GRAPHREC

USER’S GUIDE

Version 1.0.0

Prepared by Petr Koupý <petr.koupy@gmail.com>

Last Updated on April 30, 2010

Table of Contents

1 Opening Input File

Input file contains graph definition, positions of nodes, colors of graph elements and some additional information. It is assumed that input file is generated by path planning software or some format converter. Since file format is human-readable, input file can be also prepared manually. Detailed specification can be found in Multirobot File Format section. File can be opened either by toolbar button or by **File – Open...** in the main menu. This action opens dialog containing two lists. List on the left side must be first filled with input files before proceeding further. To do that, just click on **Add files...** button and choose one or more input files in the standard open dialog. Note that you must choose appropriate file format, or the file will be parsed incorrectly. After the files are searched, left list contains one line for each graph found in the input file. Each line provides some additional information for given graph:

- **Name** – Each graph in the input file should be marked by its ID number. If ID is not specified, dash is displayed instead.
- **Line** – Line at which graph definition starts in the respective file. Can be useful information for manual revision of large input files.
- **Nodes** – Number of nodes in the graph.
- **Edges** – Number of edges in the graph.
- **Entities** – Number of entities that occupy some nodes. This number is always lower than number of nodes.
- **Timesteps** – Number of time steps each solution scenario consists of. One time step is defined by arbitrary number of non-conflicting entity movements through one edge.
- **Movements** – Number of entity movements in the whole solution scenario. Each movement is considered as move of one entity through one edge.
- **Validator** –Shows either default validator or the one preferred by input file. Selected validator will be used for initial validation. Validator can be changed by right clicking on the respective line. Note that validator can be also changed later in the application menu.

List can be sorted by each of these columns and irrelevant files can be folded in order to save space. Lines can be multi selected (by mouse dragging or by holding *Ctrl*/*Shift* and left mouse clicking) and then moved to the list on the right side (by **>>>** button or by double-clicking). Note that if you select header containing the file name, all graphs specified by this file will be moved to the right list. List on the right side serves as a basket for graphs you are willing to open. Graphs can be moved between two lists until you are satisfied with selection. To confirm dialog just click **Open** button and wait until selected graphs are loaded.

2 Environment Description

2.1 Main Menu

Menu consists of several sections:

- **File** menu contains actions related to file handling. See Opening Input File and Saving Output File sections for more information.
  - **Open...** shows advanced dialog for choosing input files.
  - **Save...** shows standard save dialog for saving single file.
  - **Save All...** shows standard save dialog for saving all opened tabs into single file.
  - **Exit** terminates the application.
- **Validation** menu contains actions related to solution validation. See Validating Solution section for more information.
  - **Validator** submenu shows all available validators. Currently selected validator is ticked.
  - **Log...** shows dialog for error logging.
- **Layout** menu contains actions related to graph layouting. See Layouting Graph section for more information.
  - **Mode** submenu shows list of layouting modes.
    - **Manual (continuous)** allows full manual control over node positions.
    - **Manual (discrete)** shows background grid that defines separation granularity.
    - **Automatic** enables selected layouter and its algorithm for layouting.

  - **Layouter** submenu shows all available layouters. Currently selected layouter is ticked.
  - **Recalculate** move all nodes to random positions and enables currently selected layouter. Available only in *Automatic* mode.
  - **Snap to Grid** aligns positions of all nodes to the closest valid place in the grid. Available only in *Manual (discrete)* mode.
- **Animation** menu contains actions related to animation control and capturing. Almost all actions in this menu appear in two versions – one for currently selected tab and one for all foreground (currently visible) tabs. Following list will describe only single tab actions. See Controlling Animation and Capturing Media Files sections for more information.
  - **Play** starts the animation that will be running from selected time step until stopped or until the last time step.
  - **Step** animates only currently selected time step and then stops.
  - **Stop** finishes currently animated time step and then stops animation.
  - **Seek...** shows dialog for time step selection. Note that this can be also done by controls in the tab (see Tabs).
  - **Reset** sets the scenario to the beginning.
  - **Synchronize All** enables or disables synchronization between tabs when playing animation on more of them at once.
  - **Snapshot...** shows dialog for capturing images.
  - **Sequence...** shows dialog for capturing image sequences or videos.
- **Setup** menu contains actions related to configuration options. See Settings and Coloring Graph Elements sections for more information.
  - **Colors...** shows dialog for coloring graph elements.
  - **Options...** shows main configuration dialog.
- **Window** menu contains actions related to graphical user interface. See Tabs and Window Splitting sections for more information.
  - **Split Horizontally** splits current tab group in a horizontal stack manner.
  - **Split Vertically** splits current tab group in a vertical stack manner.
  - **Unsplit** recursively joins two tab groups on the same level in the hierarchy.
  - **Unsplit All** removes all splits.
  - **Toolbar** button enables or disables visibility of the application tool bar.
  - **Controls** button enables or disables visibility of additional controls placed inside the tabs.
- **Help** menu contains actions related to documentation and application information.
  - **Documentation...** shows simple browser for this guide.
  - **About...** shows about box, which contains version, contact and licensing information for GraphRec.
  - **About Qt...** shows about box, which contains description and licensing information for Qt framework (used by GraphRec).
  - **About FFmpeg...** shows about box, which contains description and licensing information for FFmpeg video library (used by GraphRec).

2.2 Tool Bar

Application tool bar contains frequently used actions from main menu. These actions are (in order of appearance) **Open...**, **Save...**, **Manual (continuous)**, **Manual (discrete)**, **Automatic**, **Play**, **Step**, **Stop**, **Play All**, **Step All**, **Stop All**, **Snapshot...**, **Sequence...**. See Main Menu section for their description. Note that currently there is no possibility to add or remove actions from the tool bar. Tool bar can be enabled or disabled by **Window – Toolbar** in the main menu. It can be also moved over the screen or docked to arbitrary side of the application window.

2.3 Status Bar

Left side of the status bar contains additional information about current activity of the application. Messages can indicate idle state, file opening, file saving, validating, error detection, layouting, animating, capturing and image saving. Note that some of these messages can be displayed only for limited amount of time. Right side of the status bar shows description of currently selected validator. Note that validator should be also recognized by its specific default color set.

2.4 Tabs

Each opened graph is contained in its exclusive tab. Tab header contains close button and ID number of the graph. Context help of each tab contains name of its source file. Order of tabs in one tab group can be changed by mouse dragging. Note that it is not currently possible to drag and drop the tab from one tab group to another tab group. Tab groups are described in Window Splitting section. Each tab contains horizontal slider and spin box, both of which are connected and specifies current time step of the solution scenario. In fact, horizontal slider should be considered as a time line, similarly to audio or video players. There are also two vertical sliders on the left. The upper one is intended for adjusting the speed of the animation. The lower one specifies the overall displacement of nodes – the value that is used as a hint by the active layouter. Consequently, in order to use this feature interactively you must first enable **Automatic** layouting mode.

2.5 Window Splitting

To provide a way to work with more tabs at a time, it is possible to split initial set of tabs into more groups. Tab group must contain at least two tabs or it is not possible to split it. Splitting can be stacked horizontally (**Window – Split Horizontally**) or vertically (**Window – Split Vertically**). Before splitting, it is necessary to select tab that will be the first one in new group. Note that splitting is not possible if you select the first tab, because new tab group would be the same as the current one. After the group is split, you are free to do further sub splitting, which effectively results in a binary tree hierarchy. Window space for each tab group can be adjusted by dragging dividers between them. Note that when there is more than one tab visible, user interface still controls only single tab (the one that has focus). To switch focus between tabs just click anywhere into their window area. Active tab is always emphasized by the surrounding frame. There is also inverse operation to splitting. Tab groups can be rejoined by **Window – Unsplit**. Note that before unsplitting, you must first give focus to any tab in one of the two tab groups you want to join. Since any of these two tab groups can be already sub split, it is recursively joined to the bottom of the tree hierarchy. In order to rejoin all tab groups into single flat tab group just click on **Window – Unsplit All**. See also Tabs section for more information.

3 Controls

3.1 Mouse

This section provides description of mouse usage in GraphRec, which is not obvious at the first sight. This includes:

- **Moving** nodes in graph view by clicking on them and dragging them while holding left mouse button. Not working while animation is running.
- **Scrolling** of graph view by clicking into free space and dragging while holding left mouse button.
- **Zooming** graph view by mouse wheel.
- **Selecting** nodes by holding *Ctrl* key and clicking them or dragging selection frame over them while holding left mouse button. Note that if you want to move selected group of nodes, you must still hold the *Ctrl* key during operation otherwise nodes will be unselected.

3.2 Keyboard

- **Scrolling** graph view by arrow keys.
- **Moving** graph itself by *IKJL* keys (not allowed in **Automatic** layouting mode).
- **Rotating** graph itself by *AD* keys (not allowed in **Automatic** layouting mode).
- **Zooming** graph view by *WS* keys.
- **Random** positions for nodes by pressing *R* key.
- **Selecting** nodes by holding *Ctrl* key and clicking them or dragging selection frame over them while holding left mouse button. Note that if you want to move selected group of nodes, you must still hold the *Ctrl* key during operation otherwise nodes will be unselected.

4 Validating Solution

Each solution scenario is validated by selected validator during file loading. Validator checks whether all entity movements are valid and non-conflicting. All incorrect movements are logged and marked so the animation engine would ignore them. If there are any errors, the notification will appear in the status bar. GraphRec currently provides two validators:

- **Pebble** validator for *pebble motion on graph*. The movement is valid if and only if there is an edge between source and destination node, if there is an entity in the source node and if the destination node is free in the time step just before the movement.
- **Multirobot** validator for *multirobot path planning*. The movement is valid if and only if there is an edge between source and destination node, if there is an entity in the source node and if the destination node is already free or freed in the same time step as the movement occurs. Note that this definition permits chain movements of multiple entities.

Validator can be changed by **Validation – Validator** menu or by **Setup – Options...** (see Settings). Errors can be reviewed in special dialog accessible through **Validation – Log...** menu action. Whole log can be saved into text file. Each line in the error log specifies single problem. It consists of time step specification, movement details and error description. Movement details are logged in the format `SN(E1) ---> DN(E2)`, where `SN` stands for *source node ID*, `DN` for *destination node ID* and both `E1` and `E2` are ID numbers of entities that occupy respective nodes before the invalid movement. Note that entity IDs less or equal zero are reserved for empty nodes. Errors can be visually tracked down by clicking on log entries – corresponding time step is set and affected nodes are highlighted.

Note: If you find yourself studying grammar, input file or error log in detail, you should know that arrow displayed in each movement definition is not always indicating direction of movement. Movement, if valid, can occur also in the opposite direction than suggested by the arrow. This exception applies only to isolated movements allowed by **Pebble** and **Multirobot** validator. Chain movements are not affected. This inconveniency is caused by file format that is shared between GraphRec and planning software, which **Pebble** and **Multirobot** validator were written for.

5 Layouting Graph

Generally, initial positions of graph nodes are random. However, if the graph is biconnected and input file specifies its circles, nodes are aligned into lines that correspond to those circles. In most cases, the initial layout should be further modified. Usually, the first step to improve layout is to enable automatic layouter, which will iteratively modify the layout by its embedded algorithm. Layouter can be chosen from **Layout – Layouter** menu or by **Setup – Options...** (see Settings). To enable selected layouter, toggle **Layout – Mode – Automatic**. GraphRec currently provides two layouters:

- **Fruchterman-Reingold** layouter belongs to the group of force-directed layouters. Nodes behave as repulsive particles whereas edges behave like springs. Since main principle is based on physical laws, layouter is very good predictable and intuitive. Layout is elastic and can be influenced by dragging nodes during layouting. Layouter is especially suitable for graphs containing grid or some repetitive pattern.
- **Kamada-Kawai** layouter is also force-directed. One at a time, node is hooked by springs to the anchored rest and gradually moved to the equilibrium. Since nodes are not updated all at once, layouter is less predictable and intuitive. Layout is almost non-elastic and thus not very interactive. Generally, Kamada-Kawai is slower but provides slightly better layout, especially for non-grid graphs, because it defines ideal distances between all nodes (in comparison with Fruchterman-Reingold, where ideal distance is defined only between adjacent nodes).

Automatic layouters have usually more than one local minimum at which layout is considered done. If current local minimum is not good, it helps to move nodes to its desired positions while the layouter is still enabled. This action will usually results in finding another, probably better, local minimum. It is also possible to start again with random layout either by pressing *R* key or by clicking **Layout – Recalculate**. Another way to interact with layouter during its work is to change node displacement by the vertical slider on the lower left side of the graph view. Sliding the displacement up and down for a few times may remove little imperfections in the layout, especially in the case of grid graphs.

After the automatic layout is finished, it is recommended to switch to either **Layout – Mode – Manual (continuous)** or **Layout – Mode – Manual (discrete)** and to refine the layout further. At this point, graph will probably need some zooming, scrolling, rotating and node moving (all described in Controls). Whereas continuous mode gives no restrictions on node positioning, discrete mode allows only positions defined by its background grid. Granularity of background grid can be changed in **Setup – Options...** (see Settings). Discrete mode is intended for layouts that need exact orthogonality for esthetic purposes. Before aligning nodes manually to the grid, you may find useful **Layout – Snap to Grid**, which aligns all nodes to the closest valid position in the grid.

6 Settings

Configuration can be changed in the dialog accessible through **Setup – Options...** in the main menu. Note that settings are propagated only to the tab that has focus. If you want to change settings globally, hit the **Save Default** button, close/save all tabs and reload them. Explanation of configuration items follows:

- **Animation**
  - **Length** of single time step in milliseconds.
  - **Style** of the animation progress. *Linear* animation has same speed all the time whereas *EasyInOut* animation starts slowly, then goes steady and finish slowly.
- **Layout**
  - **Layouting mode** determines whether the layout will be done automatically or manually. See Layouting Graph for more information.
  - **Layouter** shows list of available layouters. See Layouting Graph for more information.
  - **Node displacement** serves as a hint for layouter. Should be used for exact value specification. For interactive alteration of node displacement, it is better to use vertical slider on the left side of the graph view.
  - **Grid offset** defines the granularity of background grid, which is enabled during discrete layouting mode.
- **Validation**
  - **Validator** shows list of available validators. See Validating Solution for more information.
- **Visual**
  - **Node description** determines what should be displayed on the surface of the nodes in graph view. Node labels can be either empty or can contain various combinations of entity and node identification numbers.
  - **Controls visibility** enables or disables sliders and spin box in graph view.
  - **Interactive capturing** determines whether user is able to interact (e.g. zooming, scrolling) with graph view during video or image sequence capturing. If disabled, capturing is indicated only by progress bar and graph view is not rendered in order to speed up video encoding.

7 Coloring Graph Elements

Graph colors can be adjusted in dialog displayed after clicking on **Setup – Colors...** menu entry. First, choose graph element type you would like to color from the tabbed panel on the left. Then select one or multiple items from the list. List items can be multi selected by mouse dragging or by holding *Ctrl*/*Shift* and left mouse clicking. To select all items hit the **Select All** button. Since dialog is not modal, it is also possible to switch back to main window and select graph nodes by holding *Ctrl* and left mouse button while dragging mouse over them – selection will be propagated into color dialog. List can be also sorted by any column, which makes it easier to select all items of the same color etc. After selecting desired set of items, click on **Set Color** button and choose what color you would like to change. Note that this can be also achieved by right mouse clicking anywhere into selected set of items. After clicking on appropriate color type, you can choose exact color in standard dialog for color picking. Both nodes and entities have its background and foreground color. Foreground color is used for labels displayed in nodes and thus should be in contrast with background color. Moreover, entities have another pair of colors used for their final state. Final state of entity is reached after its last movement in the solution scenario. This color differentiation is used to emphasize what entities will be moving in the future and what entities are already in their final destinations.

Special explanation should be given to the third tab with *Graph* label. List in the third tab is intended for colors that are more general:

- **Edge highlight** is a color used for wide line rendered under all edges with moving entities. If the color is bright and in contrast with other colors, it effectively improves global notion of entity movement and makes it easier to see how path-planning strategy works.
- **Graph background** is a color used for background of the whole graph view.
- **Outlines and edges** is a color used for all lines that represent edges and node boundaries. This color should be in very strong contrast with graph background.

8 Saving Output File

Current tab can be saved to file by clicking on **File – Save...** menu entry and choosing appropriate file format. Information that will be saved depend on chosen file format but should include graph definition, solution scenario, node positions, coloring, current view and chosen validator. If you want to save all opened tabs into single file at once just click on **File – Save All...** menu entry.

9 Controlling Animation

Actions for animation control are all accessible in **Animation** menu. Frequently used actions are also pinned to the tool bar. All actions are divided into two groups – first group is intended for controlling tab that has currently focus (e.g. **Play**) whereas the second group controls all foreground tabs (e.g. **Play All**). Second group is especially useful when there is a need to compare two similar scenarios – it is possible to play them or step through them while seeing them both at once.

Whole solution scenario consists of several time steps that can be played through (**Play**/**Play All**) or stepped through (**Step**/**Step All**). If you want to play animation from one particular time step, click on **Seek...**/**Seek All...** menu entry and specify it. Other possibility is to slide the horizontal slider on the top side of graph view or enter exact time step into the spin box located in the top left corner of graph view. Animation speed can be adjusted by the vertical slider on the upper left side of the graph view. Animation can be stopped any time by clicking on **Stop**/**Stop All** button. Just note that currently animated time step have to be finished so if the animation length was set too long, this might take a while. In order to rewind whole scenario back to the beginning, set time step to zero or click **Reset**/**Reset All** button.

Running more parallel animations by **Play All** can be done in two modes by toggling **Synchronize All**:

- **Synchronized** animation is suitable for tabs, which animation durations are equal. Synchronization ensures that tabs are timed collectively. Note that when synchronizing tabs with different animation duration, all tabs must wait for the slowest one on the every time step.
- **Non-synchronized** animation is suitable for tabs, which animation durations are different. In this mode, tabs are timed independently. This implies that two parallel animations with the same duration might get slightly desynchronized after a few time steps.

10 Capturing Media Files

GraphRec provides two ways to capture media files. There is a possibility to take raster and vector screenshots of graph view through **Animation – Snapshot...** dialog or to capture image sequences and video through **Animation – Sequence...** dialog. Dialog is almost the same for both actions. The only difference is that in the sequence mode it is possible to select video encoder and some additional settings. Dialog items are explained in the following list:

- **Encoder** shows a list of available media encoders. When the encoder is selected, its settings are shown on the right side of the dialog. Raster and vector encoders are

described in Images section whereas the only one video encoder, FFmpeg, is described in Video Standards and Video Settings sections.
- **Width** of resulting image or video (in pixels).
- **Height** of resulting image or video (in pixels).
- **Start** specifies the time step when the capturing begins. (available only in sequence mode)
- **Stop** specifies the time step when the capturing stops (resulting media will not include this time step). (available only in sequence mode)
- **Interval** determines amount of time steps that will be skipped between two captured images in sequence mode. For example if the interval is set to 3, only every fourth time step will be captured. (available only in sequence mode for non-video encoders)
- **Name** of the resulting media file. If location already contains file with such a name, the specified name will be extended with numeral and thus no data will be overwritten.
- **Path** to the destination directory for media files. Be sure there is enough free space when capturing video.

If all settings are set, capturing can be started by hitting **Capture** button. What happens next depends on settings. In the snapshot mode, image is simply saved and notification is displayed in status bar. Dialog is not closed and thus ready to take as many images as needed. Note that in snapshot mode, dialog is modal so you are free to interact with rest of the application. This is convenient for taking images during animation. It is suggested to increase animation length so it would be easier to take image at the intended moment.

Sequence mode behaves differently. Dialog is not modal so it is not possible to interact with rest of application. After the **Capture** button is clicked, dialog is closed and encoding starts from specified starting time step and runs until manually stopped or the stopping time step is reached. If capturing was set to be interactive in the Settings, it is possible to interact with graph view during encoding (e.g. zooming, scrolling). Note however that when capturing interactively, animation speed depends on speed of CPU and does not match real time. If the interactive capturing is disabled, graph view is not rendered in order to increase performance. In this case, only simple progress bar informs about status of the encoding job.

Note: When capturing sequence with image encoders, images are taken only between time steps. This means that if capturing consists of ten time steps, only ten images will be saved to destination directory. File names of these images will contain time step number.

Note: Resulting image or video frame does not necessarily match visible area of graph view in the time of capturing. This is due to selected resolution that, in most cases, has different aspect ratio than visible area of graph view. At first, capturing frame is aligned to the top left corner of graph view and then either its width or height is increased to match output ratio. This ensures that resulting image will certainly contain intended part of graph view and will have correct aspect ratio.

10.1 Images

GraphRec currently supports encoding to three raster formats (PNG, JPG, BMP) and one vector format (SVG). Raster formats are included in *Raster Image e*ncoder. The only possible setting for raster image is to adjust quality of the output. SVG format is provided by *SVG Generator* encoder. Pay attention to the settings of resolution and DPI, because inappropriate values may result in very thin or very wide strokes in resulting vector image. To avoid this you should increase resolution when increasing DPI and vice versa.

10.2 Video Standards

This section provides list of standard settings, which should be accepted by FFmpeg codecs. Codecs, such as H.263 or RV10, are especially strict on offered settings. On the contrary, MPEG codecs provide much more freedom in custom settings selection. Note also that visual output provided by GraphRec is less complex and more static in comparison with real life movies. Due to this fact, it is convenient to use lower bitrates than suggested ones in order to save additional space.

- **MPEG4**
    - **High Definition**, 1280x720x30fps, 1280x720x25fps, 8000 kbps

  - **Home Theater**, 720x480x30fps, 720x576x25fps, 4000 kbps
  - **Portable**, 640x480x30fps, 640x480x25fps, 352x240x30fps, 352x288x25fps, 768kbps
  - **Handheld**, 176x144x15fps, 200 kbps
- **MPEG2**
  - **HD 1080p**, 1920x1080x30fps, 1920x1080x25fps, 15000 kbps
  - **HD 720p**, 1280x720x30fps, 1280x720x25fps, 9000 kbps
  - **DVD**, 720x480x30fps, 720x576x25fps, 4000 kbps
  - **SVCD**, 480x480x30fps, 480x576x25fps, 2376 kbps
- **MPEG1**
  - **VCD**, 352x240x30fps, 352x288x25fps, 1150 kbps
- **RV10**
  - **Download**, 640x480x30fps, 1000 kbps
  - **Broadband**, 560x420x30fps, 650 kbps
  - **Midband**, 320x240x15fps, 300 kbps
  - **Narrowband**, 192x144x8fps, 100 kbps
- **FLV1**
  - **Best**, 320x240x25fps, 1200 kbps
  - **Normal**, 320x240x25fps, 960 kbps
  - **Optimal**, 352x288x15fps, 780 kbps
  - **Low**, 320x240x15fps, 640 kbps
  - **Least**, 176x144x15fps, 480 kbps
- **H.263**
  - **Extended**, 176x144x15fps, 240 kbps
  - **Highest**, 176x144x15fps, 104 kbps
  - **Standard**, 176x144x12fps, 64 kbps
  - **Balanced**, 128x96x10fps, 140 kbps
  - **Smallest**, 128x96x10fps, 44 kbps

10.3 Video Settings

Currently only two additional parameters are provided by user interface:

- **GOP (Group of Pictures)** – Distance between two *I-frames* (images containing full information). Note that standard MPEG1/2 is constrained by relatively small GOP sizes (18 for 30fps, 15 for 25fps). Other formats should be used with GOP set between 100 and 300 frames.
- **Buffer** – Size of the internal buffer, which is used by chosen codec to encode frames. Default size is 8MB, which should be enough for all reasonable settings. However, if you plan to encode video in extremely high resolutions or bitrates, size should be increased to avoid corrupted video.

Note: Advanced settings are currently hidden from user in order to keep things simple. These settings are set internally to provide best quality at reasonable encoding time and decoding performance. For MPEG4 in particular, this means, that GMC (*Global Motion Compensation*), AIC (*Advanced Intra Coding*) and MV4 (*four Motion Vectors by macro block*) are enabled. General settings for all codecs is also set to higher values (this includes *macro block decision algorithm*, *motion estimation comparison function*, *Trellis quantization*). Number of *B-frames* (bi-directional frames) is set to two frames per GOP for MPEG4 and MPEG2 (other codecs uses their default count).

11 Multirobot File Format

This part of guide is intended only for advanced users who are going to create format converters or edit input files manually. If you have programming skills, note that GraphRec can be extended to include more format parsers (implementation details are described in *Programmer's Documentation*). *Multirobot* file format is the initial format supported by GraphRec. It is derived from the output of *multirobot*, the software for solving *multirobot path planning* and *pebble motion on graph*, developed by RNDr. Pavel Surynek, Ph.D. (http://ktiml.mff.cuni.cz/~surynek/). Format specification consists of Grammar and Description of values that are relevant for GraphRec.

Format specification consists of grammar (in *Extended Backus–Naur Form* with case insensitive terminals) and its semantic description. Note that there are fields defined by grammar but irrelevant for GraphRec. When it happens that whole line is irrelevant, it is not even specified by the grammar and considered as an auxiliary non-terminal. Since auxiliary non-terminal has no exact specification, grammar should be used only as a reference due to its incompleteness. It can be assumed that line standing behind auxiliary non-terminal cannot interfere with the rest of the grammar. GraphRec puts also some optional extensions to the original format. These extensions include information about node positions, graph coloring, viewport state and validation mode. GraphRec can open either original or extended file but can save only extended file.

11.1 Grammar

```
file = { graph } , '<EOF>' ;
graph = { aux } , [ id ] ,
        { aux } , vertex block ,
        { aux } , edge block ,
        { aux } , [ circle block ] ,
        { aux } , [ validator block ] ,
        { aux } , [ color block ] ,
        { aux } , [ position block ] ,
        { aux } , solution block ,
        { aux } , length ,
        { aux } ;
id = 'id:' , uint , nl ;
vertex block = 'V =' , nl , { vertex }, nl ;
vertex = '(' , uint , ':' , uint , ')' ,
         '[' , sint , ':' , sint , ':' , sint '] ' ,
         { uintwh } , nl ;
edge block = 'E =' , nl , { edge }, nl ;
edge= '{' , uint , ',' , uint , '} (' , uint , ')' , nl ;
circle block = 'C =' , nl , { circle }, nl ;
circle = uintwh , '(' , uint , ',' , uint , '): ' , { uintwh } ,
         ' [' , { uintwh } , '] ' , '{' , { uintwh } , '}' , nl ;
validator block = 'MOD =' , nl , ( 'M:IMMEDIATE' | 'M:TRANSITIVE' ) , nl ;
color block = 'COL =' , nl , [ scene ] , [ borders ] ,
              [ highlight ] , { color } , nl ;
scene = 'B_SCN:A:' , color value , nl ;
borders = 'P_BRD:A:' , color value , nl ;
highlight = 'P_HLT:A:' , color value , nl ;
color = ( 'B' | 'P' ) , '_' , ( 'EMP' | 'INH' | 'FIN' ) ,
        ':' , uint , ':' , color value , nl ;
position block = 'POS =' , nl , [ matrix ] , [ angle ] ,
                 [ center ] , { position } , nl ;
matrix = ' MATRIX:' , float , ':' , float , ':' , float , ':' ,
         float , ':' , float , ':' , float , nl ;
angle = 'ANGLE:' , float , nl ;
center = 'CENTER:X' , float , ':Y' , float , nl ;
position = uint , ':X' , float , ':Y' , float , nl ;
solution block = 'Solution' , nl , { move } , nl ;
move = uintwh , '---> ' , uintwh , [ move extension ] ,
       '(' , uint , ')' , nl ;
move extension = '[' , sint , ' ---> ' , sint , '] ' ;
length = 'Length:' , uint , nl ;
aux = ? unknown additional information ? , nl ;
nl = new line , { new line } ;
new line = '<LF>' | '<CR>' | '<LF>' , '<CR>' | '<CR>' , '<LF>' ;
numeral = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' ;
uint = numeral , { numeral } ;
uintwh = uint , ' ' ;
sint = [ '+' ] , uint | '-' , uint ;
float = sint , '.' , uint | sint ;
hnumeral = numeral | 'a' | 'b' | 'c' | 'd' | 'e' | 'f' ;
color value = '#' , hnumeral , hnumeral , hnumeral , hnumeral ,
              hnumeral , hnumeral ;
```

11.2 Description

- `(<node_id>:<IGNORED>)[<initial_entity_id>:<IGNORED>:<IGNORED>]` stands for vertex definition. Note that entity identifications less or equal zero are reserved for empty nodes.
- `{<source_node_id>,<destination_node_id>} (<IGNORED>)` stands for edge definition.
- `<circle_id> (<source_node_id>,<destination_node_id>): <whole_circle> [<new_arc>] {<existing_arc>}` stands for circle definition. Last three tokens are lists of node identifications. Circle is created by joining new arc to the source and destination node, both of which are joints of existing arc.
- `M:IMMEDIATE` selects validator for *Pebble motion on graph*.
- `M:TRANSITIVE` selects validator for *Multirobot path planning*.
- `B_SCN:A:<color>` is background color for graph view scene. Letter *A* stands for *all*, but has no practical meaning in this context.
- `P_BRD:A:<color>` is color for edges and node borders. Letter *A* stands for *all*, but has no practical meaning in this context.
- `P_HLT:A:` is color for edge highlighting. Letter *A* stands for *all*, but has no practical meaning in this context.
- `B_EMP:<node_id>:<color>` is background color for empty node.
- `P_EMP:<node_id>:<color>` is foreground color for empty node.
- `B_INH:<entity_id>:<color>` is background color for entity in non-final position. *INH* stands for *inhabited*.
- `P_INH:<entity_id>:<color>` is foreground color for entity in non-final position. *INH* stands for *inhabited*.
- `B_FIN:<entity_id>:<color>` is background color for entity in final position. *FIN* stands for *final*.
- `P_FIN:<entity_id>:<color>` is foreground color for entity in final position. *FIN* stands for *final*.
- `MATRIX:<m11>:<m12>:<m21>:<m22>:<dx>:<dy>` is transformation matrix of the scene.
- `ANGLE:<angle>` is rotation of the scene (in degrees). Currently **obsolete**, because scene is no longer rotated – rotation is saved in node positions instead.
- `CENTER:X<x>:Y<y>` is point in the scene that is aligned to the center of viewport.
- `<node_id>:X<x>:Y<y>` is position of specified node.
- `<source_node_id> ---> <destination_node_id> [<IGNORED> ---> <IGNORED>] (<time_step>)` defines one particular move of unspecified entity between two specified nodes at specified time step. Direction of arrow does not necessarily match direction of move – if suggested direction is evaluated as invalid by validator, there is a chance that the inverse direction would be evaluated as valid. Note that these lines might not be sorted by time step in the input file.

11.3 Example

```
id:1
V =
(1:0)[1:0:0]
(2:0)[2:0:0]
(3:0)[3:0:0]
E =
{1,2} (0)
{2,3} (0)
{3,1} (0)
MOD =
M:TRANSITIVE
COL =
B_SCN:A:#ffffff
P_BRD:A:#000000
P_HLT:A:#00ffff
B_EMP:1:#ffaa00
B_EMP:2:#ffaa00
B_EMP:3:#ffaa00
P_EMP:1:#000000
P_EMP:2:#000000
P_EMP:3:#000000
B_INH:1:#0000ff
```

```
B_INH:2:#ff0000
B_INH:3:#00ff00
P_INH:1:#ffffff
P_INH:2:#ffffff
P_INH:3:#ffffff
B_FIN:1:#00007f
B_FIN:2:#aa0000
B_FIN:3:#005500
P_FIN:1:#ffffff
P_FIN:2:#ffffff
P_FIN:3:#ffffff
POS =
MATRIX:1.68179:0:0:1.68179:0:0
ANGLE:0
CENTER:X-7.13524:Y-35.6762
1:X-18.7568:Y-8.08399
2:X4.0907:Y-71.4452
3:X-62.2215:Y-59.842
Solution
2 ---> 3 [0 ---> 0] (0)
3 ---> 1 [0 ---> 0] (0)
1 ---> 2 [0 ---> 0] (0)
3 ---> 1 [0 ---> 0] (1)
2 ---> 3 [0 ---> 0] (1)
1 ---> 2 [0 ---> 0] (1)
3 ---> 1 [0 ---> 0] (2)
2 ---> 3 [0 ---> 0] (2)
1 ---> 2 [0 ---> 0] (2)
Length:9
```

12 GraphRec File Format

This part of guide is intended only for advanced users who are going to create format converters or edit input files manually. If you have programming skills, note that GraphRec can be extended to include more format parsers (implementation details are described in *Programmer's Documentation*). *GraphRec* file format is refined alternative to the *Multirobot* file format, which is burdened by compatibility redundancies. Format is specified in XML and is designed against requirement to provide better locality and encapsulation of information than *Multirobot* format.

12.1 Description

While reading this section, it is recommended to refer to the example below. Note that optional tags or attributes are enclosed in square brackets. File is composed from XML header, document type `<!DOCTYPE graphrec>` and single root element `<graphrec version="<version_number>"></graphrec>`. Root element acts as a container for one or more `<solution [id="<solution_id>"]></solution>` definitions, which are further composed from following elements:

- `[<scene [bg="<background_color>"]></scene>]` contains scene definition:
    - `[<viewport x="<x_position>" y="<y_position>"/>]` specifies point in the scene that is aligned to the center of viewport.
    - `[<matrix m11="<m11>" m12="<m12>" m21="<m21>" m22="<m22>" dx="<dx>" dy="<dy>"/>]` defines transformation matrix of the scene.
- `<graph></graph>` contains graph definition:
    - `<entity id="<entity_id>" [bg="<background_color>"] [bgf="<final_background_color>"] [fg="<foreground_color>"] [fgf="<final_foreground_color>"]/>` stands for entity definition. Note that entity identification must be greater than zero.
    - `<node id="<node_id>" [ent="<initial_entity_id>"] [x="<x_position>" y="<y_position>"] [bg="<background_color>"] [fg="<foreground_color>"] [bnd="<boundary_color>"]/>` stands for vertex definition. Note that entity identification must be greater than zero.
    - `<edge n1="<first_node_id>" n2="<second_node_id>" [ln="<line_color>"] [hgl="<highlight_color>"]/>` stands for edge definition.

- `<scenario [validator="<validator_name>"]></scenario>` contains all movements off the solution:
    - `<move tms="<time_step>" src="<source_node_id>" dst="<destination_node_id>"/>` defines one particular move of unspecified entity between two specified nodes at specified time step. Direction implied by node atrributes does not necessarily match direction of move – if suggested direction is evaluated as invalid by validator, there is a chance that the inverse direction would be evaluated as valid. Note that move definitions might not be sorted by time step in the input file.

12.2 Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE graphrec>
<graphrec version="1.0">
    <solution id="1">
        <scene bg="#ffffff">
            <viewport x="-1.18921" y="-29.7302"/>
            <matrix m11="1.68179" m12="0" m21="0"
                   m22="1.68179" dx="0" dy="0"/>
        </scene>
        <graph>
            <entity id="1" bg="#0000ff" bgf="#00007f"
                          fg="#ffffff" fgf="#ffffff"/>
            <entity id="2" bg="#ff0000" bgf="#aa0000"
                          fg="#ffffff" fgf="#ffffff"/>
            <entity id="3" bg="#00ff00" bgf="#005500"
                          fg="#ffffff" fgf="#ffffff"/>
            <node id="1" ent="1" x="-18.7568" y="-8.08399"
                          bg="#ffaa00" fg="#000000" bnd="#000000"/>
            <node id="2" ent="2" x="4.0907" y="-71.4452"
                          bg="#ffaa00" fg="#000000" bnd="#000000"/>
            <node id="3" ent="3" x="-62.2215" y="-59.842"
                           bg="#ffaa00" fg="#000000" bnd="#000000"/>
            <edge n1="1" n2="2" ln="#000000" hgl="#00ffff"/>
            <edge n1="2" n2="3" ln="#000000" hgl="#00ffff"/>
            <edge n1="3" n2="1" ln="#000000" hgl="#00ffff"/>
        </graph>
        <scenario validator="Multirobot">
            <move tms="0" src="3" dst="1"/>
            <move tms="0" src="1" dst="2"/>
            <move tms="0" src="2" dst="3"/>
            <move tms="1" src="1" dst="2"/>
            <move tms="1" src="2" dst="3"/>
            <move tms="1" src="3" dst="1"/>
            <move tms="2" src="1" dst="2"/>
            <move tms="2" src="2" dst="3"/>
            <move tms="2" src="3" dst="1"/>
        </scenario>
    </solution>
</graphrec>
```