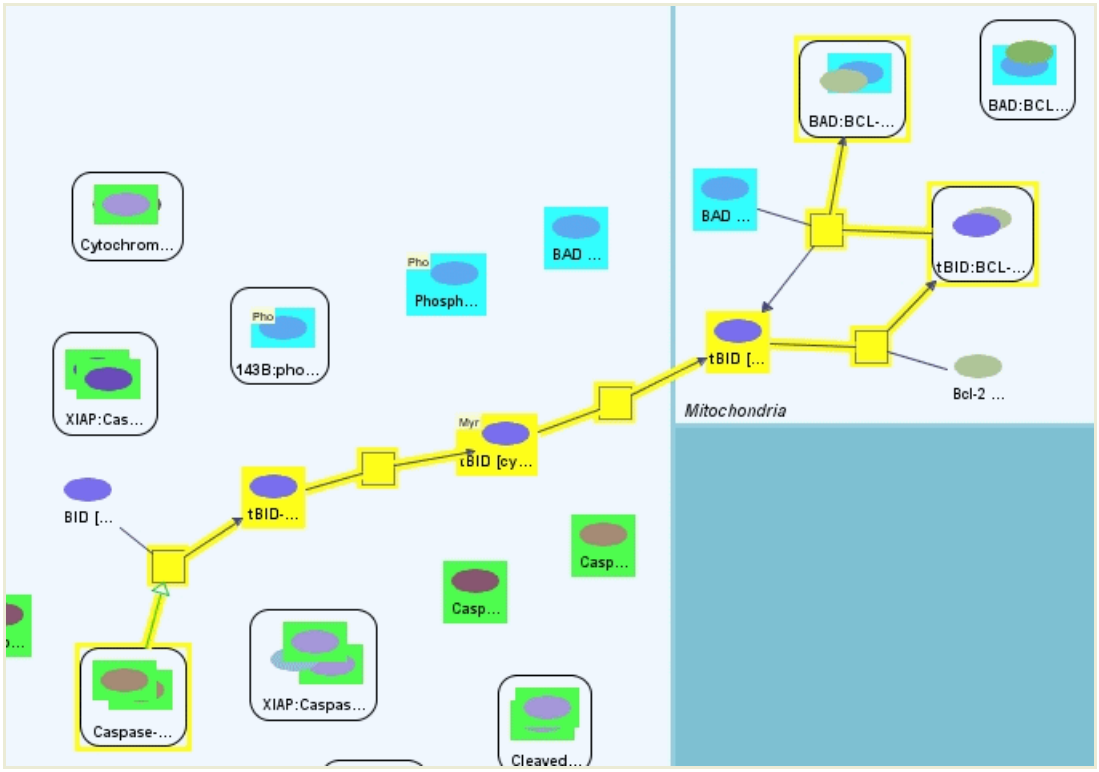


Figure 4-26 Result of the sample shortest paths query in Figure 4-25, where we find directed shortest paths (yellow) from simple states whose names contain "caspase" (green) to simple states whose names contain "bad" (cyan)

Feedback Query

This query results in a list of positive or negative cycles that contain a specified node and can be used in studying cellular networks (Figure 4-27). These queries can have signal amplifying and stabilizing roles and may help answer questions such as *"How is the concentration of a molecule stabilized?"* and *"How did a signal get amplified?"*

- ***limit*** specifies the maximum length of a cycle to be considered as a result of this query.
- ***sign*** tells whether we should filter the resulting cycles to be restricted to the positive or negative ones only.

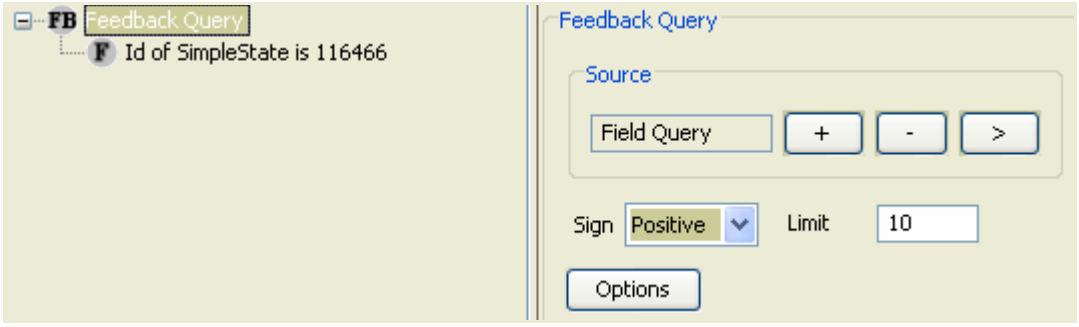


Figure 4-27 Sample Feedback Query Dialog

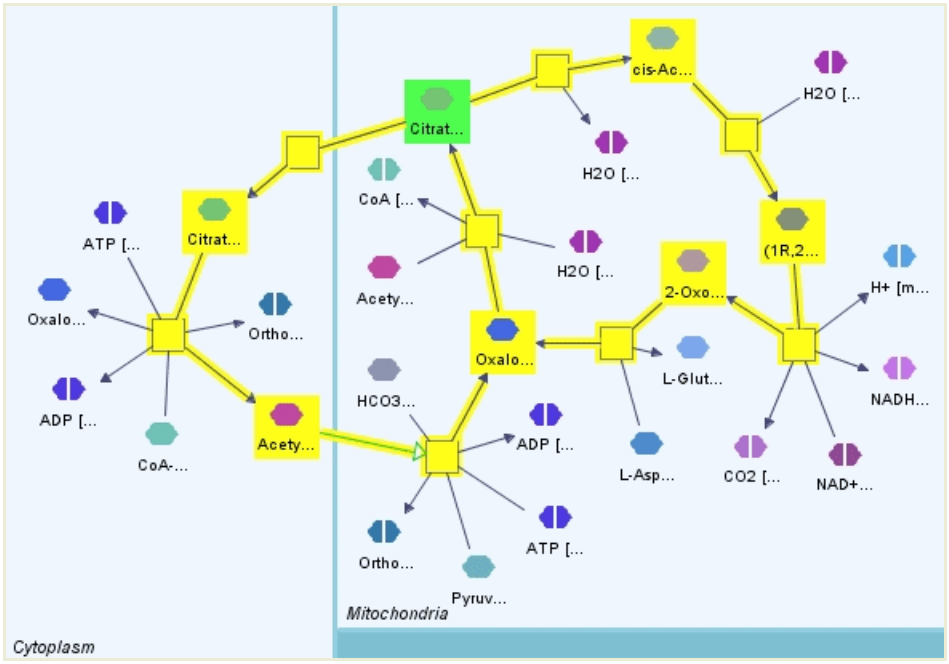


Figure 4-28 Result of the sample feedback query in Figure 4-27, where we look for positive feedback (yellow) of a given Citrate state (with specified ID) in mitochondria (green) with up to length 10; the result contains two feedback cycles, one in mitochondria (of length 10) and one through cytoplasm (of length 8).

Stream Query

This query may be used for analyzing *upstream* and *downstream* of a molecule (Figure 4-29). It has a significant role in retrieving cause/effect relationships which are essential for diagnosis, following up the development of certain events and drug design. In addition, these queries can provide answers to questions such as:

- *Which molecule activates this protein/molecule?*
- *Which processes are affected if this molecule/gene is knocked down?*
- *What are the downstream/upstream effects of certain drugs/molecules?*

The query is constructed by defining a source set as a sub-query, typically a field query, from the query dialog. The user also specifies the sign (i.e., positive or negative) of the path, and a threshold for the path length from the source or to the target. All nodes that have an outgoing/incoming path (for upstream and downstream, respectively) satisfying the constraints are returned, and highlighted with a different color.

- ***direction*** is used to specify whether we are to find the upstream of target objects or the downstream of source objects.
- ***limit*** specifies the maximum length of a path to be considered as part of the result of this query.
- ***sign*** tells whether we should filter the resulting paths to be restricted to the positive or negative ones only.
- ***ambiguity*** specifies whether or not ambiguous streams (multiple paths of conflicting sign with same source and destination) should be part of the result.

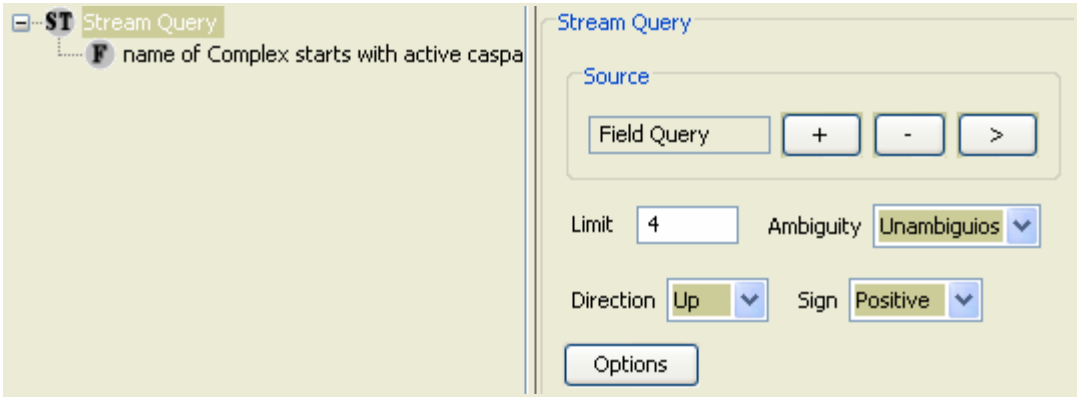


Figure 4-29 Sample Stream Query Dialog

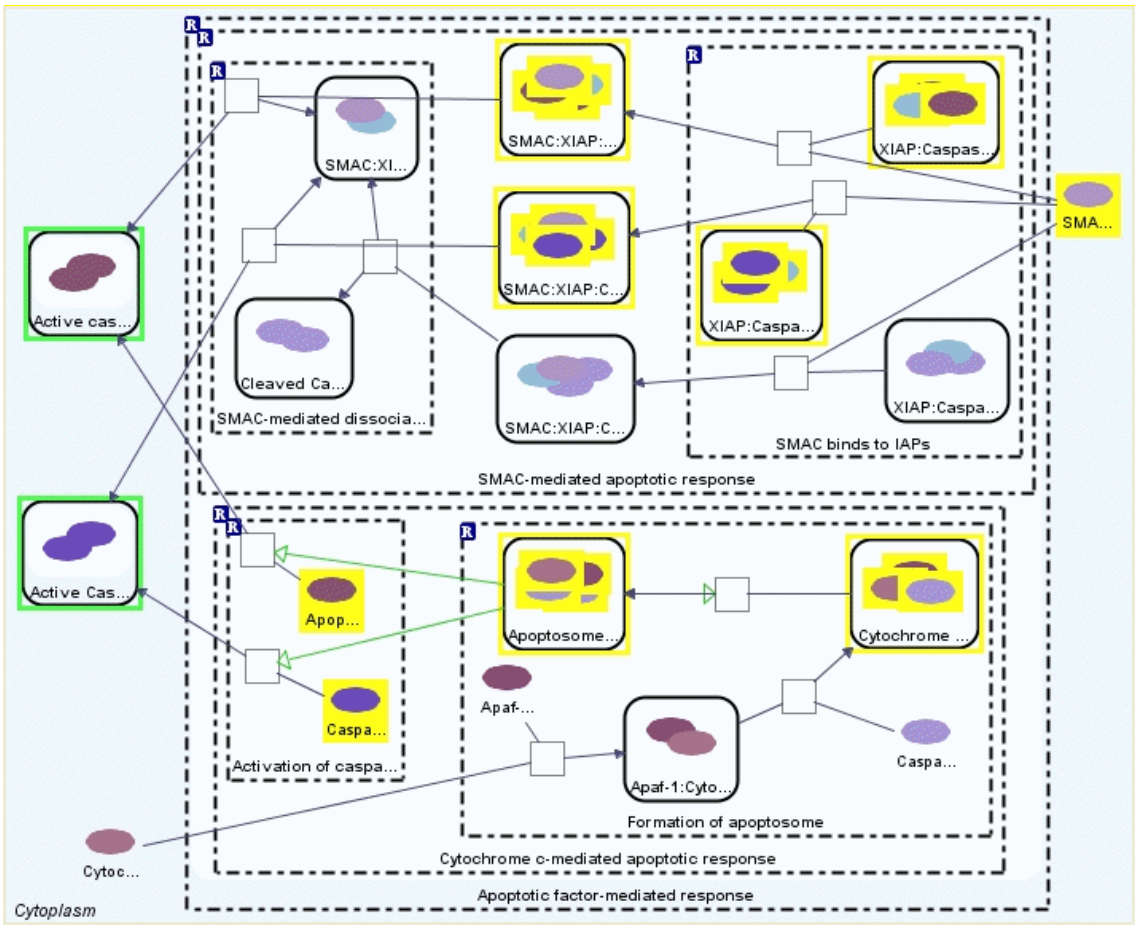


Figure 4-30 Result of the sample stream query in Figure 4-29, where we look for unambiguous positive upstream, with limit 4, (yellow) of complexes whose names start with "active caspase" (green)

4.5 Sample Session

Following is a sample session in which subsequent queries and complexity management operations are performed to form a model that might be of use to a PATIKA*web* user. Suppose the user is studying the effects of FAS Ligand on apoptosis. One good way to start is by searching for the relations between FAS Ligand and the Caspase complexes in the cell.

In order to find out the states of FAS Ligand in the cell, we perform the query in Figure 4-31, where we ask for simple states whose names start with "FASL".

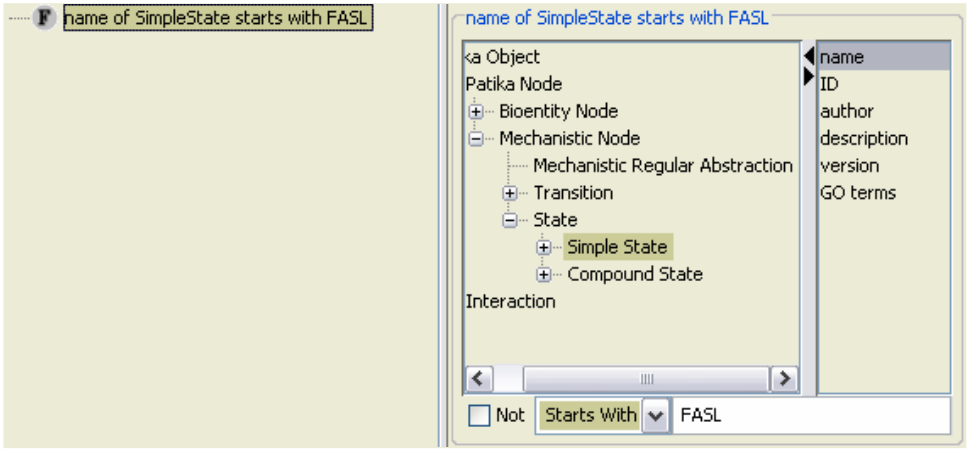


Figure 4-31 Query for simple states whose name starts with "FASL"

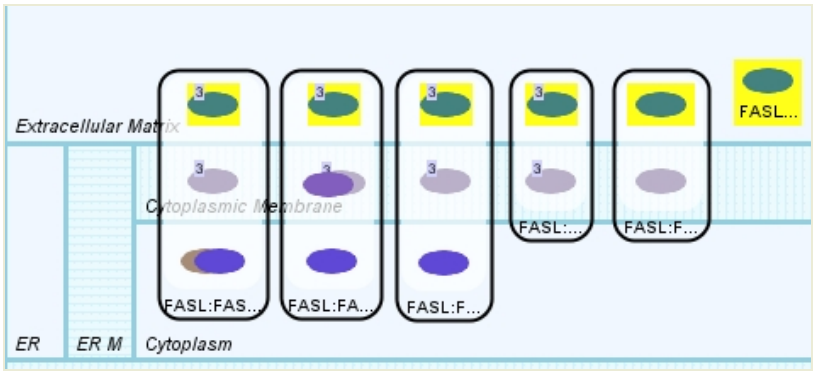


Figure 4-32 Result (yellow) of the FAS Ligand query in Figure 4-31

We see 6 states highlighted in the result of the query (Figure 4-32). One is the free extracellular FAS Ligand, and remaining ones are members of several complexes spanning the cytoplasmic membrane.

And we may check how many Caspase complexes we have in the database, which are not a precursor or a pro-caspase (Figure 4-33).

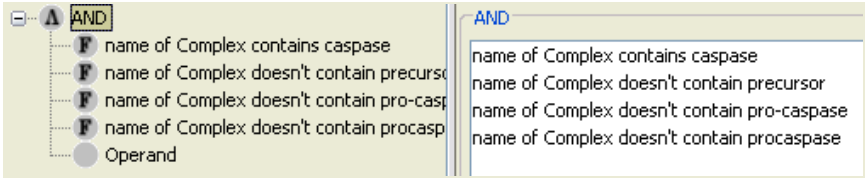


Figure 4-33 Query for Caspase complexes, which doesn't contain words "precursor", "pro-caspase" or "procaspase"

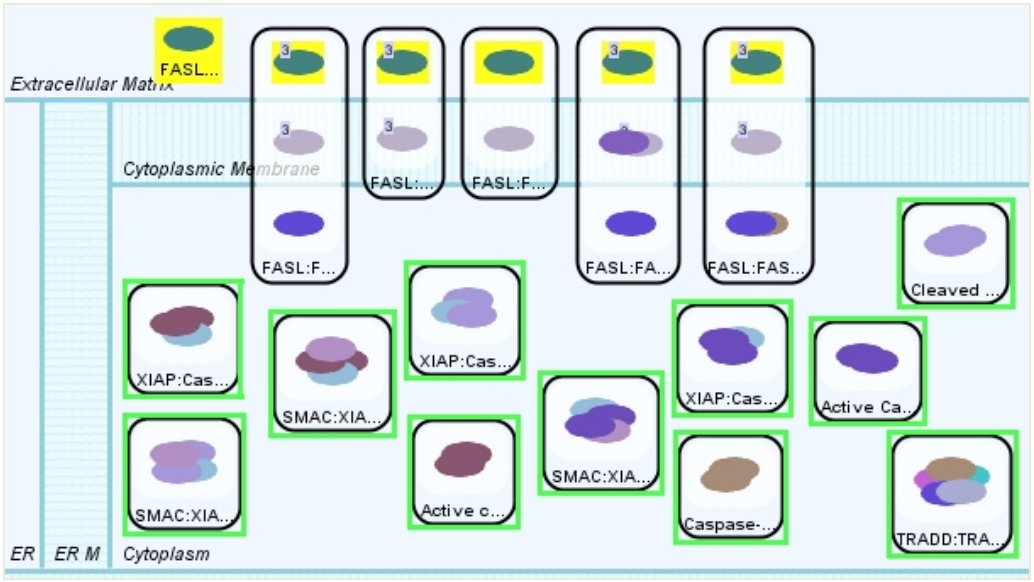


Figure 4-34 Result (green) of Caspase query in Figure 4-33 added to the existing model

Caspase query returns a total of 11 complex molecules, which are all in cytoplasm (Figure 4-34). Now we know that the database contents that we want to "start from" and we want to "reach to". The most popular query for finding relatively short paths between source and target molecules is the "Shortest Path Query" described earlier. We may use the previous FAS Ligand and Caspase field queries as the source and target fields of the shortest path query (Figure 4-35).

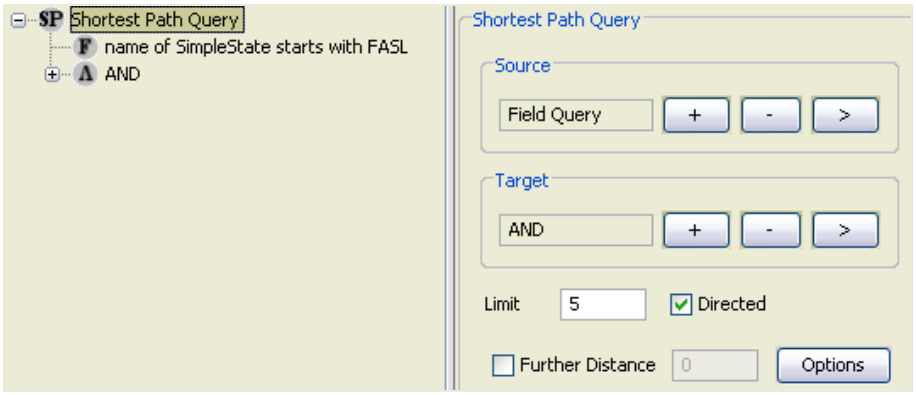


Figure 4-35 Shortest path query using the previous queries as source and target; the query limits the distance to 5 and considers directions.

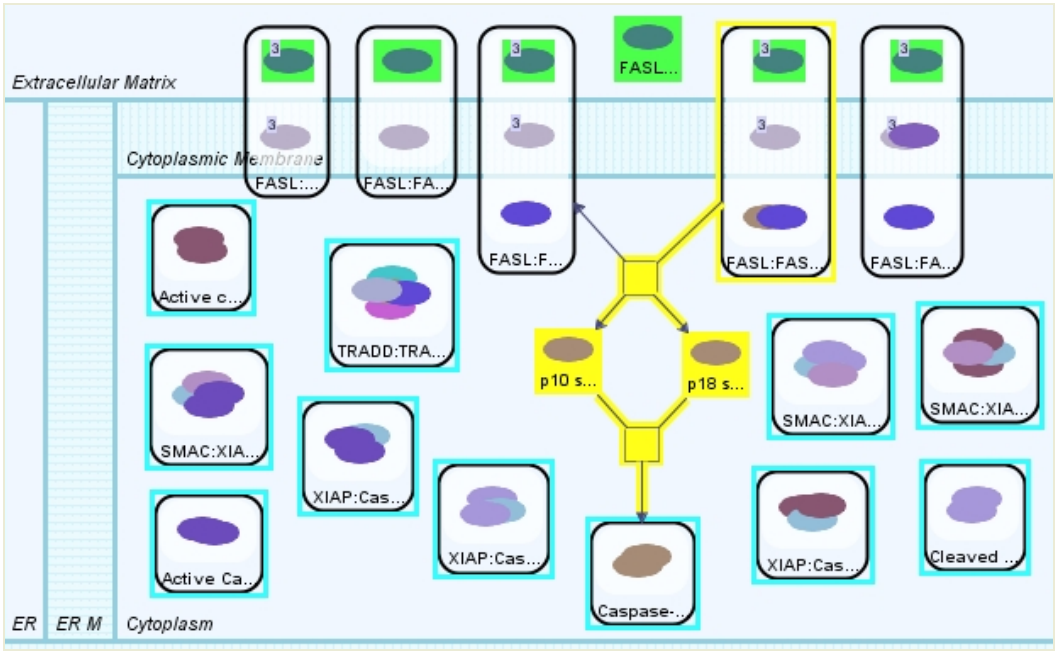


Figure 4-36 Result of the shortest path query in Figure 4-35; we find that the two shortest paths (yellow) from FAS Ligand (green) to Caspase complexes (cyan) goes to Caspase-8 dimer in cytoplasm, each with 2 transitions (4 steps).

Result of the shortest path query retrieves paths of length 4 (Figure 4-36). These are paths involving the FAS Ligand complex on the cytoplasmic membrane and the Caspase-8 dimer in cytoplasm. This picture might be very helpful but it still has many missing relations.

There are several ways to obtain a more complete picture. First alternative is to use the shortest path query with the "Further Distance" parameter. Figure 4-37 shows the same query with further distance set to 8. Since the shortest path length is 4, this query would bring us the paths from source to target nodes of length at most 12. Figure 4-38 shows the resulting model.

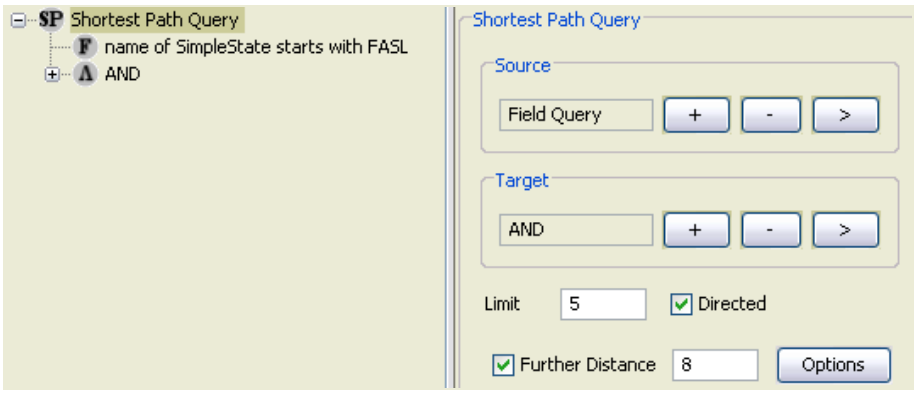


Figure 4-37 Shortest path query with further distance set to 8

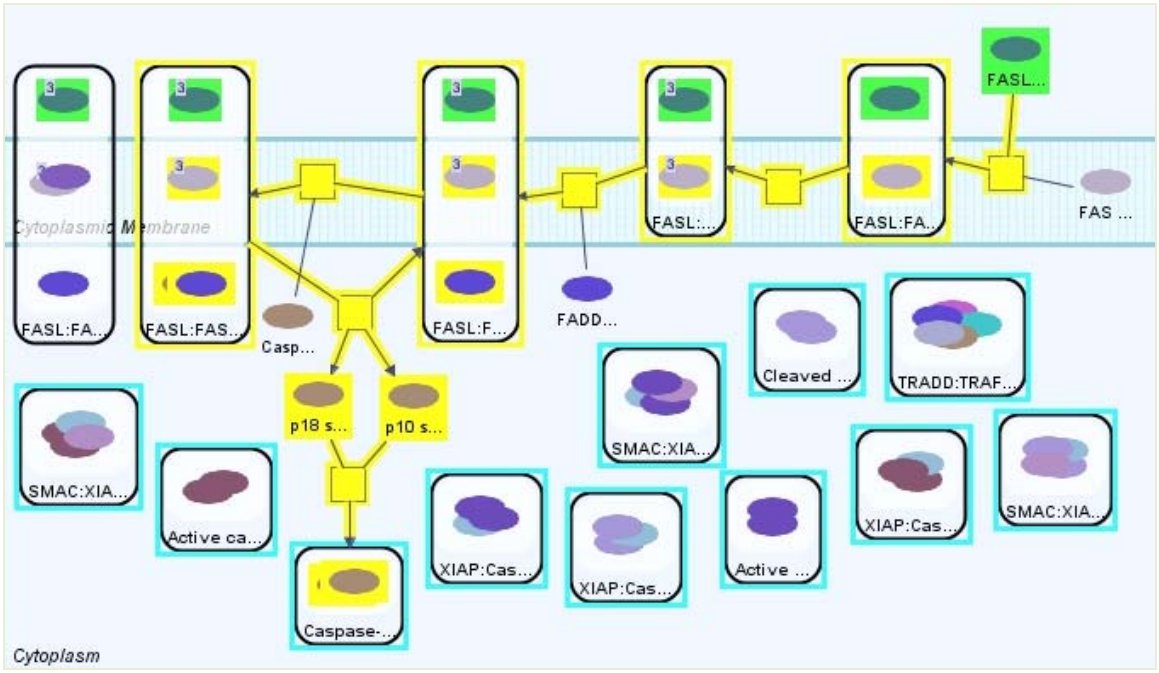


Figure 4-38 Result of the shortest path query in Figure 4-37. Paths of length up to 12 (yellow) are found between source (green) and target (cyan) sets, since the shortest path length is 4.

Another way of doing the same query is to use a "Paths-of-Interest" (PoI) query with limit 12. Since this query will bring all paths of length at most 12, between source and target sets, the result will be identical to the previous shortest path query with further distance 8. Thus PoI query is simply a more convenient way of querying paths when we have a good estimation of the length of the shortest path.

When finding paths between source and target sets is not sufficient, the user has the option to do a "Graph-of-Interest" (GoI) query. GoI query aims at completing the "missing links" (and molecules on these links) among a set of molecules of interest that is no longer than a specified limit. So a "minimal" graph including the specified objects of interest can be constructed through this query. Figure 4-39 shows a directed GoI query with limit 5, where the previous source and target sets are joined into an OR query as molecules of our interest.

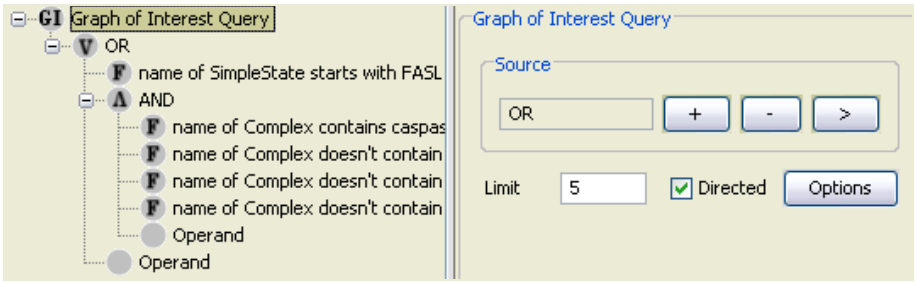


Figure 4-39 A GoI query where the previous FAS Ligand and Caspase complex queries are gathered into an OR query and used as seed (molecules of interest)

Since the GoI query finds all paths between a number of seed nodes (not from a specified source to specified target), the result contains more paths, not necessarily depicting a direction in the information flow. In the resulting model (Figure 4-40) we see that there are two isolated components. First one contains the previous FAS Ligand path we have found. We see that an additional Caspase complex is connected; however, the graph does not imply that this new Caspase complex has been involved in the FAS Ligand signaling process. Second component contains all other Caspase complexes. Notice that only two Caspase complexes have a relation with FAS Ligand

signaling process in the database (at least within the distance we have specified); the user may choose to concentrate on these for further analysis.

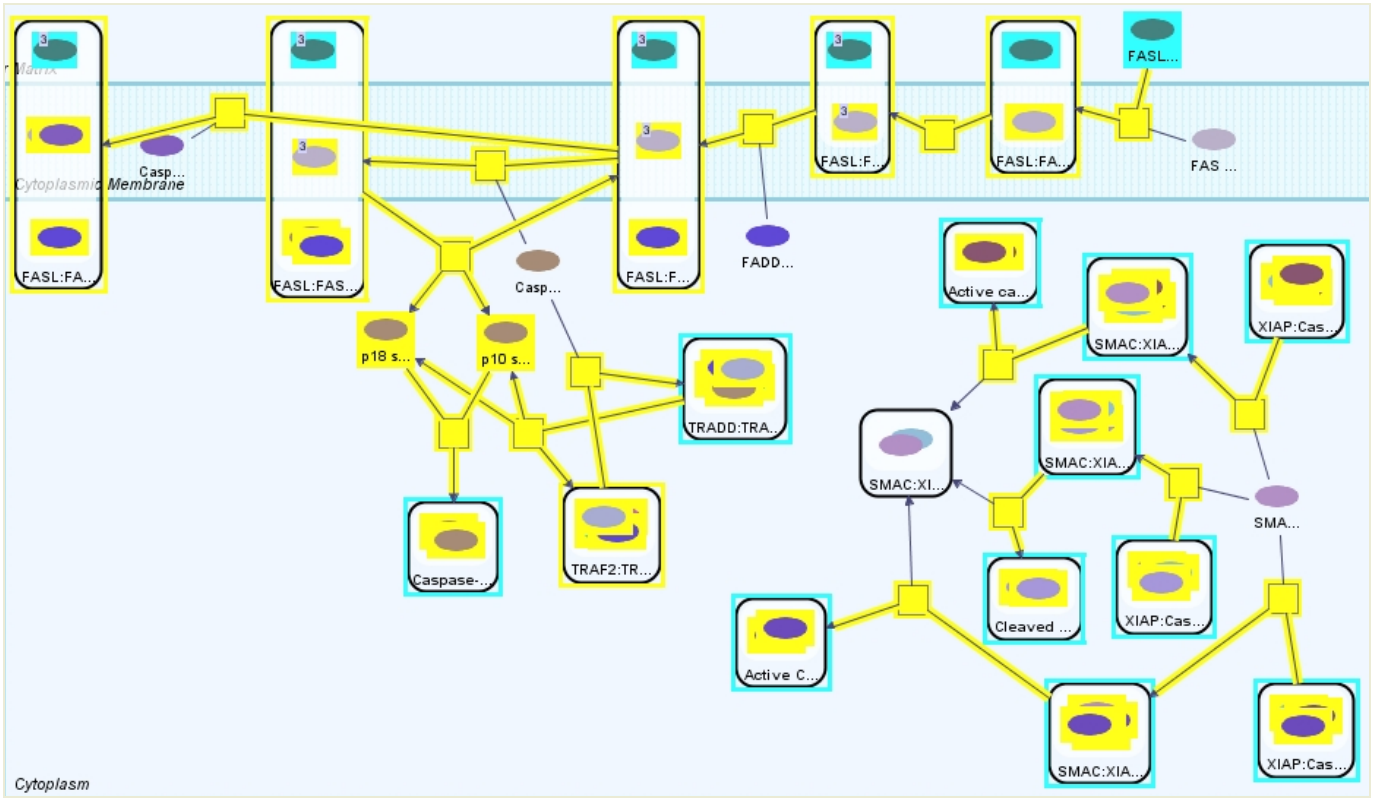


Figure 4-40 Result of the GoI query in Figure 4-39

Chapter 5

5 Microarray Data Analysis

Microarray technology helps us figure out the expression levels of thousands of genes in the cell simultaneously for different conditions. Nowadays such data is made available in many different formats. PATIKA*web* supports tab-delimited microarray data files which are most often produced by in-house experiments and some microarray databases such as NCBI Gene Expression Omnibus [23] or Stanford Microarray Database [24]. In PATIKA*web* it is possible to map microarray data onto bioentities in a bioentity view and simple states in a mechanistic view. This mapping can be visualized by color-coding and/or labeling the involved objects in both types of views (Section 5.3). In addition, such data may be analyzed using clustering methods (Section 5.4).

5.1 Conversion into Native Format

Prior to analysis of microarray data in PATIKA*web*, it needs to be converted to the **P**ATIKA **M**icro**a**rray **D**ata format (PMAD). PMAD is an XML-based format containing information about an experiment set (description and values to be analyzed together with corresponding PATIKA IDs). The "Microarray Data Conversion Wizard" in PATIKA*web* (Figure 5-1) assists in converting a tab-delimited data file into PATIKA PMAD format. For instance, conversion of NCBI GEO files and Stanford Microarray Database files are supported since they are tab-delimited text files.

The tab-delimited data file should contain some external reference IDs that can be matched with PATIKA IDs. Reference matching can be performed at the PATIKA database or on the locally loaded model providing that the model elements have some external references as their imported data sources. For the matching at the PATIKA database option, there are 8 supported major database reference IDs as listed below:

- NCBI GenBank
- HUGO Gene Symbol
- NCBI Unigene
- NCBI Gene (LocusLink)
- Swiss-Prot
- OMIM
- NCBI Ref Seq Protein
- NCBI Ref Seq Transcript

In the following demonstrations, it is assumed that reference matching at the database option is selected. Details of local matching (matching on current model) will be provided later in this section.

After the conversion is performed, a ".pmad" file is prepared for local storage.

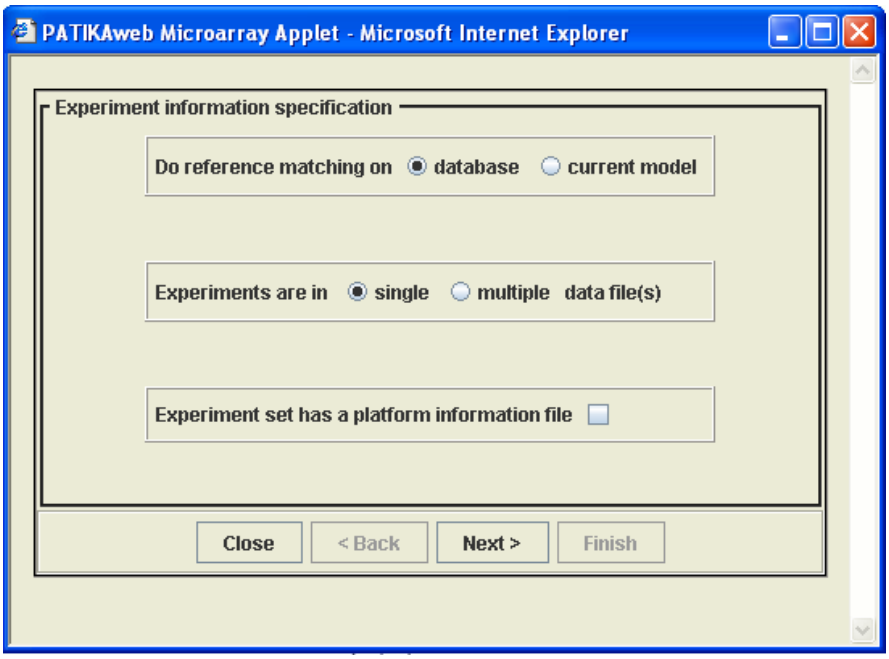


Figure 5-1 Microarray Data Conversion Wizard

A tab-delimited file consists of rows and columns. Typically, each row represents a gene and the columns along a row represent the values measured. These values might be normalized amounts measured in different experiments or different types of measurements extracted from a single experiment.

Platform Files

Sometimes it is reasonable to specify platform specific data in a file, independent from specific experiments. Platform files generally contain many external references that correspond to some spots in the microarray chip, which are specified by an internal ID. Platform files help reduce size of data files by not repeating external references.

Platform file:

Internal ID	GenBank Accession	Gene Symbol	Unigene
473847	AB000114	OMD	Hs.94070
473625	AB000115	IFI44L	Hs.389724

Data files:

Internal ID	Value
473847	100.4
473625	152.7

Internal ID	Value
473847	20.3
473625	263.1

When there is no platform file, these references are written into data files.

Data files:

Internal ID	GenBank Accession	Gene Symbol	Unigene	Value
473847	AB000114	OMD	Hs.94070	100.4
473625	AB000115	IFI44L	Hs.389724	152.7

Internal ID	GenBank Accession	Gene Symbol	Unigene	Value
473847	AB000114	OMD	Hs.94070	20.3
473625	AB000115	IFI44L	Hs.389724	263.1

Converter needs external references to map rows with PATIKA objects, and supports both kinds of formats.

Multiple experiments

Several experiment results may be presented in a single data file or each experiment may be given in a separate data file (as given above). When given in a single file, values of each experiment are presented in a separate column.

Two experiments in a single file (assuming there is a platform file not given here):

Internal ID	Experiment 1	Experiment 2
473847	100.4	20.3
473625	152.7	263.1

Using Conversion Wizard

In the first page, the user specifies if experiments are given in a single or multiple data files, as well as whether or not there is a platform file (Figure 5-1). The following wizard pages appear accordingly as described below.

SINGLE DATA FILE WITH A PLATFORM FILE

When there is a platform file and data is in a single file, then the wizard prompts for the platform file, parses it and tries to map its columns to some known references (Figure 5-2). Some column names may be automatically identified, whereas others should be mapped by the user. The "Key to Data File(s)" item should be mapped to the internal ID column, which data files use.

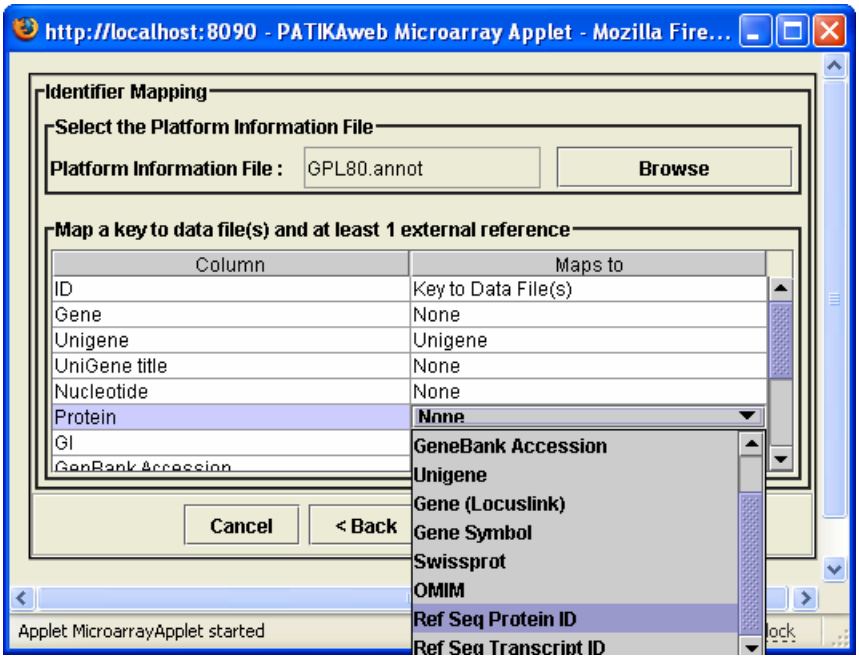


Figure 5-2 Platform file is parsed and some reference columns are mapped. The "Key to Data File(s)" item should be mapped to the internal ID column, which data files use.

The next page is for specifying the data file to be loaded. After that, internal IDs and experiment value columns are asked for (Figure 5-3). All experiment values must be in adjacent columns.

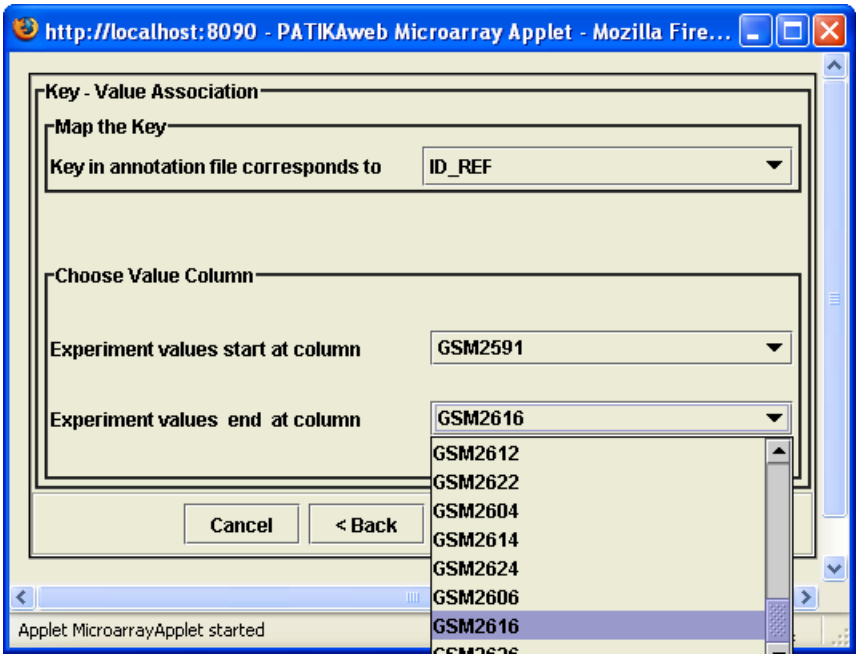


Figure 5-3 Key – value association with single data file containing multiple experiments. Here "ID_REF" column is said to be the internal ID that related the data file to the platform file.

MULTIPLE DATA FILES WITH PLATFORM FILE

When there is a platform file and each of multiple experiments is given in a separate file, the user is first asked to locate these data files (Figure 5-4).

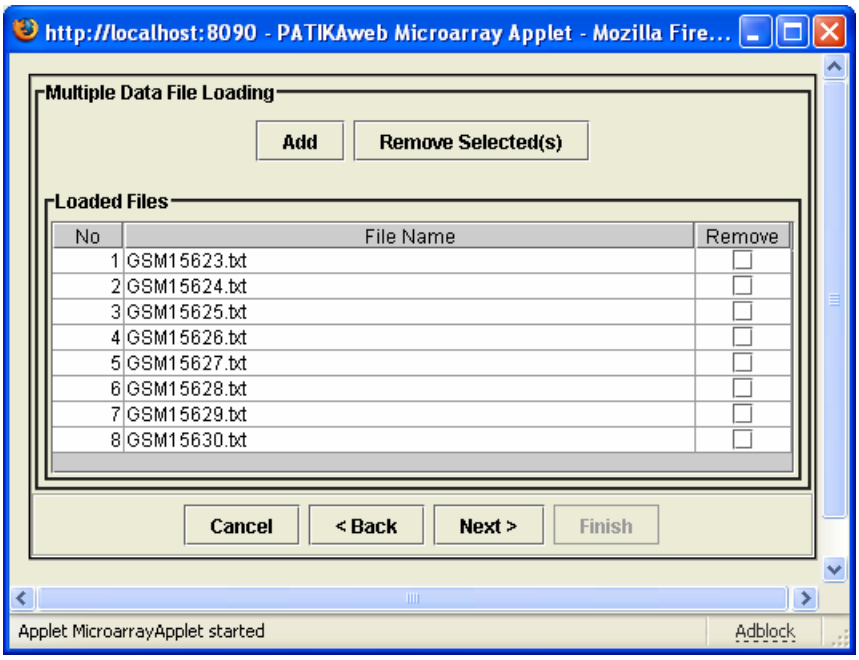


Figure 5-4 Multiple Data File Loading

Then the internal ID column (key to annotation file) and the value column are set by the user (Figure 5-5). Here, the format of the data files is assumed to be the same; thus these settings should apply to each loaded data file, otherwise mismatches may occur.

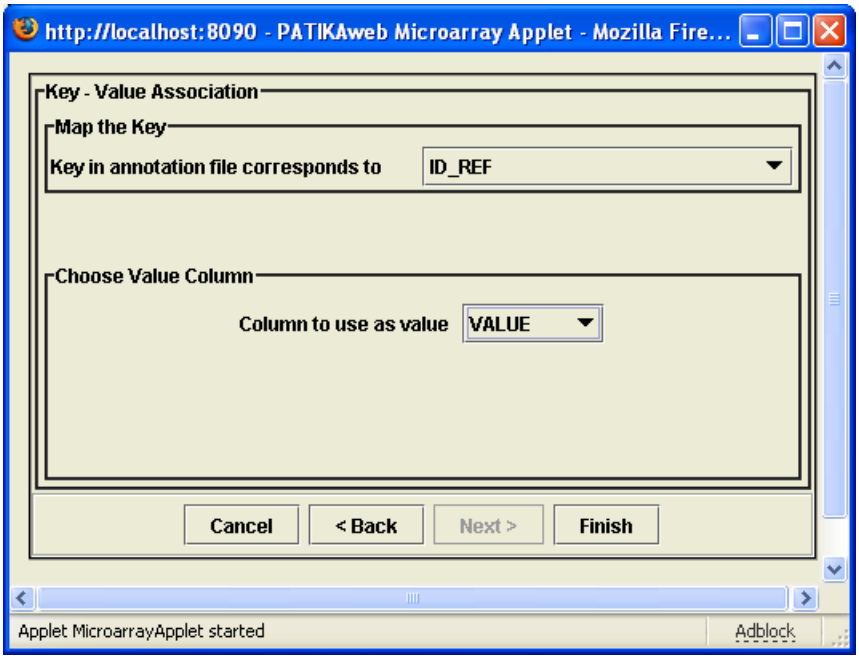


Figure 5-5 Key – value association with multiple data files

SINGLE DATA FILE WITH NO PLATFORM FILE

When there is no platform file and all data is presented in a single data file, the user should select the data file (Figure 5-6) and specify the data columns that contain experiment values (Figure 5-7). Columns containing the experiment values are assumed to be adjacent in the data file. Since there is no platform file, the user should also specify external reference columns.

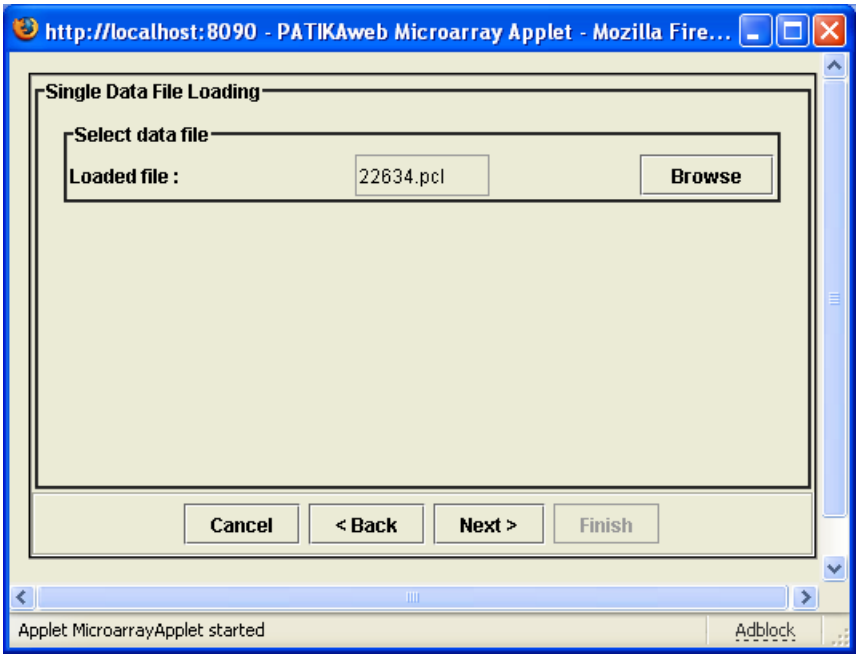


Figure 5-6 Loading single data file that contains multiple experiments which has no platform file

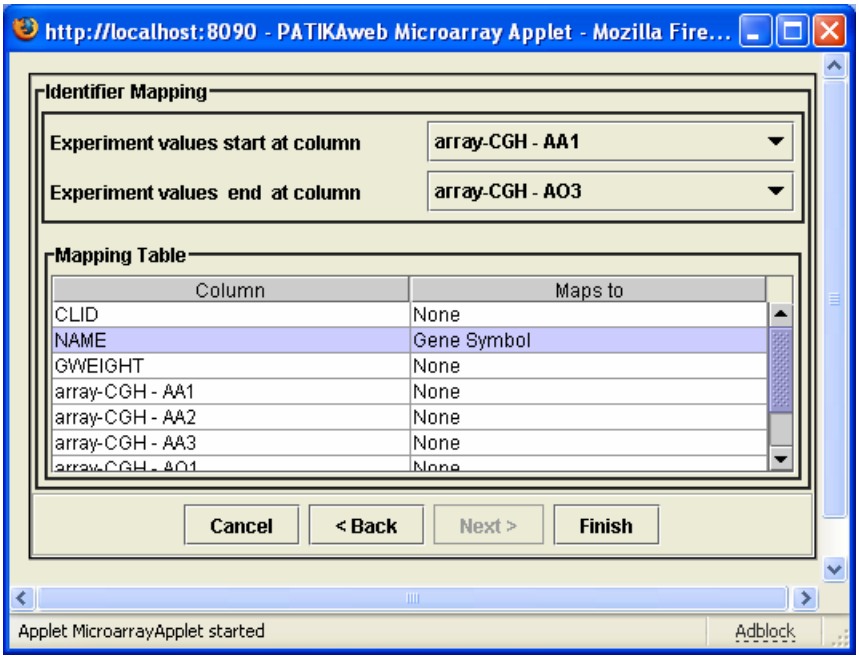


Figure 5-7 ID mapping when there is a single data file with no platform file

MULTIPLE DATA FILES WITH NO PLATFORM FILE

When there is no platform file and values of each experiment are provided in a separate data file, the user first specifies the data files (Figure 5-8). Then he selects the value column and external reference columns in data files (Figure 5-9). The format of all data files should be the same; otherwise files will not load successfully.

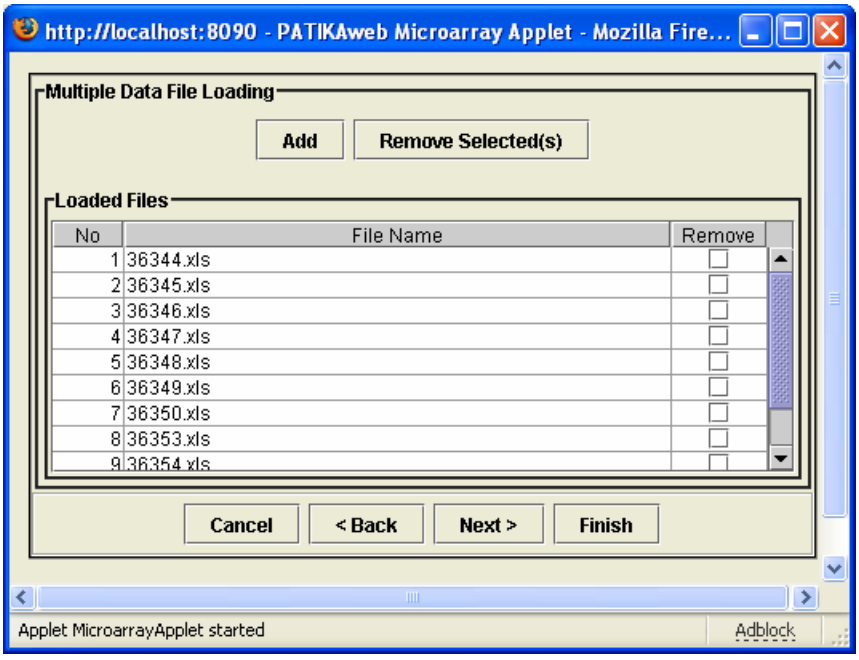


Figure 5-8 Loading multiple data files with no platform file

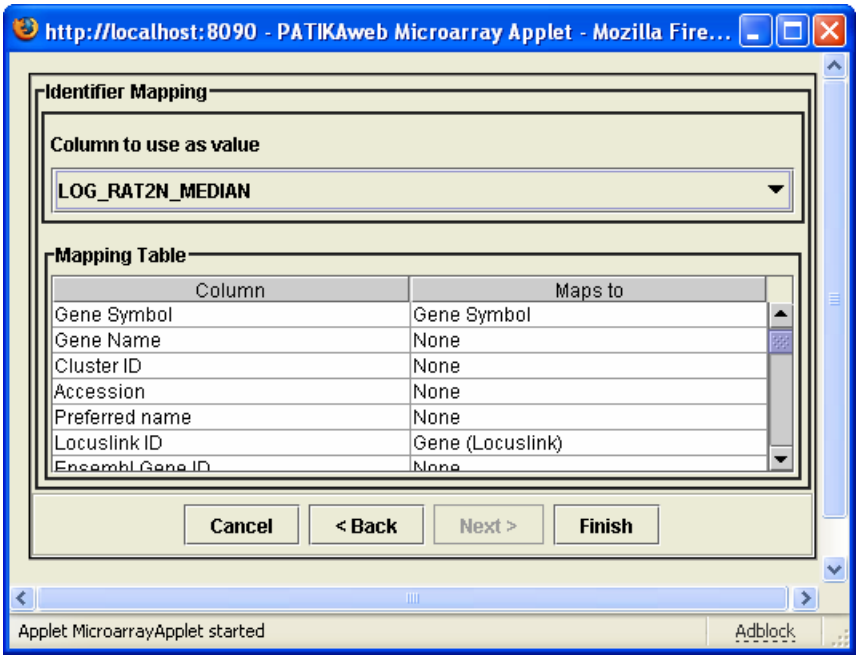


Figure 5-9 ID mapping when there are multiple files with no platform file

Local Reference Matching

As mentioned earlier, above demonstrations are based on matching the genes in the microarray data with the PATIKA objects using external references stored in the PATIKA database. It is also possible to import a model from other pathway databases (say using the BioPAX format) and perform microarray data analysis on that specific model. Since the model is locally imported, its elements will not exist in PATIKA database. In such a case, reference matching on the local model option must be selected during data conversion as shown in Figure 5-10.

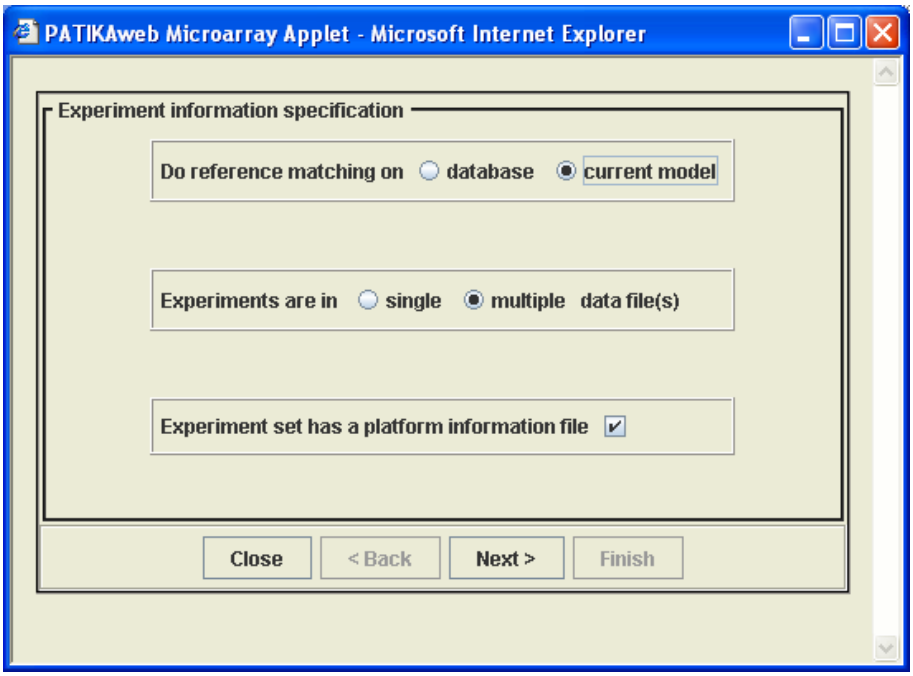


Figure 5-10 Microarray Data Conversion Wizard with local reference matching (matching on current model) option selected

During column mapping reference types found in microarray data files must be mapped to the ones found in the current model as shown in Figure 5-11. To get a good performance in matching, local model elements should be annotated as much as possible. PATIKA automatically extracts external references from the loaded models or from imported BioPAX models.

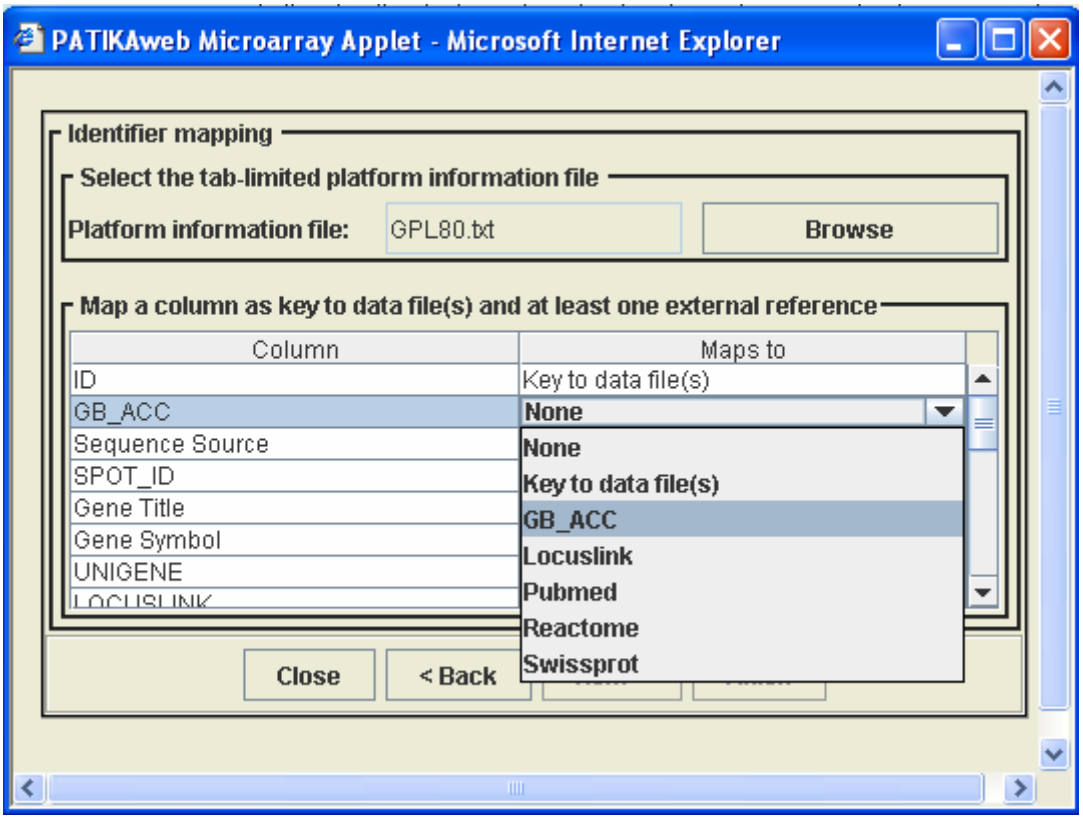


Figure 5-11 Platform file is parsed and some reference columns are mapped. The "Maps to column" refers to the external references found in the current model.

The rest of the steps are the same as the steps of reference matching on the PATIKA server explained earlier. Note that, the '.pmad' file constructed for local persistence will not work with any other model because it will be model-specific. Similarly, ".pmad" files constructed by matching references at the PATIKA database will not work with models that do not belong to PATIKA database.

5.2 Loading Microarray Data

As desired, ".pmad" files can be loaded into PATIKA*web*. Upon load, all visualized objects of the current pathway model are associated with the first experiment in the loaded data file. As the pathway models change on new queries for instance, the loaded data is associated with any newly introduced objects whenever possible.

5.3 Visualizing Microarray Data

Figure 5-12 shows the mechanistic view of a pathway model with microarray data mapped onto the pathway model objects.

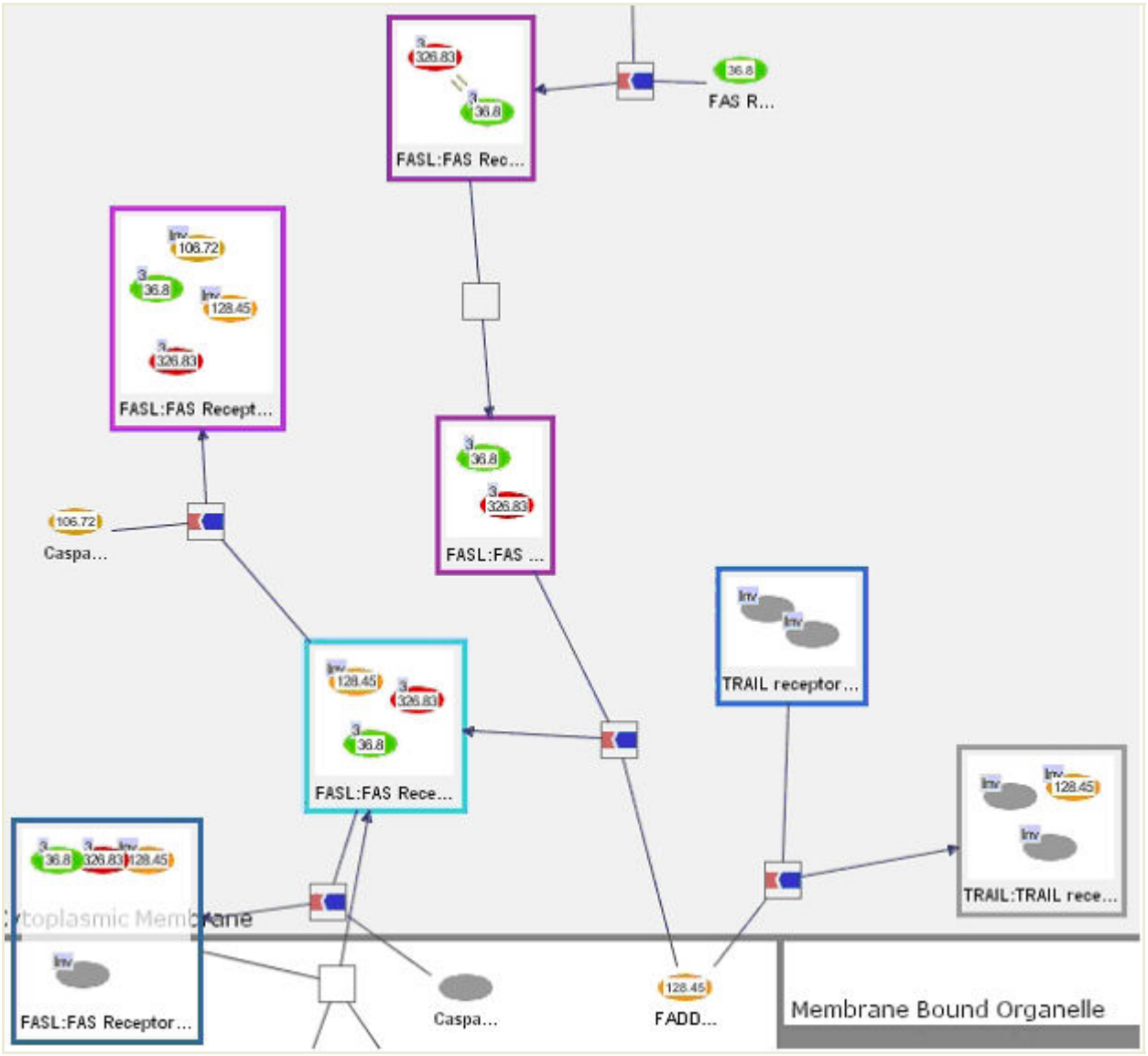


Figure 5-12 The mechanistic view of a pathway model with microarray data mapped onto the pathway model objects. The data is shown by color-coding (whose settings are provided by the dialog in Figure 5-14) and labeling (values of expression discrepancies).

Microarray data being currently visualized can be managed through the "Microarray Data Management Dialog" (Microarray | View Data Manager) shown in Figure 5-13. In the top area, general information about the loaded experiments is displayed. The right panel shows information on the selected experiment, which are listed in the left panel. The user may choose the option to visualize a single experiment by ticking only one of the experiments, or alternatively choose to compare two experiments and tick two of the experiments. Upon clicking "Update", the colors and/or values of both bioentity and mechanistic objects will change.

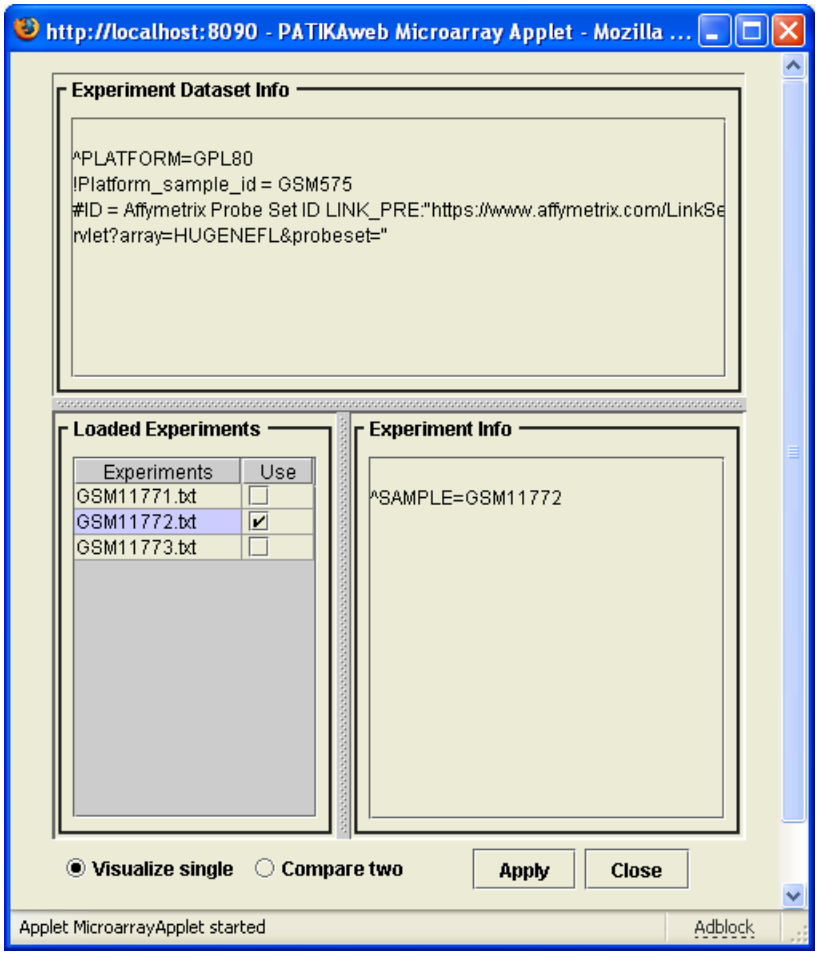


Figure 5-13 Microarray Data Management Dialog

Configuring Visualization Settings

Visualization options for microarray data can be configured using "Microarray Settings Dialog" (Microarray | View Visual Settings) in PATIKA*web* (Figure 5-14). You can specify any number of colors and corresponding values for desired color-coding. Then in-between values are displayed with in-between colors computed accordingly. Alternatively, the user may select the option "Auto-calculate values" and the color coding will be done between the two selected colors ranging from the minimum and maximum values in the experiment set(s).

Furthermore, optionally, values can be displayed on related pathway view objects as labels in addition to the color-coding of the objects by clicking on the option "Display values on view".

The microarray data will not be visualized on views unless "Visualize microarray data on view" in this dialog is checked. When you're finished with analysis of a microarray data set, you may want to perform "Microarray | Discard Data" to de-allocate this data.

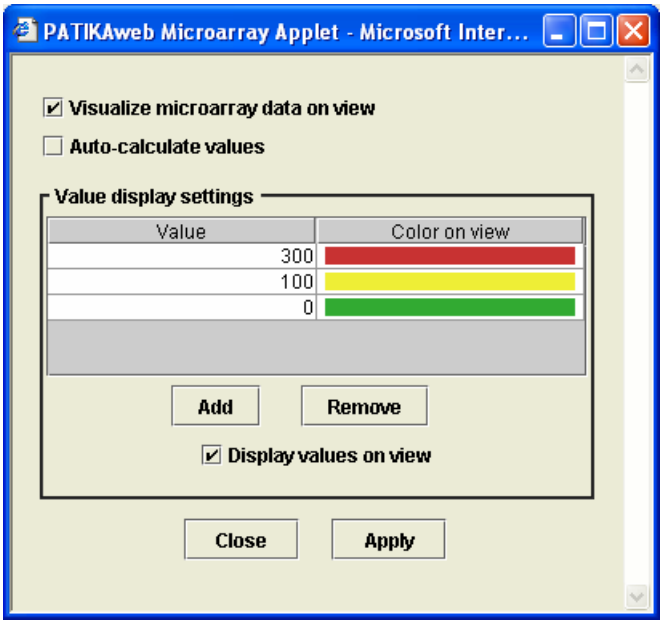


Figure 5-14 Visual Settings Dialog

5.4 Cluster Analysis

The purpose of microarray cluster analysis is to group genes on the basis of similarity/dissimilarity of their expression profiles. As molecular biologists are performing more and more microarray experiments, they become more dependent on cluster analysis and other biostatisticial methods because it is almost impossible to make sense of whole-genome expression data manually. Cluster analysis of microarray data has already demonstrated great potential for disease identification, finding genes responsible for specific diseases [25] and drug discovery [26].

There are two types of cluster analysis tools. Tools in the first group solely perform clustering and their visualization power, if any, is limited to a 2-D Cartesian plot or a dendogram. Tools in the second group are pathway visualization tools that include microarray data analysis components. However, their visualization power is limited to color coding. In this respect, PATIKAweb is unique in its integrated tools to perform cluster analysis and visualize the results as partitioned pathways.

The "Cluster Analysis" menu is a sub-menu located within the "Microarray" menu. Prior to performing cluster analysis, microarray data needs to be converted to native format and loaded as described in sections 5.1 and 5.2. Below is some background information on clustering analysis and steps to perform clustering analysis and cluster visualization in PATIKAweb.

Background on Clustering Analysis

In order to perform cluster analysis correctly, methods and parameters for clustering should be clearly understood.

- **Normalization:** Microarray data normally undergoes a pre-processing prior to be submitted to major microarray databases. However, it is not enough because errors

introduced in the pre-process needs to be eliminated in order differentiate between biological variations and variations due to the measurement errors.

Even when the processes are error-free, one needs to perform normalization for a correct statistical analysis. To illustrate, in the bellow diagram un-normalized and normalized expression profiles of genes A, B and C extracted from 8 experiments are shown. If normalized expression profile is used, A and B are placed in the same cluster. Biologically, A and B display a strong co-expression and it is reasonable for them to be in the same cluster after a cluster analysis. However, if un-normalized expression profile is used, A and C will be considered closer than A and B from the computational point of view. This is because using a Euclidian distance metric for clustering will result in A and C to be in the same cluster while B in a different cluster [27].

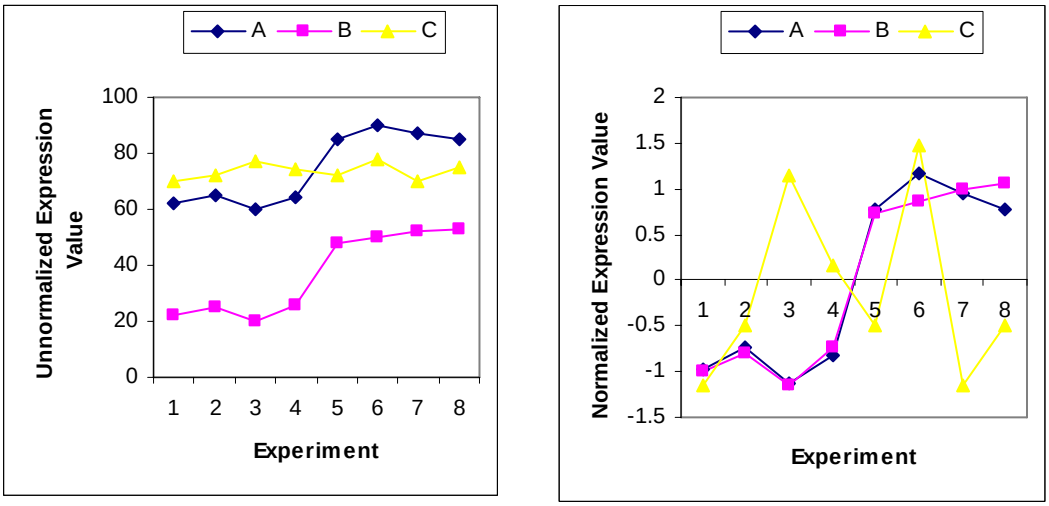


Figure 5-15 Normalization of genes A, B and C clearly reveals the correlation between genes A and B.

An oligonucleotide array measures actual amount of gene expression in a sample while cDNA array measures the gene expression difference between a control sample and a test sample. Since, cDNA arrays and oligonucleotide arrays are very different in representation of expression information, they are handled separately. PATIKAweb supports normalization of oligonucleotide arrays. There are a lot of normalization techniques; below are the most common ones used in oligonucleotide microarray data analysis.

- Scale Normalization of Oligonucleotide Arrays: Scale normalizing makes average intensities of all the arrays the same (200) by multiplying genes of each array with the normalization factor calculated for that array.

- Standard Normalization of Oligonucleotide Array Genes: Since, clustering algorithms are very sensitive to distance measures and these distance measures are based on gene expression levels at different experiments, genes need to be

standardized across experiments in order to make expression level of each gene with mean 0 and standard variance 1. This normalization method eliminates the scaling differences between genes. This provides the algorithms with the ability to prevent mistakes originated from expression scales of different genes and focus on the pattern rather than individual expression characteristics of a gene.

- **Filtering:** Not all genes provide useful information during cluster analysis. Even worse, some genes cause noise in data. Genes that do not show much variation through samples can be eliminated. To do filtering, arrays are scale-normalized and then genes are sorted according to their variance. Genes in the lower part are filtered out. Since arrays are scale-normalized prior to filtering, scale normalization is disabled upon selection of filtering check box to prevent duplicate normalization.

- **Distance Metrics:** A distance metric is a way of measuring similarity between genes. Since microarray data is multi-dimensional, genes need to be related with each other using a distance metric. Along many of them, below are the most popular distance metrics used for measuring similarity/dissimilarity of genes. Details of metrics are not given here and the user is kindly asked to view [29] for a brief summary.
    - Euclidian Distance
    - (Absolute) Pearson Correlation
    - Manhattan Distance
    - Canberra Distance
    - Minkowski Distance
- **Clustering Algorithms:** Among various unsupervised clustering algorithms, k-Means and Agglomerative Hierarchical clustering algorithms are supported. PATIKA*web* makes use of clustering packages of the R language for this purpose [30]. The user needs to specify number of clusters in both cases. In the hierarchical clustering case, the clustering dendogram will be cut according to the given number of clusters. Details of these algorithms will not be given here and for detailed information user is kindly asked to read Sections 5.1 and 5.2 of [31].

Cluster Analysis Dialog

This dialog gathers the required parameters to perform clustering.

For Hierarchical Clustering method the user needs to specify distance metric, linkage method and number of clusters as shown in Figure 5-16. Normally, hierarchical clustering algorithm outputs a hierarchic similarity of genes not discrete clusters. However, it is possible to have discrete clusters by cutting the hierarchy tree (dendogram) at specific points.

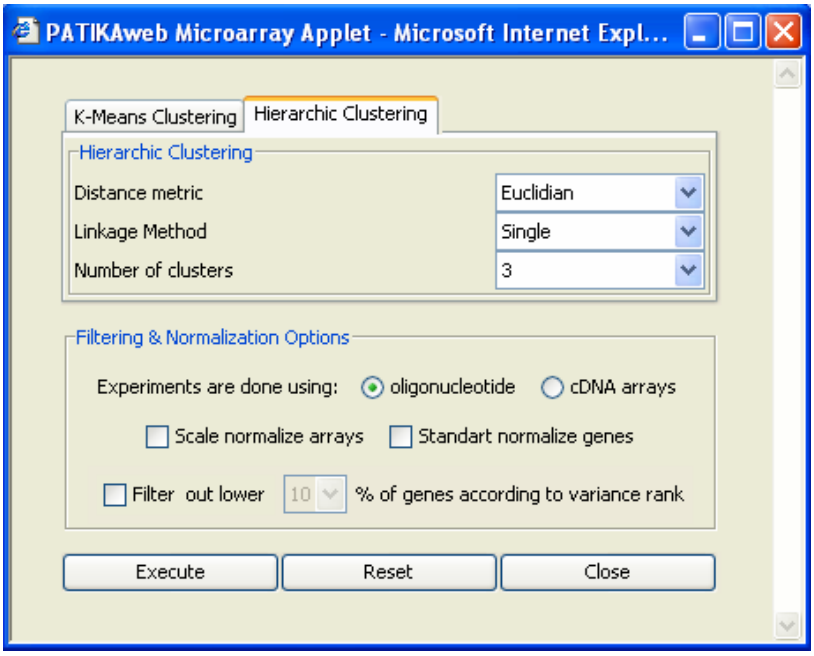


Figure 5-16 Cluster Analysis Dialog with Hierarchic Clustering method selected

For k-means Clustering method the user needs to specify the number of clusters and maximum number of iterations as shown in Figure 5-17. "Euclidian Distance" metric is used by default.

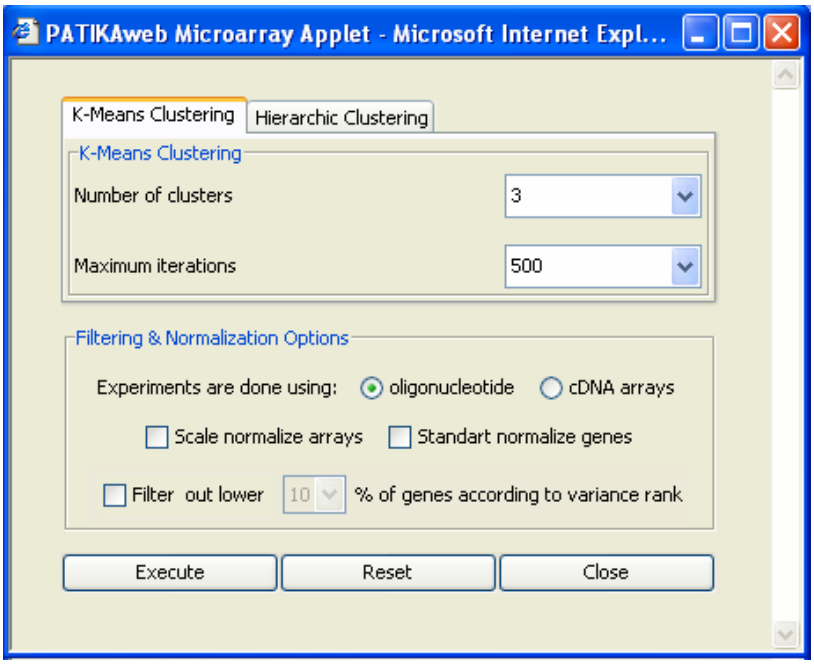


Figure 5-17 Cluster Analysis Dialog with k-Means Clustering method selected

Normalization and filtering of oligonucleotide arrays are optional. In case of normalization, the user needs to specify the type of normalization method(s). Note that, both types of

normalization methods can be applied on data at the same time. For the filtering option, user needs only to specify lower cut percentage of genes.

The clustering of data is done at the server side. Upon pressing the execute button, the dialog will send the data and clustering parameters to the server. After clustering finishes, the server sends an XML based **P**atika **C**lustering **A**nalysis **F**ile (pcaf) for persistence storage.

Loading Cluster Analysis File

As desired, ".pcaf" files can be loaded into PATIKA*web*. This will enable the visualization of clusters, assuming a valid cluster analysis file has already been loaded.

Cluster Visualization Dialog

Visualization of clusters can be done at both bioentity and mechanistic levels depending on the type of current view.

Basically clustering information can be visualized in two ways. One is using the highlighting facility in PATIKA. That is, each cluster is assigned a unique color and all biological objects in that cluster are highlighted using this color. Alternatively, a regular abstraction is created for each cluster containing all objects in that cluster (Figure 5-18 Figure 5-20).

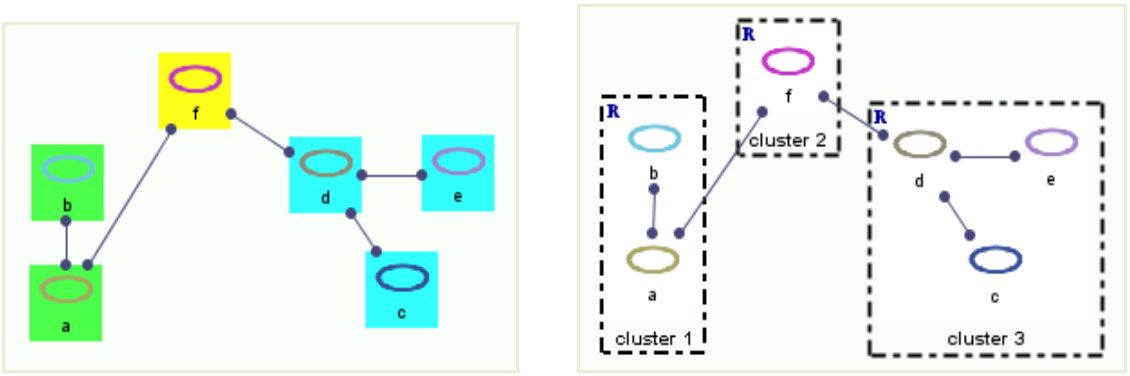


Figure 5-18 A hypothetical bioentity model that is partitioned into three clusters: *{a,b}*, *{e,d,e}* and *{f}*. On the left, each cluster is highlighted with a different color. On the right, each cluster is grouped and presented within a regular abstraction.

At the mechanistic level, there are three options as shown in Figure 5-19. Note that, during visualization, only genes' all simple state and complex member state products are visualized. If a complex has members from different clusters, that complex will not be highlighted or put into any cluster abstraction.

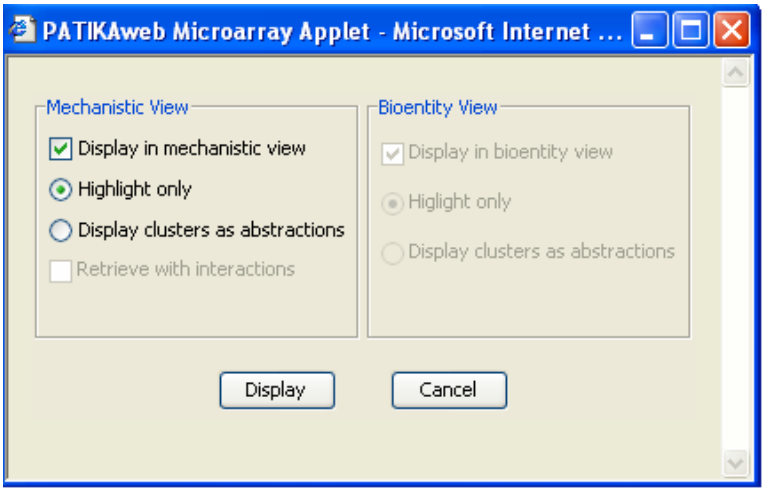


Figure 5-19 Cluster Visualization Dialog

Upon selection of 'Highlight Only' option elements of each cluster will be highlighted with a different color as shown in Figure 5-20.

Note that, there may be nodes that are not highlighted. There are several reasons for this. Firstly, the node may be unidentified during data conversion into native format process because either it has no associated data or its external references do not exist in PATIKA*web*'s external references database. Secondly, the node may be eliminated during clustering due to missing values of this node in some of available experiments. The reason for retrieval of these genes is to form a minimally connected pathway by bridging the gap among the actual genes of interest.

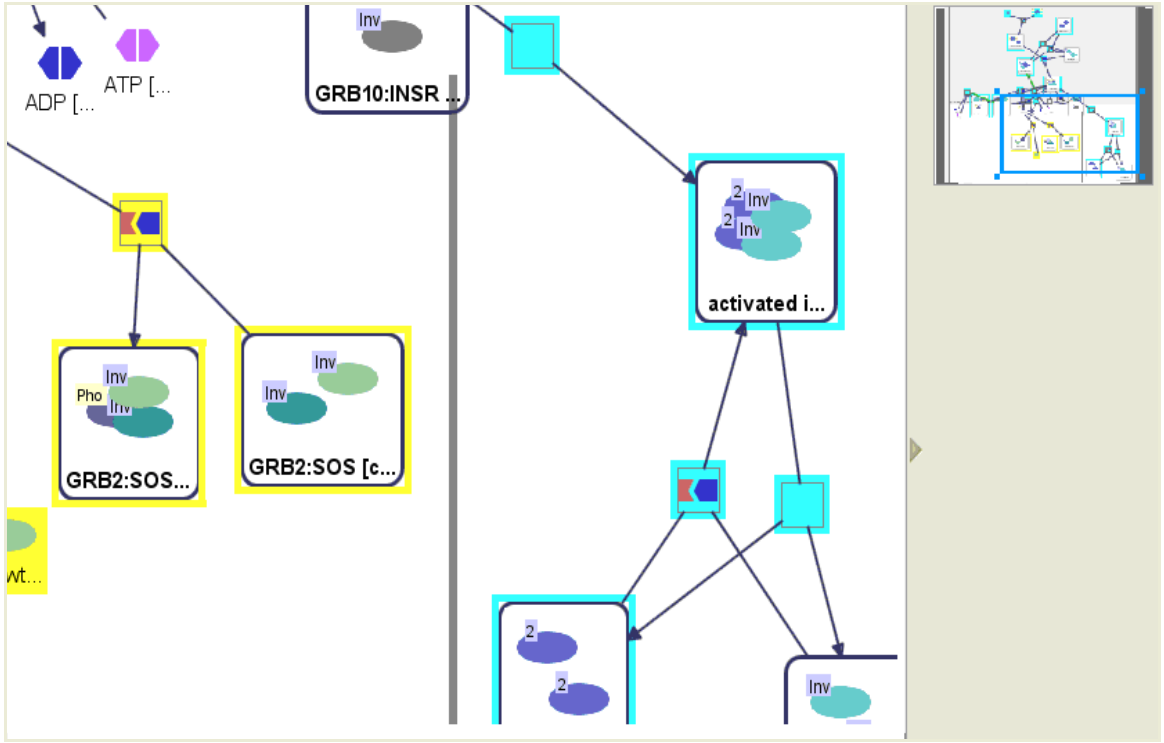


Figure 5-20 Highlighting clusters at mechanistic level; there are 2 clusters in this graph highlighted with 2 different colors.

If "Display clusters as abstractions" option is selected, than each cluster's elements will be grouped into a separate abstraction as shown in Figure 5-21.

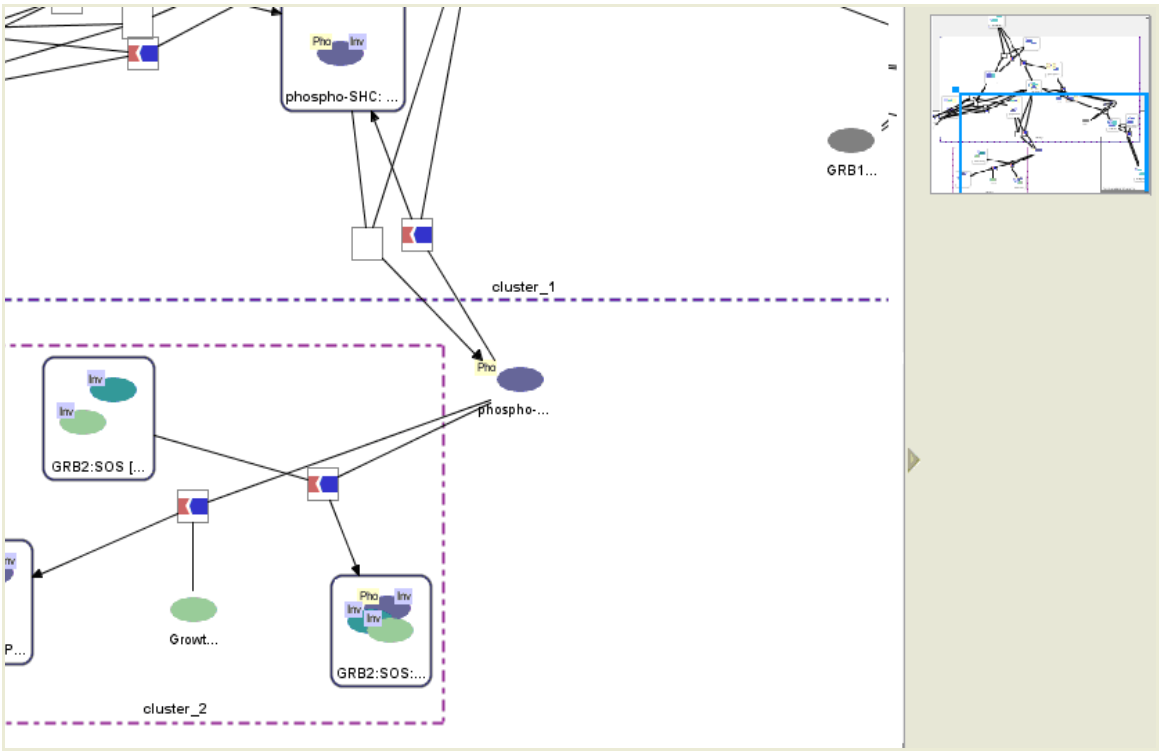


Figure 5-21 Visualizing clusters as abstractions at mechanistic level; in this view, there are 2 clusters represented with abstractions (bounded by *dashed* rectangles).

To simplify the view, user has the option not to retrieve interactions among the clusters. Then, the view will be as shown in Figure 5-22.

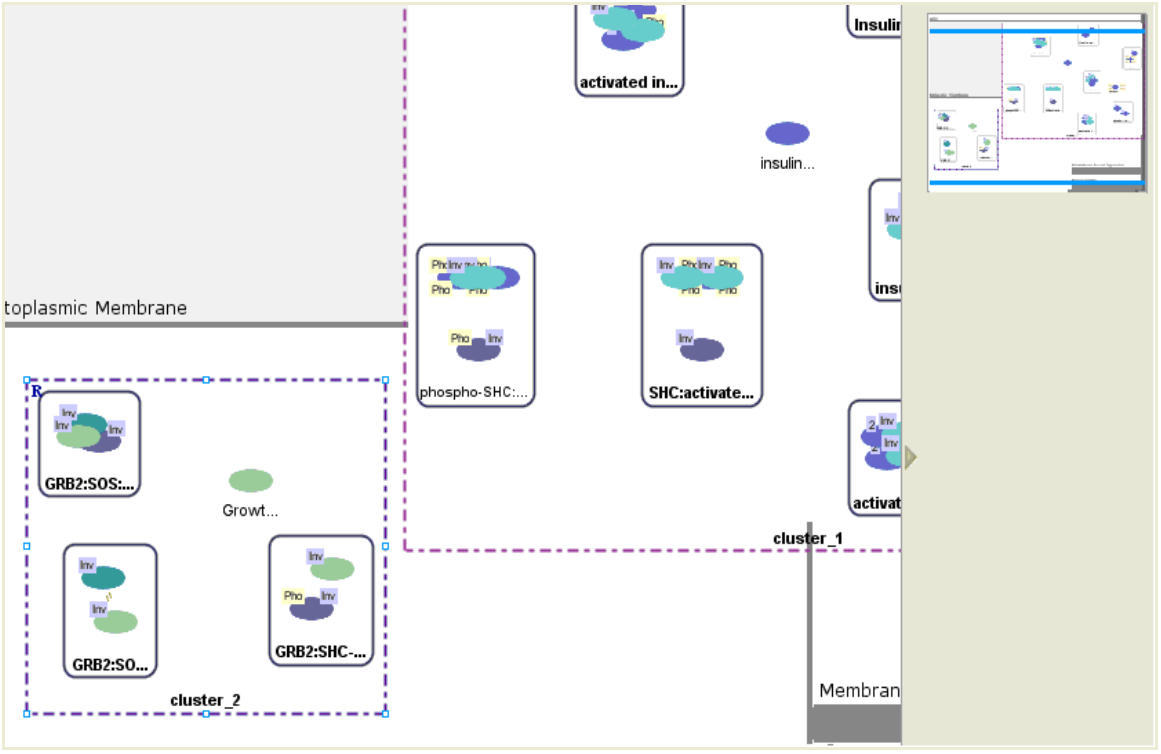


Figure 5-22 Visualizing clusters as abstractions without interactions at the mechanistic level

At the bioentity level, the user has the same options as mechanistic level but retrieval of interactions is not optional this time. Note also that, genes' protein products are also visualized.

Upon selection of 'Highlight Only' option, elements of each cluster will be highlighted with a different color as shown in Figure 5-23.

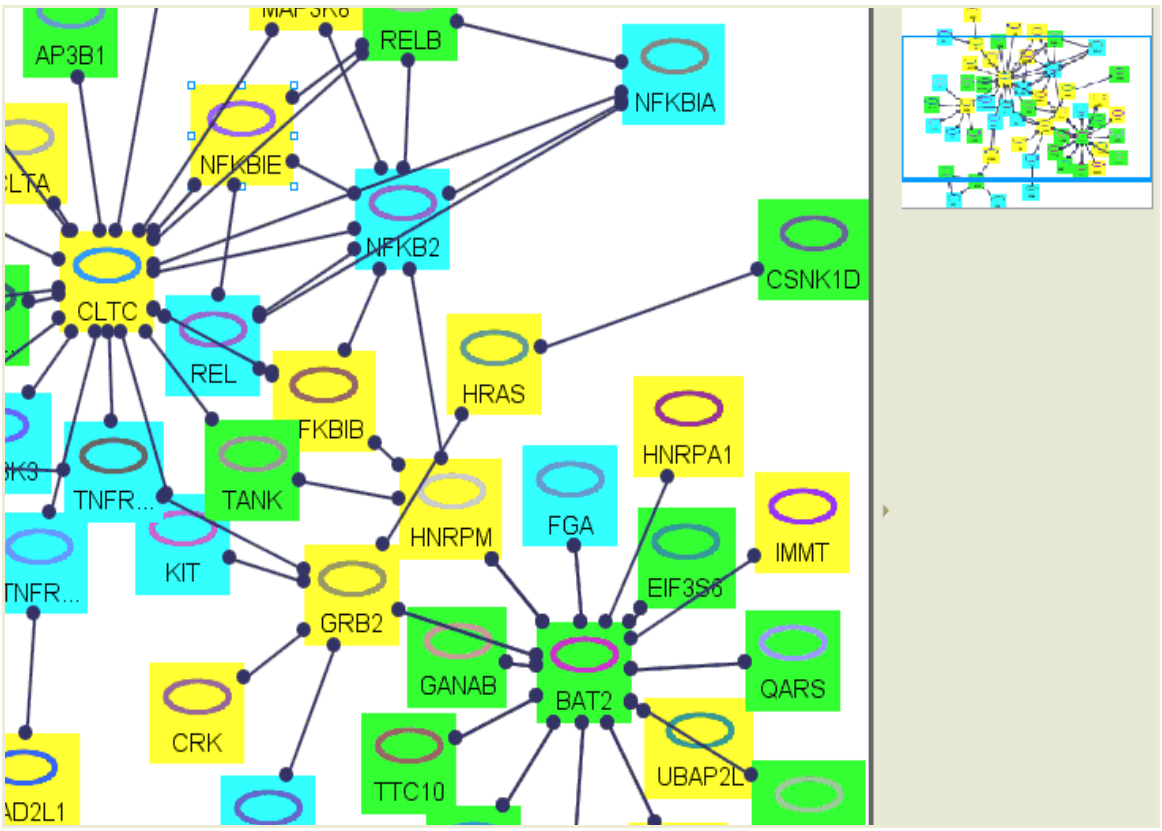


Figure 5-23 Highlighting clusters at bioentity level

If "Display clusters as abstractions" option is selected, then each cluster's elements will be grouped into a different abstraction as shown in Figure 5-24.

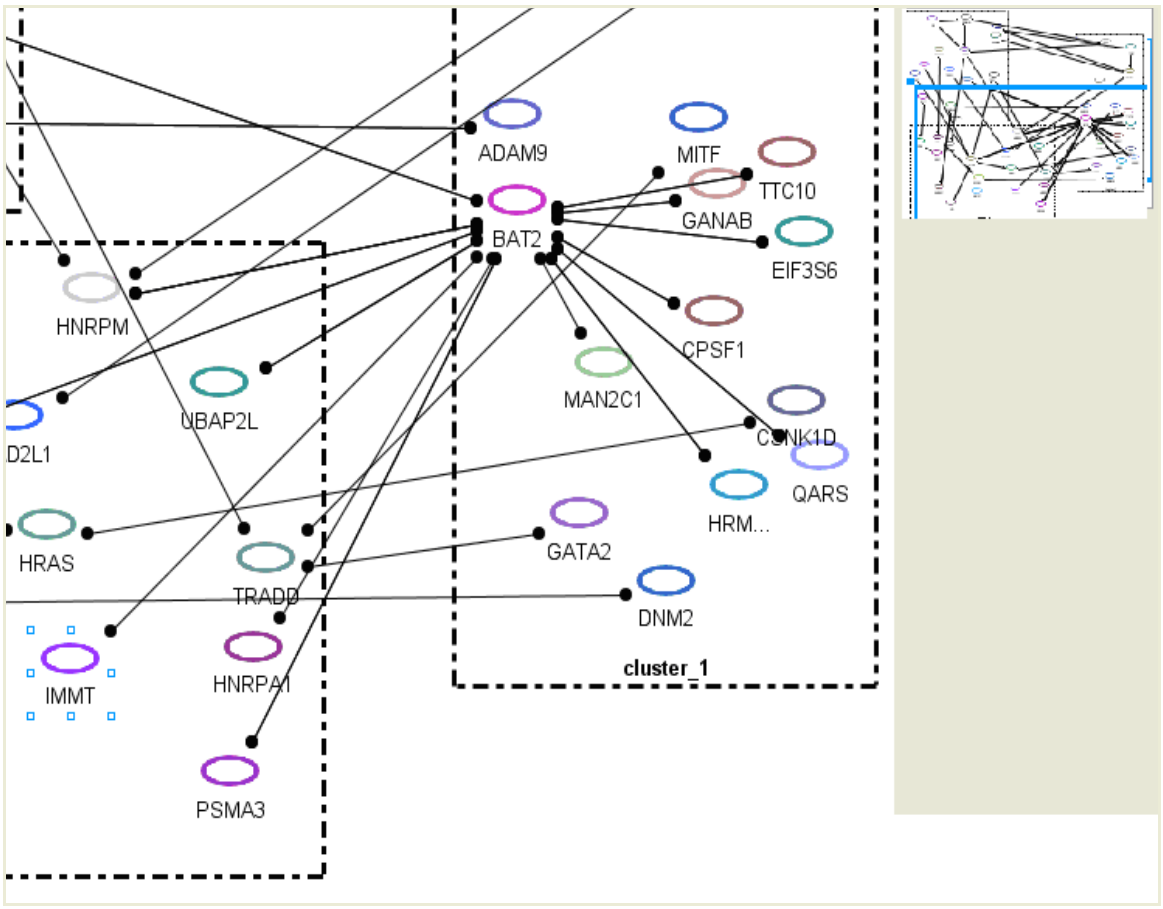


Figure 5-24 Visualizing clusters as abstractions at bioentity level

Chapter 6

6 PATIKA Ontology

This chapter discusses the PATIKA ontology, an improved ontology for cellular pathways.

We have described in [1] an ontology to model networks of cellular processes through integration of information on individual pathways. Our ontology is suitable for modeling incomplete information and abstractions of varying levels for complexity management. Furthermore, it facilitates concurrent modifications and extensions to existing data while maintaining its validity and consistency.

6.1 PATIKA Objects

Every first class object in the PATIKA ontology is a *PATIKA object*, which describes the common functionality and information. A PATIKA object has an *ID* that uniquely describes it with a *version*, an *author* (the user who first created this object), and a *data source*, describing how this phenomenon was observed and points to the literature references. A data source can be:

- Experimental
- Inferred
- Imported
- Other

Every PATIKA object also has a name and description, which comply with external naming conventions and vocabularies (such as Human Genome Nomenclature [3]) whenever possible. Finally every PATIKA object is optionally associated with a set of GO Terms [4].

6.2 Bioentities

More than often, actors, especially macromolecules have a common path of synthesis and/or are chemically very similar. For example, a p53 protein may be in native, phosphorylated, and MDM2-bound forms. Another example is cytoplasmic and extra-cellular calcium. These molecules usually have different information contexts. It is possible to model all these molecules as separate entities; however, it is not practical as these grouping are very natural and complies with the current biological paradigm. Moreover there is a wealth of information at this level of detail, thus we address entities at this level as well. Therefore it is more agreeable to maintain such biological or chemical groupings as *bioentities* while representing these 'minor' changes in their information context with *states*.

In most genomic and proteomic databases such as GeneCards [7], SwissProt [8] or GO [4], and in high-throughput data such as microarray and Y2H, bioentities form the unit entries. Each bioentity stores a set of external references mapping to these databases, and acts as a gateway to the external resources.

We hope to cover other entities like operons in the future versions of the ontology.

6.3 States

We model the actors of these events as states. The term is very generic and encapsulates macromolecules (e.g., DNAs, RNAs, and proteins), small molecules e.g., ions, ATP, and lipids), or even physical actors (e.g., heat, radiation, and mechanical stress). States also represent molecular complexes, or conceptual abstractions that behave like state. Depending on their nature, states are classified as either compound or simple.

Simple States

Simple states represent tangible and unit phenomena. They belong to a bioentity. Each state of a bioentity represents a change in the information context. Those changes are represented with the following bioentity variables:

- **Cellular Localization:** Each state has a compartment in the cell. Changes in compartment means a change in the molecules information context, since the set of molecules it can interact with changes. This property is single and mandatory for all states. Compartments are described in Section 6.6.
- **Attachment:** Apart from localization, a protein might have been attached to a membrane. Like cytosolic farnesyl attached proteins. Modeling attachment as a separate variable have several advantages over modeling each compartment-attachment pair as a different location, other than this, they are equivalent.
- **Complex Binding:** This property describes a functional change due to long-lived non-covalent bonds between molecules; for instance p53 bound state of MDM2. This property is multiple and optional.
- **Homomerization:** Non covalent bonding of two or more of the same state is not considered as a complex formation for the sake of simplicity. Instead we use a separate property for modeling such a state. This property is multiple and optional.
- **Isomerization:** Conformotional non-covalent changes within the molecule. This property is multiple and optional.
- **Chemical Modification:** Cleavages, group additions/removals and other covalent changes are classified in this group. This property is multiple and optional.
- **Aberration:** Two types of aberrations exist: sequence and structure aberrations (i.e. due to misfolding). Sequence aberrations can be chromosomal or point mutations as well as polymorphisms. We can define a more detailed sequence state variable annotation scheme using GO. Sequence aberrations can occur both on DNA, RNA and protein, although it must originate from a DNA. Structure aberrations can occur in RNA and Protein states and model changes due to aberrant folding proteins, physical factors or prions. Most of those aberrations are non-specific; that is, we are actually

grouping combinatorially many different molecules under a name, but it is ok since they do not exhibit any function. In the case of prions these aberrant forms can be functional, but since this is a very special and exceptional case, we will neglect it for now. This property is multiple and optional.

Any combination of bioentity variables forms a unique state of this bioentity. Note that only a very small portion of the state space actually occurs in biological systems.

An important side note is in pathway drawings, it is common represent different states as a single biological entity, even when the mechanistic detail is known. This is an oversimplification as different states can have very different and sometimes conflicting effects. Mapping such information to PATIKA graphs might not be trivial, as in most cases the mechanistic detail is unknown. PATIKA allows defining relations at both bioentity and state level to address these levels of detail and abstraction.

In some cases, a bioentity's states are also labeled with various logical (as opposed to physical) tags, such as active form, open channel or resting state. It is best to capture such tags with the states name, as they are context dependent.

Another point is states can be *ubiquitous*; that is, they participate in a significantly high number of reactions. Most of the time this is true for small molecules such as ATP, or water, which have generic and structural roles. For visualization and analysis, such states can be problematic. PATIKA allows labeling states as ubique, and handles them differently during visualization (e.g. splits up cytosolic ATP for each reaction it participates in) and query (e.g. ignores ATP during shortest path query).

States map to a class of molecules/entities rather than a single molecule. It might be that this group is not totally homogenous. For example it is not desirable for most of the cases to model the rotamers of a protein as different states, as there are combinatorially many of them, they are very short lived (in the range of nanoseconds) and switching from one of them to another is almost instantaneous. However PATIKA ontology does not define hard lines for the level of abstraction, as it readily provides a framework for modeling and representing multiple levels of detail. So we can say that state variables can be incomplete and overlapping. An example of incomplete state variable is "phosphorylated p53". However this representation poses a subtle problem. Since we do not know at which site the p53 is phosphorylated, relationship between this state and phosphrylated p53 at 153Arg is not clear. It might be that two authors actually talk about different states, or the latter is a non-proper subset of the first. A sensible approach is to delagate this issue to the submitter, and to the curator/expert, as it is really hard to come up with a context free resolution rule. If the first is the case, then the submitter must modify the phosphorylated p53 entry to bring it into the correct level of detail. On the other hand, if it is the second case they must convert the p53 phosphorylated into an incomplete state to indicate different levels of detail.

Obviously there are still a lot of important issues to cover, most important ones being combinatorial states, generics, polymerization, semi-quantitative phosphorylations. We are constantly evaluating examples (use-cases) from the biological literature to come up with improving our ontology to include these "hard" cases.

Compound States

A *compound state* is a grouping of other PATIKA objects, which exhibits a state-like behavior, and needs to be addressed at this level. There are two types of compound states: *complex* and *abstraction*.

In biological systems molecules often form clusters for performing proper tasks, behaving like a single state. We consider each member of a molecular complex as a new state of its biological entity. The function of a molecular complex is affected by the specific binding relations within itself. Therefore these binding relations must be represented in the model as well. Moreover, members of a molecular complex may independently participate in different transitions; thus one should be able to address each member individually. In addition, a molecular complex may contain members from multiple neighboring compartments. In that case, always one of those compartments is a member type compartment. It is actually possible to model complexes in a similar fashion to membrane spanning proteins.

Complex states have a set of simple state members named *complex members*.

Complex states do not have a bioentity, as they are not simple. However their members have their own bioentities. This information is used for complexes as well (e.g. for querying purposes).

An important question is "what is a complex really? Do we model short-lived binding relations as complexes or activation relationships?". PATIKA's answer is "never use a compound graph unless you need to", and this also applies to complexes. If an activation relation would do the trick, it is best to use it. If that level of detail is not sufficient for another user, they can re-edit it to add a complex at that point.

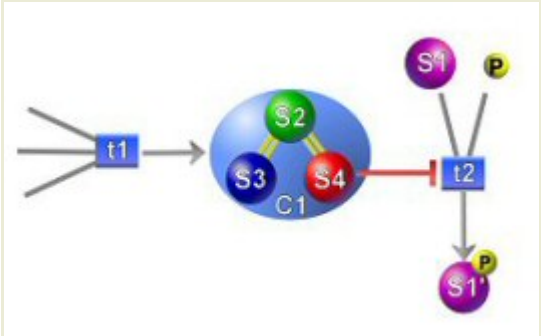


Figure 6-1 A portion of a pathway containing a molecular complex of three states

6.4 Transitions

A cell is not a static entity, neither are its actors. Molecules in a cell are synthesized, modified, transported and degraded constantly to respond to the changes the environment, or to accomplish a task. One can model such changes as quantitative chemical reactions. However this would reduce the coverage of the model, as currently both molecular concentrations and rate constants for most of these reactions are unknown. It is often preferred to represent these changes qualitatively since this better suits current experimental data.

A *transition* has a set of states as its *substrates* (inputs) and *products* (outputs). A transition occurs only when all of its substrates are present and activation conditions are satisfied; a function of the certain other states. These states are called the effectors of a transition. Two types of effector relations are defined, *activator* and *inhibitor*, for positive and negative regulation respectively. When a transition occurs, all of its products are generated. We take great care to make all PATIKA transitions compatible with the canonical biochemical paradigm.

PATIKA uses a pragmatic approach for formally defining transitions: any event that changes one or more states to another set of states is a transition. This definition delegates the exact definition of transition to the exact definition of state, and as mentioned above, level of modeling detail for PATIKA states are very flexible. It follows that PATIKA ontology can model transitions at multiple levels, allowing high coverage, without losing from its content. Two transitions are equal if they have the same set of substrates and products. This reveals two invariants for transitions:

- A transition has at least one substrate and one product.
- There cannot be two transitions with same set of substrates and products.

Although transitions can have a very large spectrum, we expect that most of them will fall to the certain classes. Those classes are captured by PATIKA transition tree. Under certain circumstances, multiple transitions having the same state as a substrate may affect each other through depleting this common substrate. This happens when the equilibrium constant of a transition is relatively much higher than the others. If such a difference occurs among the equilibrium constants of transitions, we call the transition with the highest equilibrium constant exhaustive over other transitions for the common substrate. Transitions having the same order of equilibrium constant, on the other hand, are said to be cooperative. Transitions that have one or more substrates are exhaustive on each other, through a depleting substrate. Which one of these substrates is likely to deplete is up to the modeler.

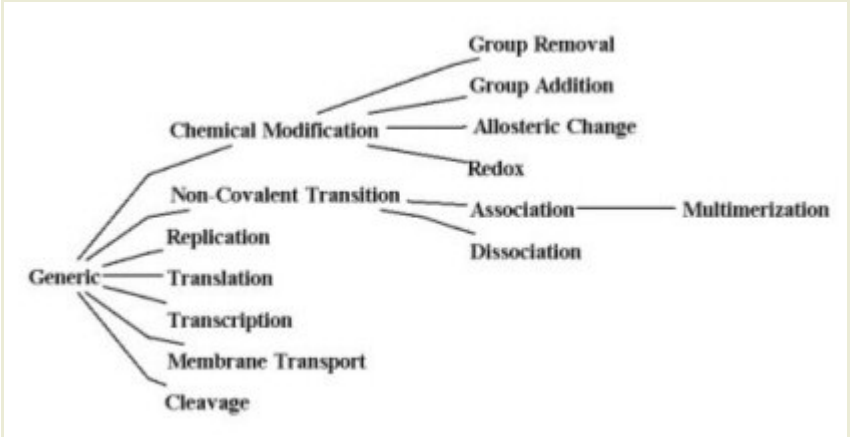


Figure 6-2 Transition type tree

Two transitions are called inverse of each other iff one transitions product set is other's substrate set and vice versa.

Following properties are missing from the current ontology, but were discussed at some point and left out for future versions.

Transition Rules

The term transition logic coins a rather wide spectrum. In modeling transition logic we can use Boolean predicates, linear equations, stochastic models, pi-calculus, etc. PATIKA ontology assumes that the representation and equality of the transition is independent of the transition logic. We assume that transition logic is represented in the transition rule, which is not an internal part of the ontology. Currently the only way of associating the transition rule with the transition is via custom user object.

Effector Combinations

Currently we assume that any combination of effectors can regulate a transition. This might not be the case; for example two inhibitors may never be present together in the cell, or when two inhibitors are present they cancel out each other. So, one point of view is to think of each effector combination as a separate transition. If it turns out that actually only a small number of all combinations of effectors are significant, a possible approach is to use the already existing compound graph notion to include children nodes into the transition for all significant combination sets, in order to be able to address them separately.

6.5 Interactions

Relations between bioentities, states and transitions are described using *interactions*, which can be directed or undirected. Interactions are divided into two based on their level of detail.

Mechanistic Interactions

Mechanistic interactions define relations between states and transitions at the chemical level of detail. There are five types of them:

- A **substrate** relation is a directed relation with a state as its source and a transition as its target, which indicates that the state is consumed by the transition. A substrate relation has a stochiometry attribute, which describes the number of its source states consumed per target transition. Stochiometry value defaults to 1. A substrate relation may also define an exhaustive property of its target transition over the source substrate. Exhaustiveness defaults to false.
- A **product** relation is a directed relation with a transition as its source and a state as its target. It indicates that the state is produced by the transition. It also has a stochiometry attribute to describe states produced per transition.
- An **activator** relation is a directed relation with a state as its source and a transition as its target. It describes the enabling or facilitating of the transition via the source state.
- An **inhibitor** relation is a directed relation with a state as its source and a transition as its target. It describes the disabling or impeding of the transition via the source state. Irreversible inhibitions should be modeled as a separate state of the modified enzyme.

- A **bind** relation is an undirected relation with two complex members as their source and target. It describes a non-covalent bonding between these two states. If all binding relations were known for a complex, than the graph defined by binding relations and members would be connected.

Bioentity Interactions

Bioentity interactions describe relations between bioentities but not states. They represent incomplete information, and always map to one or more mechanistic level interaction, although latter one might not be identified yet. There are six types of bioentity interactions:

- **PPI (Protein-Protein Interaction)** is a bi-directional relation, indicating that two proteins are observed to interact with each other in a Y2H or co-precipitation system. In other words, there is at least one state of entity A that somehow interacts with B. One or more mechanistic level relations might be associated with this entity level relation. For example state 1 of protein A might be bound by protein B, where state 2 of protein A might be bound and cleaved by B. Even the nature of the chemical reaction does not necessarily be similar. Compartment information and n-ary relations can not be captured by PPI. Some sample databases that contain PPI data include Incyte [9], DIP [10], BIND [11], IntAct [12], PIM [13], ProNet [14] and Mint [15].
- **TR (Transcription Relation)** is a uni-directional relation, indicating that at least one state of source node activates/inhibits expression of at least one DNA state of the target. Although there is combinatorial information on TR, we are yet to incorporate this to our ontology. Some sample databases that contain TR data include Transfac [16], RegulonDB [17] and ooTFD [18].
- **Co-clusters:** A bi-directional relation indicating that expression levels of RNAs of two bioentities are tightly coupled as measured by microarray experiments and clustering algorithms. This maps to a common transcription factor or a regulation path in the "big picture".
- **Literature:** A bi-directional or uni-directional relation indicating that two bioentities are referred significantly together in the literature; at some cases there can be inferred relationships like A activates B.
- **Inferred:** There are so called "pathway inference" algorithms that attempt to find bioentity level information mostly based on bioentity data. Inferred edges can be used to capture such data.
- **Derived:** These edges represent that there is a transition in the mechanistic graph that is adjacent to at least one state of each bioentity. Depending on the exact semantics this edge might have sub-types. For example a control edge might indicate that source bioentity has a state that is an effector of a transition, from which a state of the target is produced.

6.6 Compartments

A significant number of transitions transport molecules between *cellular compartments*. Transitions that a state can participate in are strictly related to its compartment; thus a change

in the compartment means a change in the state's information context. We choose to incorporate the state's compartment in the model.

As the compartments and their adjacencies are cell type dependent, compartmental structure should be modeled as part of the ontology.

Membranes pose an additional problem since not only a molecule may be located completely inside the membrane but also it may span one or both of its neighboring compartments. For membranes there are four types of sub-locations, two sides of the membrane, inside membrane and spanning membrane.

6.7 Abstractions

Network of molecular interactions derived from current biological data is incomplete and complicated. Complete network of cellular events is clearly beyond human perception. Different levels of abstractions are necessary to make effective analysis of cellular processes and dealing with complexity better.

Representing a cellular pathway as a single process or grouping related processes under a certain cellular mechanism would enhance the comprehensibility of the network of events (Figure 6-3). Such mappings are already present and may also be valuable for querying. We model such groupings using regular abstractions. Regular abstractions can be arbitrarily nested and can intersect. However they can not be addressed directly; that is, they have no incident edges.

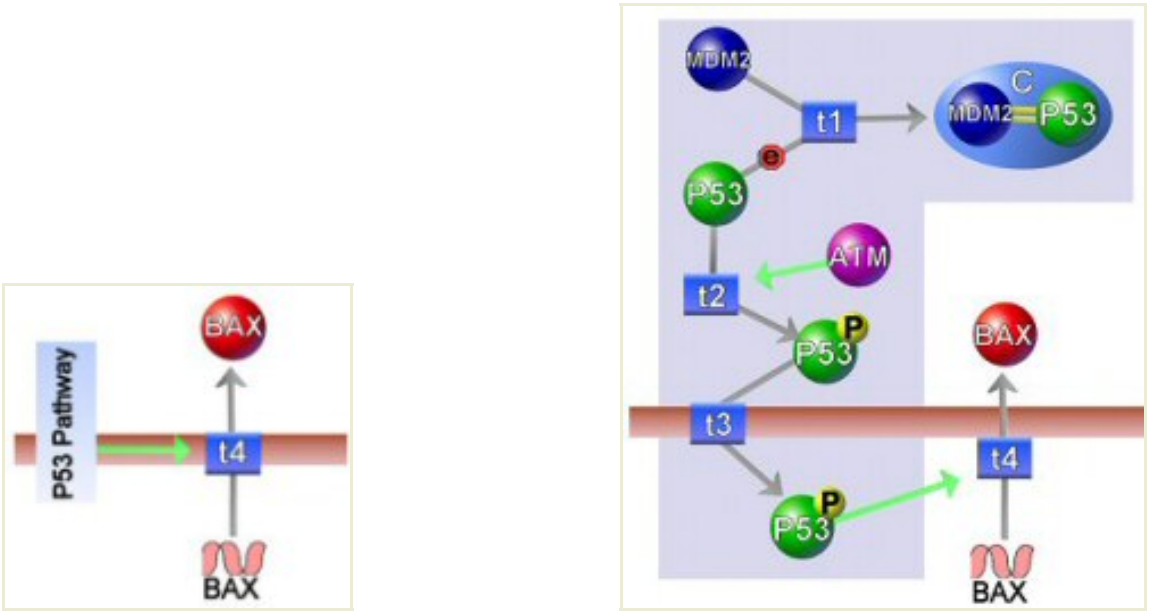


Figure 6-3 A cellular pathway is represented as a grouping

Since the data on cellular processes is not complete, different levels of information may be available for certain events. In cases where it is not identified which state among a set of states constitutes the substrate, product or effector of a transition, or where target transition of an effector is obscure, we may need to abstract these states (transitions) as a single state (transition) to represent the available information despite its incomplete nature. An edge defined

on an incomplete state means that it is actually defined on at least one state inside but the exact state is not known. A similar semantic applies to incomplete transitions.

In biological systems, a gene is often duplicated throughout its evolution serving a different function. A special case occurs when this differentiation serves as a specialization of a generic mechanism. For example when referring to the wnt gene, we actually mean nineteen various similar genes in human [5]. These genes are all activated by different stimulus at different tissues and can lead to different responses even though the signal processing mechanism is similar. Bhalla also describes common process motifs in signaling pathways, which are even more elementary operations that are reused through the entire network [6]. Our ontology supports representation of such homologies using abstractions.

Chapter

7

7 PATIKA Database

PATIKA*web* provides access to data from a number of popular pathway databases, and interfaces with major databases and ontologies. PATIKA Database integrates data from several sources, including Entrez Gene [19], UniProt [20], PubChem [21], GO [4], and Reactome [22]. The database focuses only on human pathway data and currently contains several thousands of states of many different biological entities and a few thousands of reactions. However stay tuned as we are currently working on integrating other protein and interaction databases, most notably BIND, IntAct, and HPRD.

Users can query and access this data using PATIKA*web*'s query interface (Chapter 4 on Querying the PATIKA Database), and can save their results in XML or export to common picture formats. Furthermore, our BioPAX and SBML exporters are provided as part of this Web service.

Release Notes

Here we summarize changes made to PATIKA*web* 2.0 since version 1.0:

- Support for microarray data analysis;
- Support for advanced graph-theoretic queries;
- Import facility from BioPAX level 2;
- Abstraction/nesting support at Biological entity level;
- Improvement to layout support.

References

[1] E. Demir, O. Babur, U. Dogrusoz, A. Gursoy, A. Ayaz, G. Gulesir, G. Nisanci and R. Cetin-Atalay (2004) "An Ontology for Collaborative Construction and Analysis of Cellular Pathways", Bioinformatics, 20(3), 349-356.

[2] U. Dogrusoz, E. Giral, A. Cetintas, A. Civril, and E. Demir (2004), "A Compound Graph Layout Algorithm for Biological Pathways", LNCS, vol. 3383, pp. 442-447.

[3] H.M. Wain, M.J. Lush, F. Ducluzeau, V.K. Khodiyar, S. Povey (2004) Genew: the Human Gene Nomenclature Database, 2004 updates. Nucleic Acids Res. 32 Database issue: D255-7. (PMID: 14681406)

[4] Gene Ontology: tool for the unification of biology. The Gene Ontology Consortium (2000) Nature Genet. 25: 25-29 (http://www.geneontology.org).

[5] J. Miller (2001) The Wnts. Genome Biol., 3, reviews 3001.1–3001.15.

[6] U. Bhalla (2002) The chemical organization of signaling interactions, Bioinformatics, 18, 855–863.

[7] GeneCards (http://bioinfo.weizmann.ac.il/cards)

[8] SwissProt (http://www.expasy.org/sprot)

[9] Incyte (http://www.incyte.com)

[10] DIP (http://dip.doe-mbi.ucla.edu)

[11] BIND (http://bind.ca)

[12] IntAct (http://www.ebi.ac.uk/intact/index.jsp)

[13] PIM (http://proteome.wayne.edu/PIMdb.html)

[14] ProNet (http://pronet.doubletwist.com)

[15] Mint (http://mint.bio.uniroma2.it/mint)

[16] Transfac (http://www.transfac.com)

[17] RegulonDB (http://www.cifn.unam.mx/Computational_Genomics/regulondb)

[18] ooTFD (http://www.ifti.org/ootfd)

[19] Entrez Gene (http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=gene)

[20] UniProt (http://www.uniprot.org)

[21] PubChem (http://pubchem.ncbi.nlm.nih.gov)

[22] Reactome (http://www.reactome.org)

[23] NCBI GEO (http://www.ncbi.nlm.nih.gov/geo)

[24] Stanford MicroArray Database (http://genome-www5.stanford.edu)

[25] Lee JS, Thorgeirsson SS. (2004) "Genome-scale profiling of gene expression in hepatocellular carcinoma: classification, survival prediction, and identification of therapeutic targets." , *Gastroenterology*, 127(5 Suppl 1): S51-5.

[26] Erik C. Gunther, David J. Stone, Robert W. Gerwien, Patricia Bento, and Melvyn P. Heyes. "Prediction of clinical drug efficacy by classification of drug-induced genomic expression profiles in vitro", *Proceedings of the National Academy of Sciences USA* (2003) , 100: 9608-9613

[27] Shannon W, Culverhouse R, Duncan J. (2003) "Analyzing microarray data using cluster analysis". *Pharmacogenomics*. 4(1), 41-51.

[28] Microarray Tutorial (http://www.ucl.ac.uk/oncology/MicroCore/HTML_resource/tut_frameset.htm)

[29] Distance Measures (http://www.ucl.ac.uk/oncology/MicroCore/HTML_resource/Distances_detailed_popup.htm)

[30] R Project (http://www.r-project.org/)

[31] Jain, A.K., Murty M.N., and Flynn P.J. (1999) "Data Clustering: A Review", *ACM Computing Surveys*, 31(3), 264-323.