AsapBMLFlowVisualisation

Jan Pöppel

jpoeppel@techfak.uni-bielefeld.de

January 3, 2015

Quickstart

From **AsapRealizerDemo/java/resource/** run
go.bmlflowvisualizer

Spread needs to be running for the BMLFlowVisualizer to work. Basic information about the program and it's functionality can be found under the Info menu (Menu → Info).

Contents

1 Overview

The BMLFlowVisualizer is a tool designed to visualise the bml flow in a running Asap environment. It allows the user to see what kind of bml blocks are send over ipaaca at what time. The bml blocks can also be inspected to view their messages. Furthermore the user gets information about the current scheduling state of already received bml blocks.

2 Installation

The BMLFlowVisualizer comes with a *build.xml* allowing the use of *ant*. This has so far however only been tested within the working environment of the Sociable Agents Group of the university of Bielefeld and some commands might not work in different environments.

3 Functionality

This tool is supposed to make it easier to understand and debug the bml flow in an Asap environment. To that end, the bml blocks are shown in their temporal context as well as in the context of their connections (see section 4.1 for details). More detailed information about a certain bml block can be browsed by opening the BML Information Window (see section 4.3) by double clicking a block in any of the visualisations. Furthermore it is possible to search for a specific block using the search window (4.2). A recorded session can also be stored and loaded at a later time using the *Save* and *Load* options, provided by the menu or their respective shortcuts $CTRL+S$ and $CTRL+L$. A short summary about the different colours (see section 4.4 for more details) and the basic functionality of the tool can be viewed using the Information Window, accessible using the menu button *Info* or the shortcut $CTRL+I$.

4 Details

This section gives more detailed information about the different windows and the meaning behind the block colours.

4.1 Main Window

Figure 1 shows the main window of the BMLFlowVisualizer. The left half of the window visualises the history of the current Asap session, while the right half visualises more detailed information about the current time. The red line on the left (as well as the number in the textbox below) indicates the current time, which detailed visualisation can be seen on the right half.

The history section is basically separated into 3 subsections. The top subsection visualises the blocks in their planning phase (submitted and in preparation). The second subsection visualises the blocks in their scheduling phase (pending and lurking). The last subsection shows the blocks in their execution phase (e.g. in_exec and done). All blocks should go through all three phases and be represented in each of them. The width of each

block visualises the time the block has spend in its three phases (planning, scheduling and execution).

The detailed section consist of two panels. The top panel visualises the bml blocks that are currently being planned, while the lower one visualises the blocks that are currently being scheduled and playing. It also shows their relative connection to each other. A dotted line between two blocks means that they have an „append“connection while a complete line means that they have a „chunk“connection. In the detailed section the block width has no special meaning.

The main window also provides a few control options: The first from the left in figure 1 allows to set the current time. The time can be set using the textbox and the button *Jump to time*.

The slider allows to zoom in and out of the history panels.

The *Play* button „plays“back the recorded session starting from the current time all the way to the end. When playing the button will toggle to *Pause* which stops the playback.

The last button *Go to End* moves the current time to the last recorded timestep.

The checkbox *Update on feedback only* controls the way the bml flow is recorded. The default setting starts recording when the first bml message is received and stops when no new message was received for at least fife seconds. During this time the visualisation is updated every 100ms. When checked the visualisation will only be updated when a new bml message is received, resulting in a lower consumption of resources at the cost of jumping visual updates.

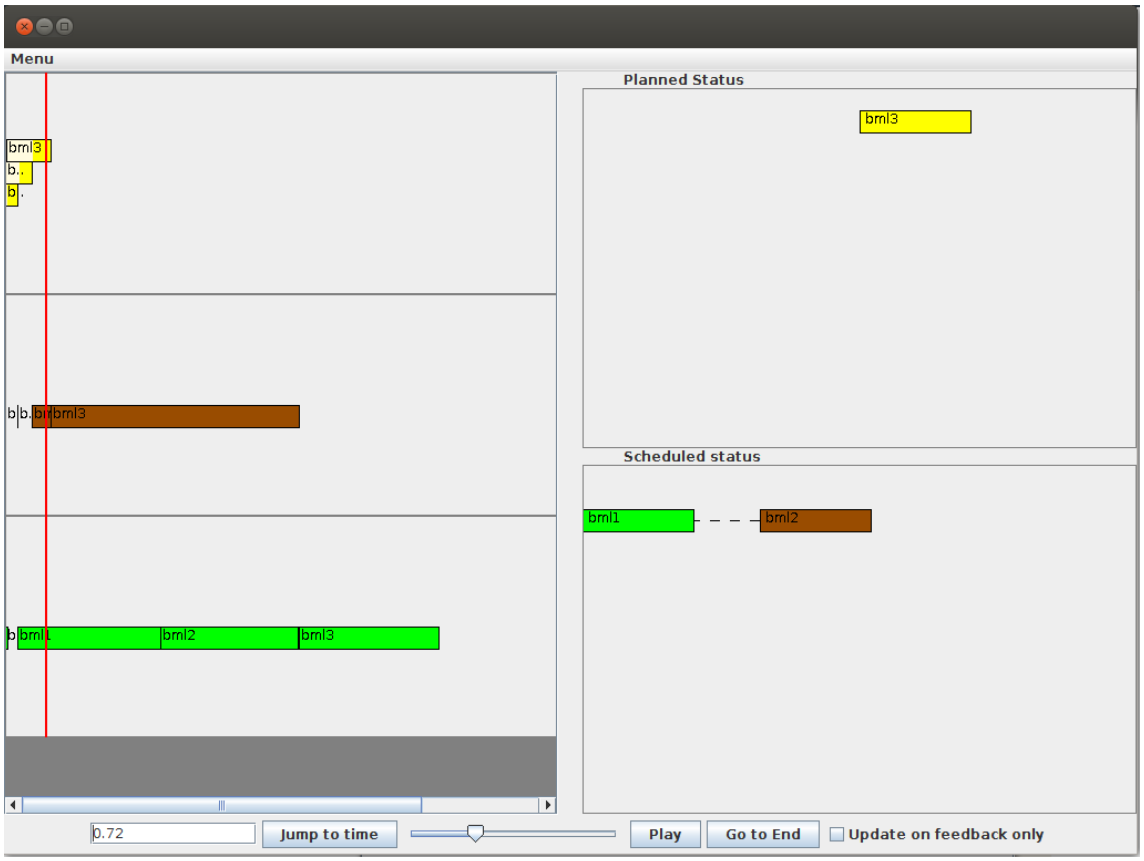

Figure 1: Main window of the BMLFlowVisualizer. The left half visualises the time history of the current session, while the right half shows more detailed information about the current timestep. The red line on the left symbolises the current time.

4.2 Search Window

Figure 2 shows the search window that can be accessed using the menu or the shortcut $CTRL + F$. It allows to find a specific bml block since it lists all received blocks in a sortable table. The table can be sorted by any column by clicking at the column headers. A double click on any block row opens the block information window (see section 4.3) about that block.

Furthermore the textfield at the bottom allows to search for a specific bml block by it's id. The id does not need to be known completely since the search will toggle through all bml blocks that have a matching prefix when the *Search* button is pressed. The row of the currently found bml block is highlighted as can be seen in the figure.

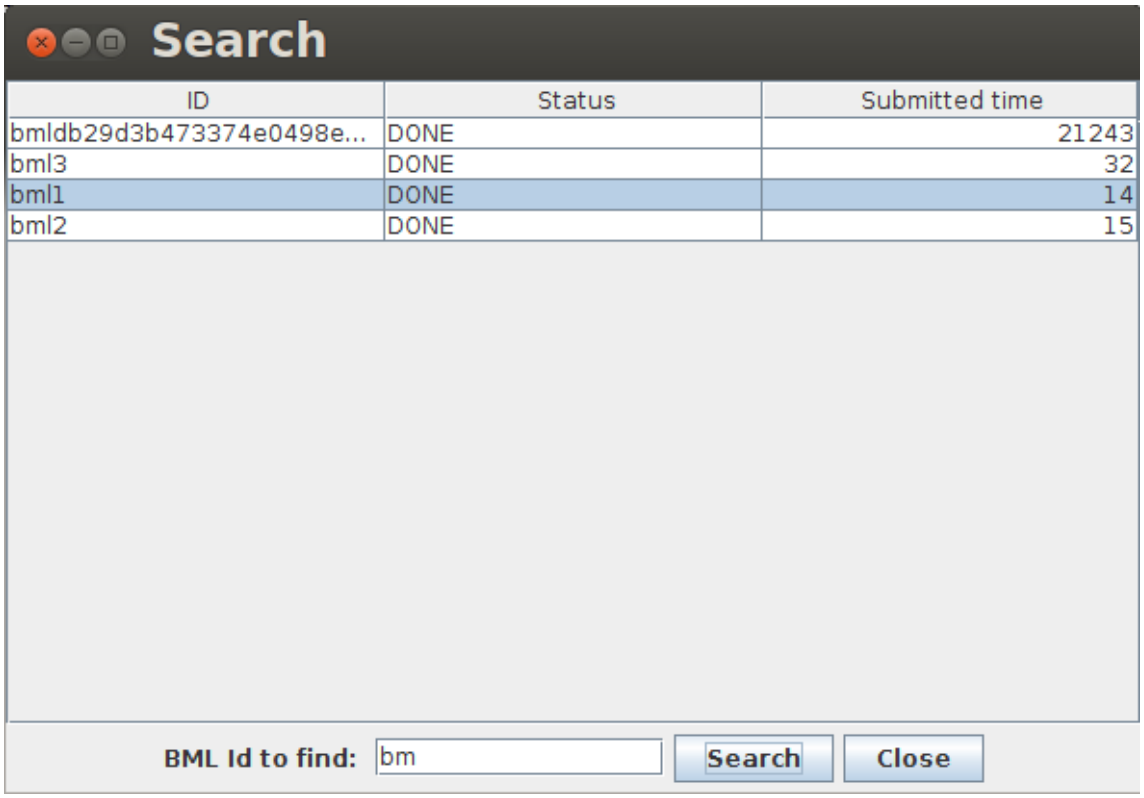

Figure 2: Search window of the BMLFlowVisualizer. Shows all received bml blocks with their current status and the time of their submission.

4.3 BML Information Window

Figure 3 shows the information window for the block with the id bml1. The three columns show all information about the block. The first column shows the behaviours this block contains. The middle column lists all the messages the were received related to this bml block. A double click on any of the messages opens up a popup showing the actual xml of the message as can be seen in the second window in figure 3. The last column shows the states the block has been in so far. Highlighted in yellow is the current state and the last received message (according to the current time selected in the main window). By selecting either a message or a state and using the *Jump to selected time* buttons will set the current time in the main window to the time of the selected field.

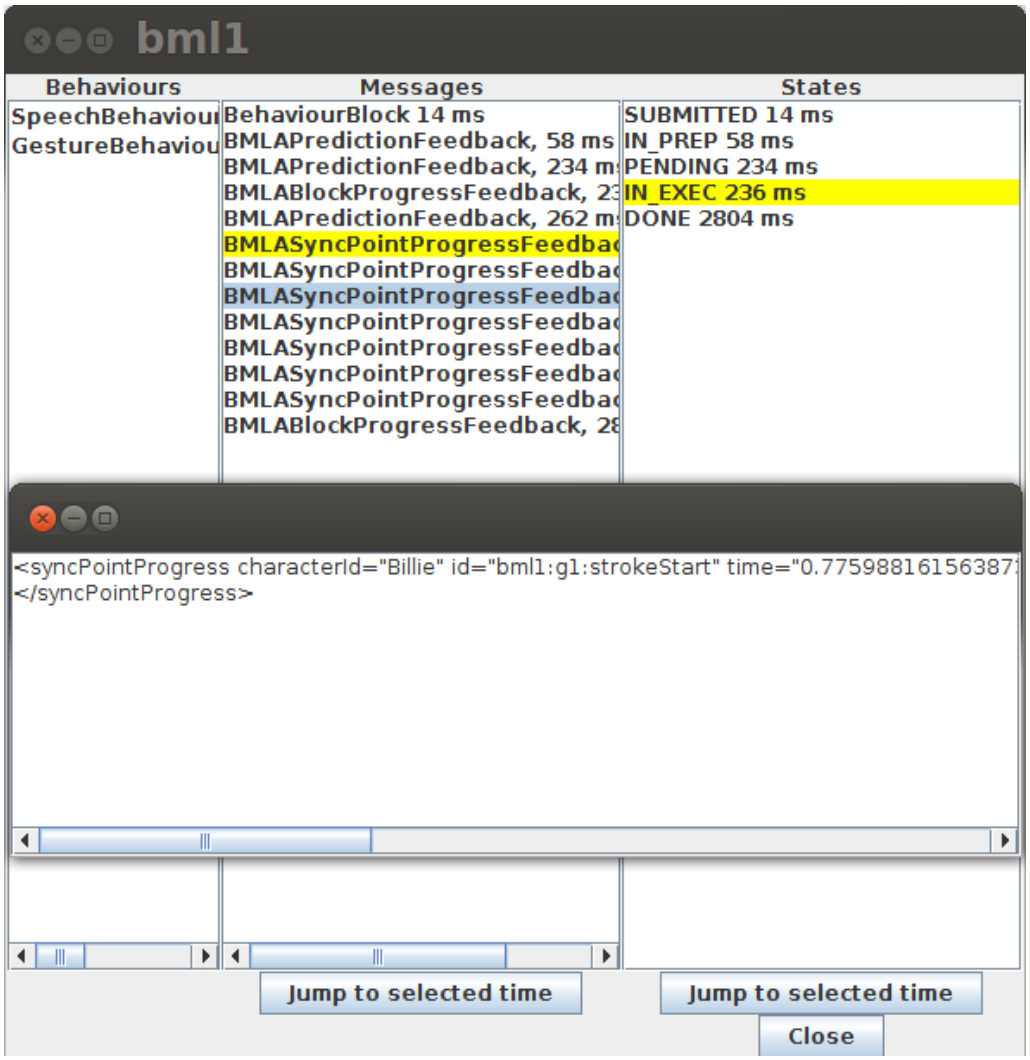

Figure 3: Block information window of the BMLFlowVisualizer. This gives detailed information about the different behaviours this block contained as well as the different received messages that are related to this block. The second window shows the textual information about a specific message.

4.4 Block colours

As can already be seen in the example in figure 1, the blocks have different colours. The colours represent the current state of the bml blocks. A quick overview about the meaning of the different colours can be found in the Information window, accessible using the menu or the shortcut $CTRL + I$. Currently not all possible bml states have their own colour, mainly because not all possible states were actually used at this time.

Figure 4 shows the above mentioned Information window, including the meaning of the different colours

5 Implementation Details

In this section a few implementation details will be given.

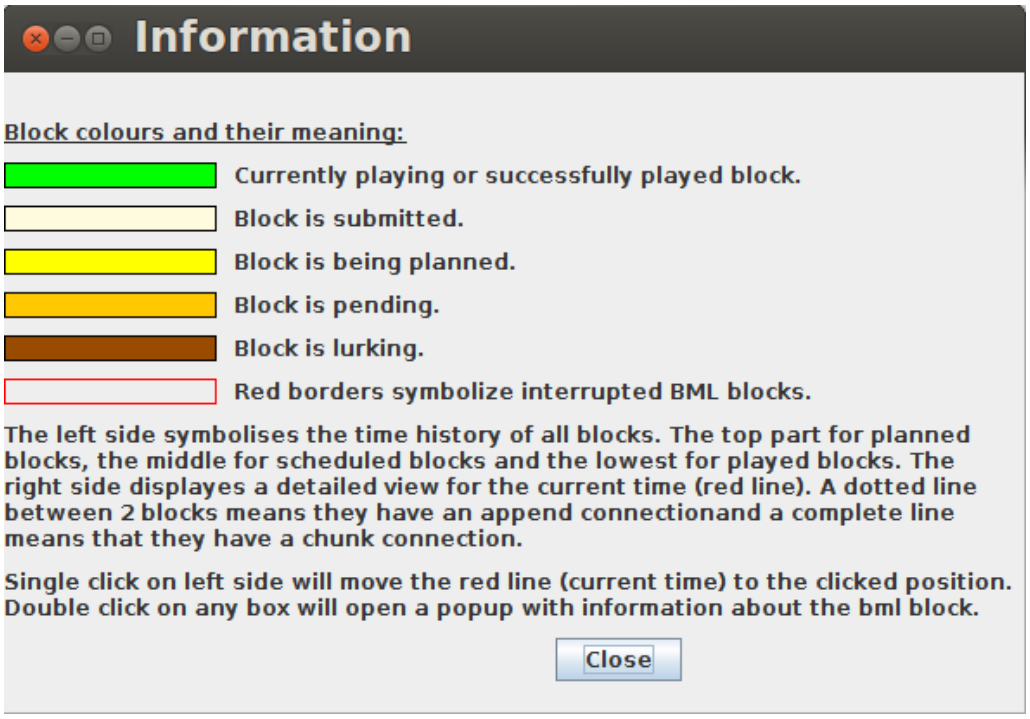

Figure 4: General Information window of the BMLFlowVisualizer. This gives information about the different block colours as well as an overview about the program in general.

5.1 BMLBlocks

A BML session always consist out of at least one bml block that is processed and goes through several phases. Information about the phases is being published to registered listeners by events (see section 5.2 for details). In the BMLFlowVisualizer all the received information about a given bml block is stored in an object call BMLBlock. The information this class stores include all the feedback messages that were received concerning the block and information about the bml block status including the times these statuses were updated. Furthermore the BMLBlock class provides helper functions for other functionality of the program like formatting the included behaviours.

5.2 Event flow

The general concept of the event flow can be seen in figure 5. There are two types of events: bml_block information and feedback. Both these events are listened two by two separate listeners. In both cases the event message that is received is stored in a wrapper object called BMLInformation for saving (see 5.3 for more details). Afterwards, the event is processed which differs depending on the event type.

bml_block: A bml_block message is fairly simple. When receiving such an event. It is checked if the behaviour block contains the *REPLACE* composition. In that case, the list of stored bmlBlocks is cleared since this signals a session reset.

If the list of BMLBlocks does not contain a BMLBlock with the id of the received bml block than the block will be added to the list. This check is only necessary, because network timing might deliver a feedback before a bml_block event is received. Usually

the bml block should not be present before and will be added to the list. Furthermore, each received bml block is analysed for interrupt behaviours or attributes. If an interrupt is found, the targeted bml block will be interrupted, i.e. the interrupted flag will be set for that BMLBlock.

Feedback: There are several different types of feedback events. The types that are used at this point in time in the BMLFlowVisualizer are: BMLABlockProgressFeedback, BMLAPredictionFeedback, BMLWarningFeedback and BMLSyncPointProgressFeedback. Regardless of the event type, the feedback message will be added to the responsible BMLBlock and the status of the BMLBlock will be updated according to the event message.

BMLABlockProgressFeedback is used to communicate the sates DONE and IN_EXEC.

BMLAPredictionFeedback is used to communicate the states IN_PREP, PENDING and LURKING. Furthermore, behaviour synchronisation points will be updated according to contained behaviour predictions.

BMLWarningFeedback and BMLSyncPointProgressFeedback are not used to communicate any bml state progression, but can contain further information about bml blocks.

These messages are just added to the feedback list of the responsible BMLBlock and can be viewed in the BML Information window.

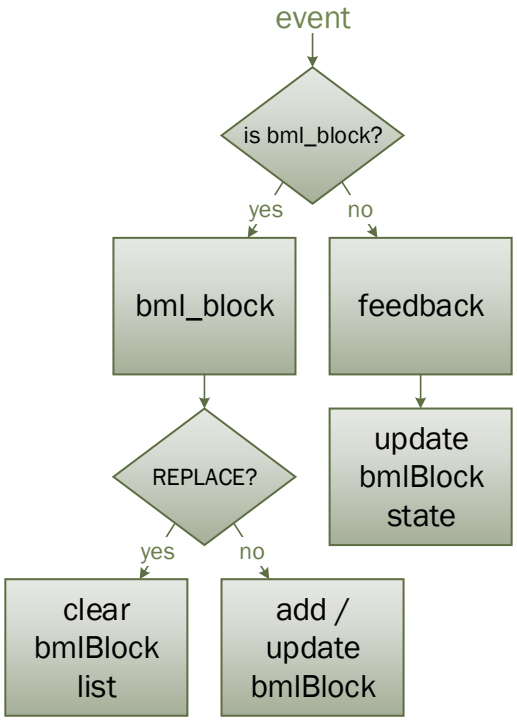

Figure 5: Overview about the event flow when receiving a bml message (either a bml_block message or a feedback message).

5.3 Information about saving and loading a session

The BMLFlowVisualizer can save a recorded session, which can be loaded and inspected at a later time. This is achieved by storing all the received feedback or bml blocks in a list of wrapper objects called BMLInformation. The BMLInformation stores the actual message, the type (feedback or bml_block information) and the time the the message was received relative to the start of the first message that was received. When saving a session, the list of BMLInformation is serialized using standard Java serialisation and the resulting bytestring is stored in the given file.