Hyper Graph Module API

This document specifies and describes the API for a Generic and Extensible Hyper Graph Module.

# Graph API

### (MAKE-GRAPH &key <name><class>)

This creates and returns a graph object. If <name> is specified it should be a string and it will be retrievable via *find-graph*, otherwise the graph will have no name and will not be retrievable via *find-graph*.

### (FIND-GRAPH <name>)

Returns the graph object named <name> if such a graph exists.

### (DELETE-GRAPH <graph>)

Deletes the graph specified by <graph> if such a graph exists. The argument <graph> can be a graph object or a string naming a graph.

### (PRINT-GRAPH <graph>)

This function prints a list of the vertices, the edge-types and the edges of the specified graph object.

### (SAVE-GRAPH <graph> <mode> <destination>)

This function saves the graph specified by <graph> to a file using the adjacency list representation of a graph. Current ly the only supported <mode> is :lisp and <destination> is expected to be a file-name.

### (RESTORE-GRAPH <mode> <source>)

This function restores the graph described by <source> and returns an object of class *graph*. It is assumed that an adjacency list representation of the graph is contained in <source>. Current ly the only supported <mode> is :lisp and <source> is expected to be a file-name.

### (MAP-GRAPH-VERTICES <graph> <function>)

This function applies <function> to all vertices of <graph>. The argument <function> should be a function of one argument that will be invoked with a vertex object.

### (MAP-GRAPH-EDGES <graph> <function>)

This function applies <function> to all edges of <graph>. The argument <function> should be a function of one argument that will be invoked with an edge object.

# Graph Vertex API

### (MAKE-GRAPH-VERTEX <vertex-name> <graph> &key <class><value><weight>)

This function creates and returns a vertex object and adds it to specified graph in such a way that it can be retrieved by name from the specified graph. The argument <vertex-name> must be a string and the argument <graph> must be a graph object.

(VERTEX-NAME <vertex>)

(VERTEX-VALUE <vertex>)

(VERTEX-WEIGHT <vertex>)

(ADD-GRAPH-VERTEX <vertex> <graph>)

(DELETE-GRAPH-VERTEX <vertex> <graph>)

(FIND-GRAPH-VERTEX <vertex-name> <graph>)

(FIND-OUTBOUND-VERTICES <vertex> <edge-type><graph>)

(FIND-INBOUND-VERTICES <vertex> <edge-type><graph>)

# Graph Edge API

(EDGE-NAME <edge>)

(EDGE-VALUE <edge>)

(EDGE-WEIGHT <edge>)

(EDGE-SOURCE <edge>)

(EDGE-TARGET <edge>)

(MAKE-GRAPH-EDGE <edge-type><vertex-1><vertex-2><graph> &key <class><weight>)

(MAKE-DEFAULT-EDGE-NAME <edge>)

(ADD-GRAPH-EDGE <edge> <graph>)

(DELETE-GRAPH-EDGE <edge> <graph>)

(FIND-GRAPH-EDGE <source-vertex> <edge-type> <target-vertex> <graph>)

(FIND-GRAPH-EDGES <source-vertex> <target-vertex> <graph>)

(FIND-OUTBOUND-EDGES <source-vertex> <edge-type> <graph>)

(FIND-INBOUND-EDGES <target-vetex> <edge-type> <graph>)

(FIND-EDGES-OF-TYPE <edge-type> <graph>)

## Graph Edge Tables

%FIND-GRAPH-EDGE

FIND-EDGE-TABLE

DELETE-EDGE-TABLES

EMPTY-EDGE-TABLES-P