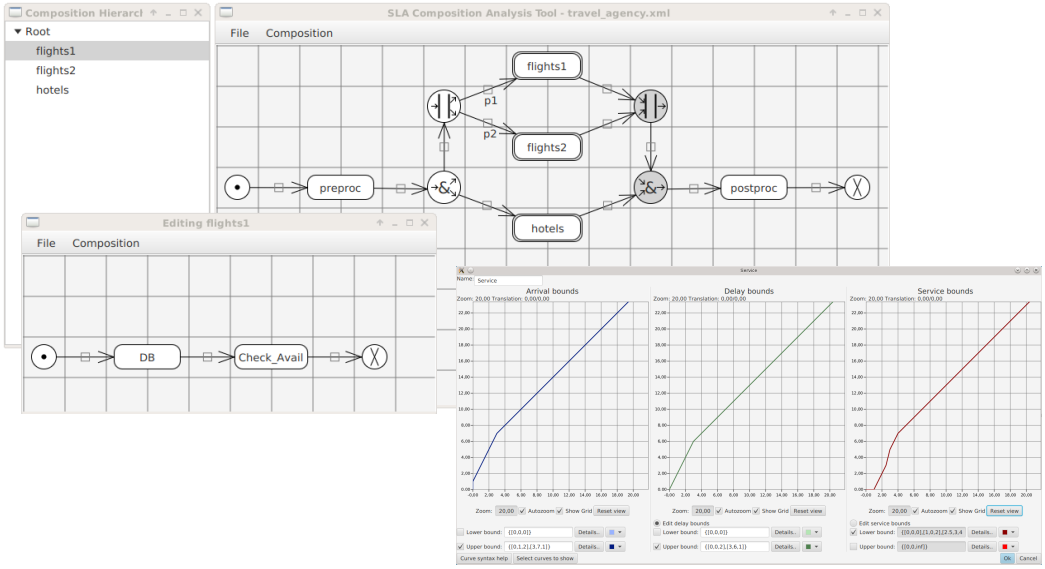
SLA Tool – A Tool for the Analytical Evaluation of Service Level Agreements –

Manual

Falko Bause, Peter Buchholz, Johannes May
Informatik IV, TU Dortmund
{bause,buchholz,may}@ls4.cs.tu-dortmund.de

February 2017



Contents

List of Figures

1 Introduction

SLA Tool [1] supports the analysis of quantitative Service Level Agreements (SLAs). Bounds for delays of composed systems are determined from bounds for the load and the delay of the individual components. Additionally SLA Tool allows for the calculation of bounds for the required service capacity necessary to guarantee the quality of service defined in the SLAs.

This document is structured as follows. Installation instructions for SLA Tool are given in Sect. 2. Sect. 3 contains an introductory tutorial and Sect. 4 explains the use of SLA Tool in detail.

2 Installation

The latest version of SLA Tool can be obtained as a zip archive from `http://ls4-www.cs.tu-dortmund.de/cms/en/research/software/SLA_Tool/index.html`

The archive contains the GUI of SLA Tool and a collection of Octave scripts which are used for analysis.

2.1 License

SLA Tool is free software, licensed under the Apache License, Version 2.0 (the "License"); You may obtain a copy of the License at `http://www.apache.org/licenses/LICENSE-2.0`

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

2.2 System Requirements

SLA Tool is available for Linux and Windows. On both systems, at least Java 7, JavaFX 2.2 and Octave 3.8.2 are required.

2.2.1 Linux

Depending on your Linux distribution JavaFX might be included in your Java package, OpenJDK users might need to install OpenJFX separately.

2.2.2 Windows

On Windows JavaFX is contained in the Oracle JRE or Oracle JDK.

2.3 Installing SLA Tool

2.3.1 Linux

1. Extract `slatool1.0.zip` to a local directory:

```
unzip slatool1.0.zip -d ~
```

The directory should now contain a subdirectory called `SLA Tool`, which contains three files (`sla.jar`, `run.sh` and `manual.pdf`) and three subdirectories (`libs`, `OctaveScripts` and `samples`).

2. Change to the `SLA Tool` directory and run `./run.sh` to start SLA Tool's GUI.

2.3.2 Windows

1. Right click `slatool1.0.zip` and select "Extract..." from the files context menu and choose a target folder (the recommended directory is `C:\`).

2. Open the directory `SLA Tool` inside your target directory and try to execute the tool by double clicking `run.bat`. If Java is not in your `PATH`-variable, this may fail and the tool will not start. To solve this problem, edit the second line of the batch file and append a `;` and the path to the directory which contains your java.exe file (such as `;C:\Program Files\Java\jre1.8.0_111\bin`).

3. Usually a message indicating that Octave could not be executed will appear. Afterwards the configuration dialog for tool settings will appear, allowing you to set the Octave executable. The default path chosen by Octave during its installtion is `C:\Octave\Octave-VERSION\bin\octave-cli.exe`. After setting the correct path, launch the tool again.

3 Quick Start Guide

The following sections explain step-by-step how SLA Tool's GUI can be used to build up an example network of services and how this network can be analyzed. This quick-start guide will only explain the basic functionality that is necessary to specify the example network and gives a first overview of the features of SLA Tool. A detailed description of SLA Tool that introduces all the features and properties will be given in Sect. 4.

3.1 Specification of an Example Network of Services

The following example is from [1] and represents an agency which sells travel packages to customers via the Internet (cf. Fig. 20). The agency uses services offered by external providers to run its business model. The SLA specifications of external services is known (cf. Table 1) and the agency wants to determine a beneficial orchestration of services.

1. Starting SLA Tool by calling `run.sh` gives a window with a nearly empty canvas as shown in Fig. 1.

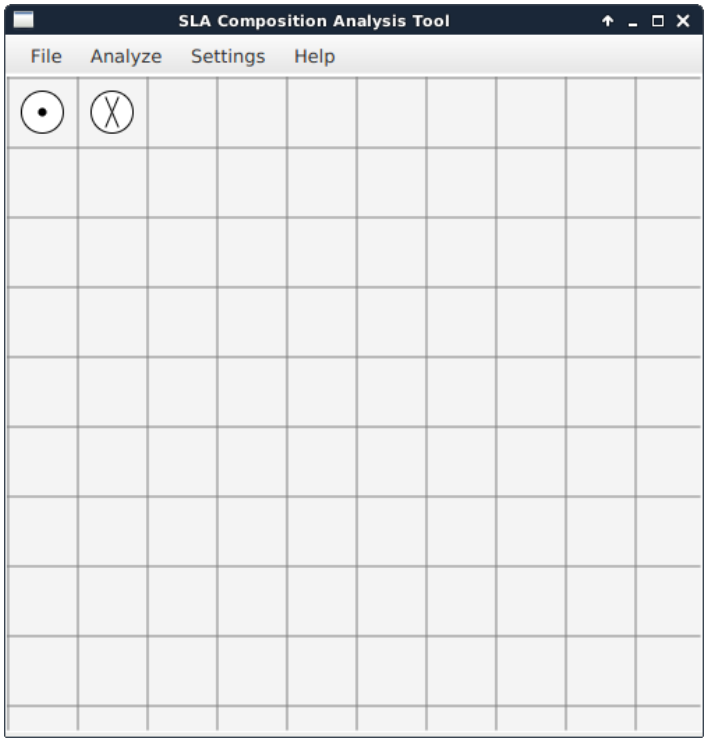


Figure 1: Main canvas after initial start

Since a network of services has exactly one source and one sink node these nodes are already present. Both nodes can be moved by clicking and dragging them with the left mouse button.

2. Describing a network of services is essentially the specification of a graph with nodes and edges. The first two nodes of the graph, the source and the sink, are already present, so let us insert a third node. Right-clicking on the canvas gives the menu shown in Fig. 2 and selecting `New service` results in the situation displayed in Fig. 3

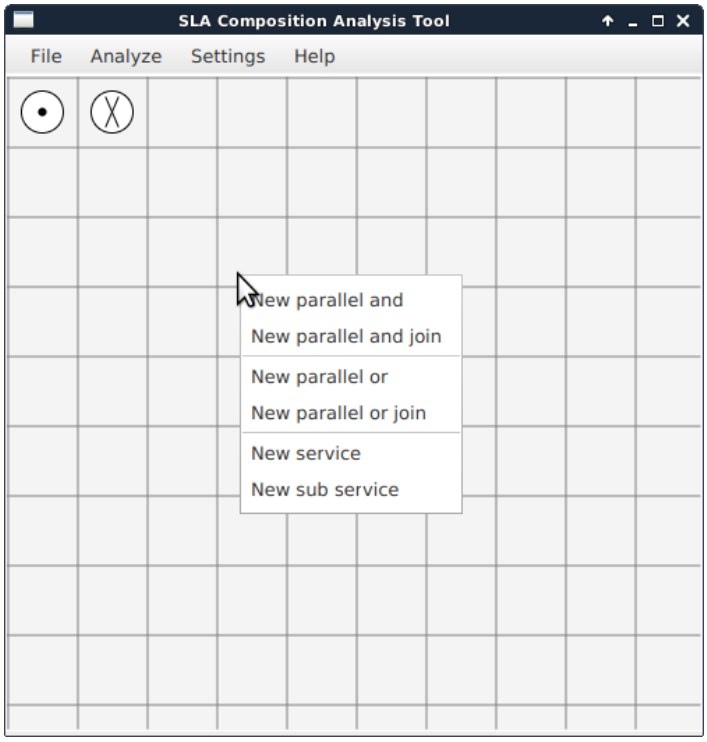


Figure 2: Inserting a service node

For the next step we will configure the parameters of the new service, which can be done by selecting the `Edit Service` option from the right-click menu of the service node (see Fig. 4). This opens the parameter window of the service as shown in Fig. 5.

Next we specify the parameters of the service node.

3. The parameter window of a service shows curves for arrival, delay and service bounds. All curves are specified by piecewise linear segments and can be edited in a corresponding text field below the shown curves. In our example only upper bounds will be specified and the default lower bound $\{[0,0,0]\}$ is left unchanged.

   First we edit the upper bound for arrivals, let's call it $\alpha^U(t)$, and change it to $\{[0,6,20]\}$, which describes a single line segment starting at x-value 0, y-value 6 (i.e. at point $(0,6)$) with a slope of 20. The textually specified curve is automatically displayed as a graph above the corresponding text field. Next the upper bound for delays is changed to $\{[0,0,2]\}$. This results in an updated display of the curves for delay bounds and additionally the service bounds are updated,

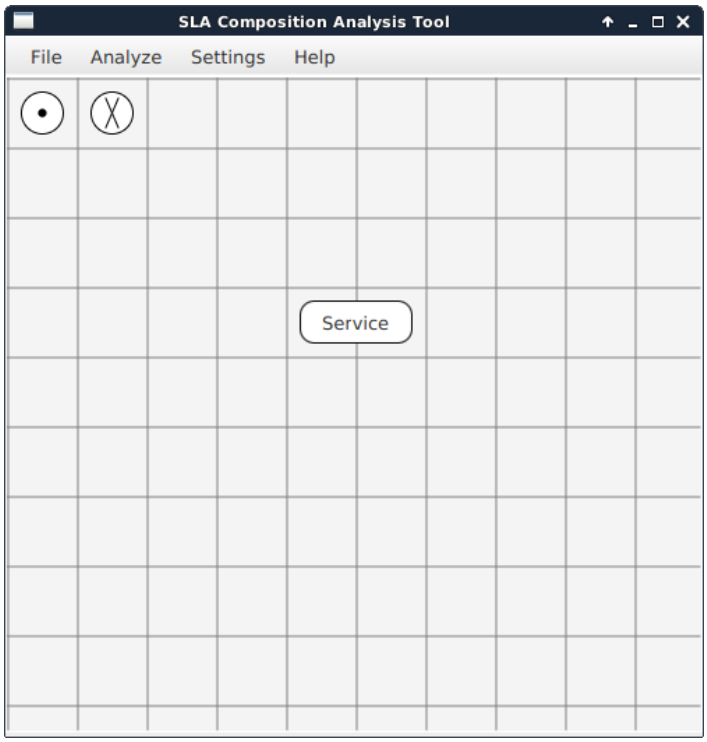


Figure 3: After insertion

which specify the service capacity being necessary to satisfy the delay bounds for the given arrival bounds.

These upper bounds can be interpreted as follows. Considering the arrivals, at most 6 load units are allowed to arrive at the same time and in any time interval $[0, t]$ at most $\alpha^U(t)$ load units are allowed to arrive in total in order to guarantee the specified delay. The x- and y-axis of the delay curve can be viewed as a mirrored version of the arrival curve. The x-axis describes the number of load units and the y-axis the delay/time for those load units in order to be served. E.g. $\{[0, 0, 2]\}$ specifies that a load unit needs at most two time units for service and a point, e.g. $(3, 6)$ of the service curve specifies that 3 arriving load units will leave the service node at most 6 time units after their arrival, provided the arrival stream observes the upper arrival bound.

Furthermore we change the name of the service in the upper left text field to `preproc` resulting in the window shown in Fig. 6. More details on the specification of bounds and their interpretation are given on page 19. Finally we close the window by clicking the `OK`-button.

4. The preproc-service we just configured is the first step that customers take to book a journey, so we add an edge from the source to the service by left-clicking the source and afterwards the service (see Fig. 7).

   After moving the source node to a better position, the graph will look like Fig. 8.

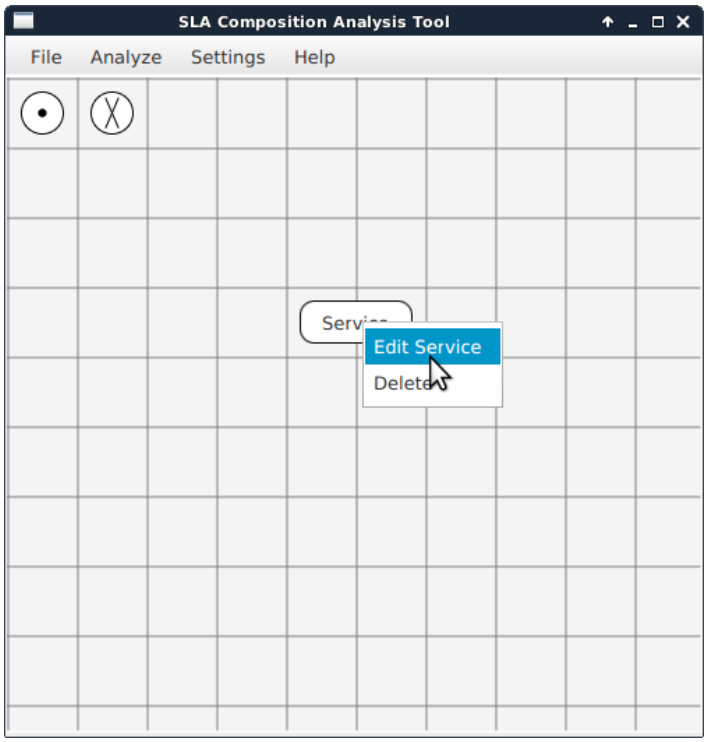


Figure 4: Editing the service node

5. After customers have been served by the preproc service, they have to choose a flight and a hotel. For this example we assume that two different external providers for the flight selection exist, but only one provider for hotel selection. Instead of clients performing hotel and flight selection sequentially, we decide to parallelize these calls, which requires a new type of node. But first we will create the needed services for hotel and flight selection. Since they are externally provided, they might be modeled by a more complex system behind the scenes. This is where hierarchical models can help. They allow us to specify an entire network of services within a service node. We use this feature by creating three subservice nodes (see Fig. 9), rename each of them using the right-click menu and end up with a graph like Fig. 10.

   To edit the structure of these subservices, you can select `Edit subservice` from its right-click menu. This will pop up a window like Fig. 11 which has the same structure as the editing window for the root composition node. In this example both flight services will use two sequential services for database access and flight selection which results in an arrangement like Fig. 12.

   The bounds for these services are configured as shown in Table 1.

   The hotels subservice is modeled by just one service node as shown in Fig. 13 with the bound specification given in Table 1.

6. After configuration of the subservices, we need to model the simultaneous selection of hotels and flights as well as the splitting of customers to the two flight

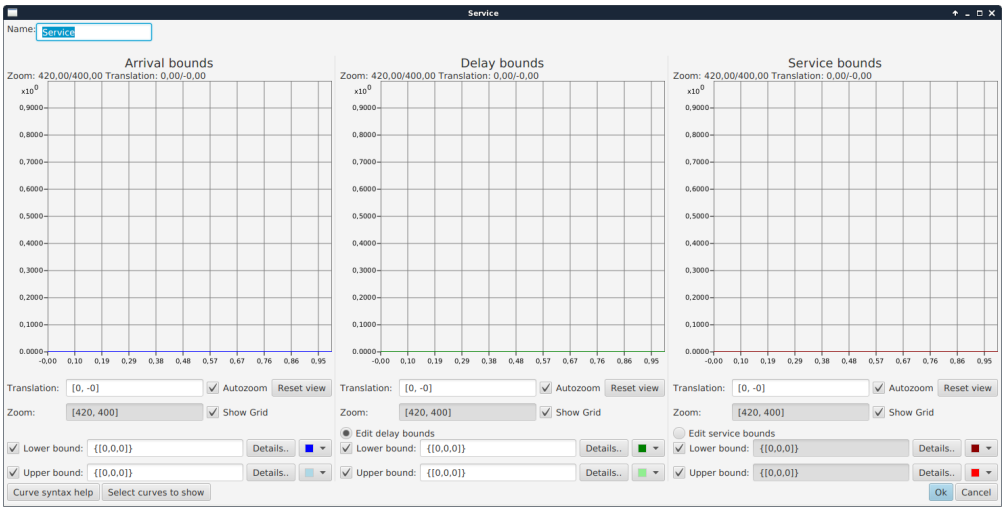


Figure 5: Parameter window of the service node

providers. For both of these tasks we can use split nodes, which have one incoming and two outgoing edges. The first kind of split node is the `and split`, which is used for modeling the simultaneous traversal of two branches. The second kind is the `or split`, which models a decision for one of two branches based on probabilities. Both splits also require a join node that has two inputs and one output, which leads to a structure of opening and closing nodes similar to XML. We will now model the split between `flights1` and `flights2`. As explained above we can use an `or split node` for this, which can be created from the same right-click menu as services and subservices. Since every split also requires a join, we also create an `or join node` and connect both to the two `flights` services (see Fig. 14).

For modeling the simultaneous call of hotels and flights services the `and split node` is used. We insert an `and split node` and an `and join node` and connect them as shown in Fig. 15. This structure models the fork of a request after having been served by the preproc service resulting in a "call" to the `or split node` and one of the `hotels` service.

7. The `or split node` we inserted in step 6 offers some configuration options. The configuration dialog is opened by right clicking the `or split node` and choosing `Edit Or Split Node` from the menu (see Fig. 16).

   The dialog allows setting values for `p1`, `p2` and `bmax`. `p1` is the probability for a request to be passed to the edge marked with `p1` and `p2` is the probability for the other edge. Both values have to be chosen such that $0 \leq$ `p1`, `p2` ≤ 1 and `p1` $+$ `p2` $= 1$. If you enter a value for `p1` the matching `p2` value will automatically be calculated and vice versa. The settings for the travel agency model are `p1` $= 0.3$ and `p2` $= 0.7$ (see Fig. 17). A value of `bmax` = 0 means that arriving load can be split into equal parts, as it is the case for fluid. `bmax` = 1 means that calls can have maximal size of 1 and cannot be split.

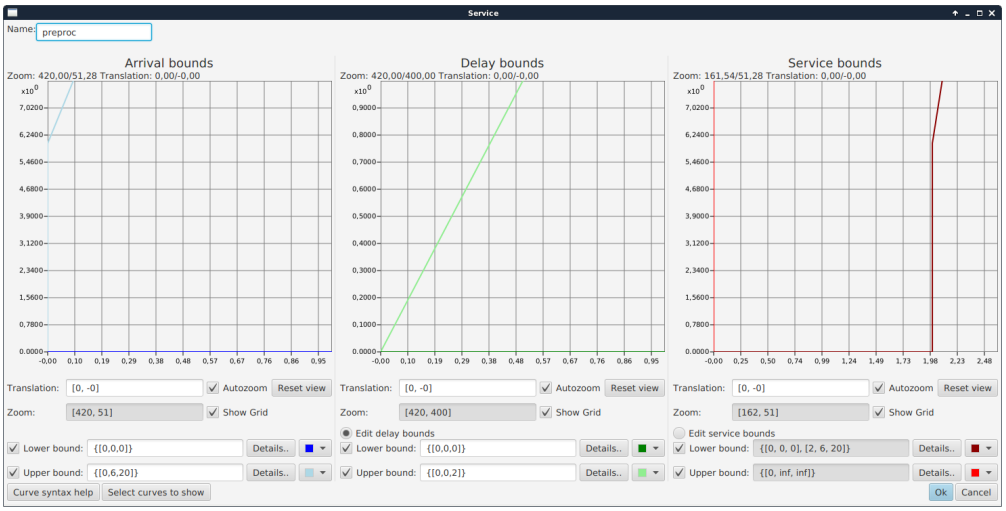


Figure 6: Edited parameter window of service node `preproc`

8. After the hotel and flight selection is done, the actual booking of the travel package is finalized in a last step. Therefore we specify an additional service called `postproc` with bounds set as described in Table 1 and add an edge from the `and join node` to this service (see Fig. 18).

9. The last step completing the structure of our model is the connection of service `postproc` with the sink.

10. In order to specify an analyzable model, we finally have to describe the arrival bounds for the source. These, in our example, upper bounds represent the maximum number of customers entering the system. The interpretation is similar to the one for arrival curves. In the tool the bounds of the source node can be specified by right clicking the source node and selecting `Edit Arrival bounds`. Enter the arrival curves shown in Table 1 and click `Ok` (see Fig. 19).

Fig. 20 presents the complete network together with the hierarchy window and a window showing subservice `flights1`. The SLA specification of all services is given in Table 1. All lower bounds are $\{[0, 0, 0]\}$.

3.2 Analyzing the Network

This section explains the analysis of the network described in Section 3.1. To start the analysis click the `Analyze composition` button in the `Analyze` category of the application menu (see Fig. 21). The analysis usually takes a few seconds and once it is done, a window like Fig. 22 will open.

The delay bounds are the bounds for the accumulated delay when customers arrive according to the previously configured arrival bounds of the source node. A point as e.g. $(169.33, 4104.93)$ of that curve expresses that if 169.33 customers are in the system their delay will be at most 4104.93 time units. A point (x, y) of a service bound expresses that in x time units the service is able to handle y customers.

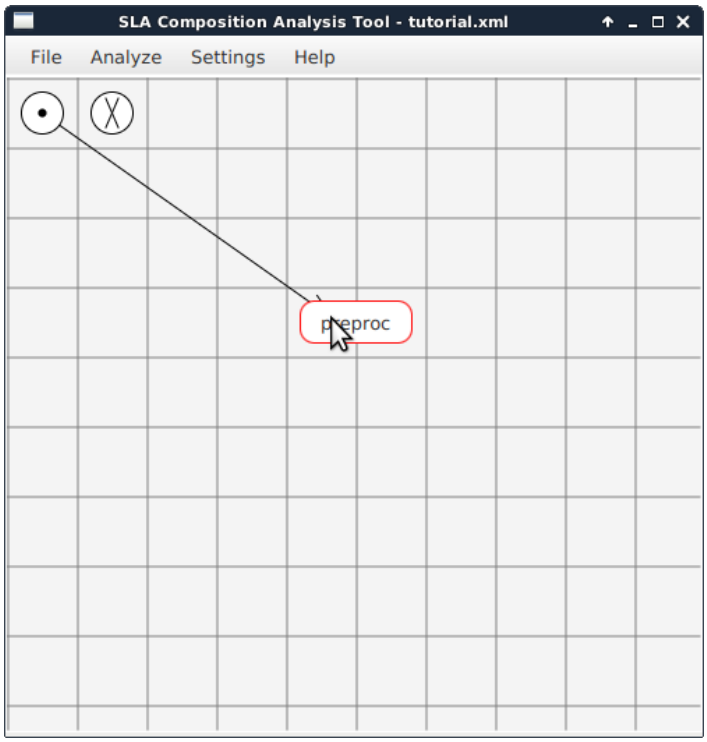


Figure 7: Adding the first edge

Table 2 shows the resulting upper delay bounds that can be guaranteed by the agency directing different portions `p1`, a parameter of the `or split node`, of the load to `flights1` assuming that each customer has to use one of the two `flights` services (`bmax`=1). (Note that $p2 = 1 - p1$ implying $0.125 < p1 < 0.625$). It follows form the SLA specifications that it is beneficial to direct as much as possible requests to `flights1`, but only in very low traffic periods. For typical situations it is much better to select $p1 \approx 0.52$, since the corresponding resultant upper delay bound is optimal. This means that the final delay of 15 time units per call is reached as soon as possible and the accumulated delay at some given time is minimal.

Table 1: Arrival and delay curve specifications for example network

Service	Upper arrival bounds	Upper delay bounds
preproc	$\{[0, 6, 20\}$	$\{[0, 0, 2]\}$
flights1: DB	$\{[0, 6, 12]\}$	$\{[0, 0, 3]\}$
flights1: Check_Avail	$\{[0, 1, 20], [3, 61, 5]\}$	$\{[0, 0, 10], [3, 30, 8]\}$
flights2: DB	$\{[0, 6, 12]\}$	$\{[0, 0, 3]\}$
flights2: Check_Avail	$\{[0, 1, 8], [1, 9, 7]\}$	$\{[0, 0, 20], [1, 20, 6]\}$
hotels: ProcReq	$\{[0, 10, 20]\}$	$\{[0, 0, 3]\}$
postproc	$\{[0, 6, 20]\}$	$\{[0, 0, 2]\}$
source	$\{[0, 20, 50], [2, 120, 8]\}$	—

Table 2: Different `p1` values and their corresponding upper delay bounds

`p1`	Upper delay bound
0.13	$\{[0, 0, 47.0], [2.02, 94.92, 33.0], [169.33, 5616.83, 30.6], [267.89, 8632.46, 28.55], [22210.07, 635153.81, 15]\}$
0.2	$\{[0, 0, 42.61], [2.05, 87.36, 28.61], [169.33, 4873.87, 26.21], [241.34, 6760.85, 24.97], [1490.8, 37960.64, 15]\}$
0.3	$\{[0, 0, 38.2], [2.13, 81.3, 24.2], [169.33, 4126.9, 21.79], [211.5, 5045.76, 21.42], [646.41, 14360.53, 15]\}$
0.4	$\{[0, 0, 34.79], [2.27, 78.85, 20.79], [169.33, 3551.99, 18.38], [188.38, 3902.08, 18.38], [381.51, 7451.96, 15]\}$
0.49	$\{[0, 0, 32.38], [2.47, 80.02, 19.48], [6.61, 160.7, 18.38], [169.33, 3152.32, 15.98], [275.42, 4847.4, 15]\}$
0.50	$\{[0, 0, 32.17], [2.5, 80.42, 19.53], [6.5, 158.55, 18.17], [169.33, 3117.12, 15.76], [268.17, 4675.04, 15]\}$
0.51	$\{[0, 0, 31.96], [2.53, 80.88, 19.67], [6.39, 156.83, 17.96], [169.33, 3083.02, 15.55], [261.3, 4513.28, 15]\}$
0.52	**$\{[0, 0, 31.75], [2.56, 81.39, 19.88], [6.29, 155.48, 17.88], [169.33, 3071.25, 15.48], [258.4, 4449.68, 15]\}$**
0.53	$\{[0, 0, 31.55], [2.6, 81.96, 20.14], [6.19, 154.31, 18.14], [169.33, 3113.52, 15.73], [285.31, 4938.16, 15]\}$
0.54	$\{[0, 0, 31.36], [2.63, 82.59, 20.44], [6.1, 153.36, 18.44], [169.33, 3164.09, 16.04], [318.56, 5557.48, 15]\}$
0.55	$\{[0, 0, 31.17], [2.67, 83.28, 20.71], [6.0, 152.29, 18.71], [169.33, 3207.9, 16.30], [360.7, 6327.63, 15]\}$
0.56	$\{[0, 0, 31.0], [2.71, 84.04, 21.19], [5.92, 151.95, 19.19], [169.33, 3288.6, 16.79], [415.81, 7426.42, 15]\}$
0.6	$\{[0, 0, 30.29], [2.9, 87.83, 23.8], [5.6, 152.1, 21.8], [169.33, 3722.17, 19.4], [188.38, 4091.6, 19.4], [1210.6, 23918.52, 15]\}$
0.62	$\{[0, 0, 29.97], [3.01, 90.26, 26.35], [5.46, 154.73, 24.35], [169.33, 4144.76, 21.94], [192.57, 4654.66, 21.89], [7269.71, 159589.96, 15]\}$

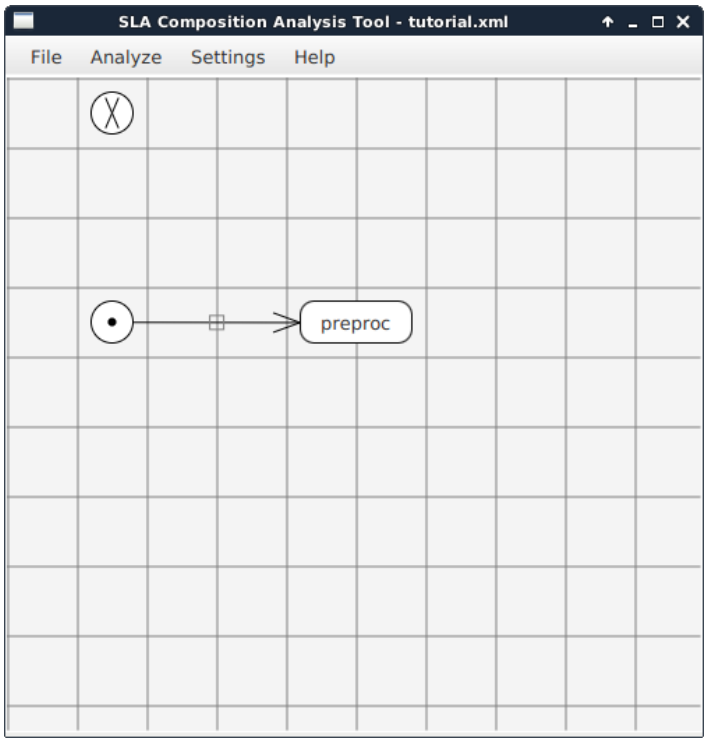


Figure 8: First connection between source and service.

4 SLA Tool in Detail

In the following we will explain the menus and options of SLA Tool in detail.

4.1 Specification of a Network of Services

4.1.1 General Outline

A network of services is a directed acyclic graph with different types of nodes. Every node type has an exact number of mandatory incoming and outgoing edges. Connections between nodes model how arrivals to the system are routed, while the different nodes manipulate the curves describing incoming arrivals based on some parameters.

Placing nodes on the grid To create a new node, right click in an empty spot on the composition canvas and choose the desired node type from the context menu (cf. Fig. 2). Any node can be repositioned by left clicking and dragging it over the grid.

Connecting nodes To create an edge between two nodes, left click the beginning node of the edge first. An arrow from the clicked node to the mouse pointer will be displayed. Once you left click the target node for the edge, it will be added to the graph (cf. Figs. 7 and 8).

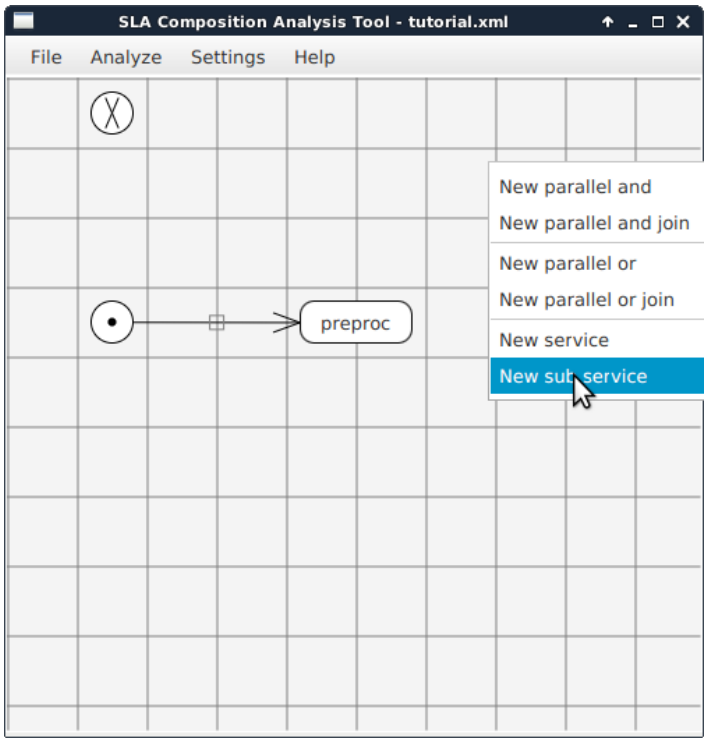


Figure 9: Adding a subservice

To cancel an edge, right click anywhere in the composition window and the arrow from the starting node to your mouse cursor will disappear.

4.1.2 Building a Network

Removing nodes All nodes except the source and the sink can be removed by right clicking them and clicking `Delete` in the menu.

Removing connections To remove a connection between two nodes, right click the gray square in the middle of the edge and click `Delete edge` in the menu. (c.f. Fig. 23)

Valid network structure Every node type has an associated number of incoming and outgoing connections, see Sec. 4.2 for details. Each node in a composition must have none or exactly the number of incoming and out-coming connections for its type. Also join and split nodes must be arranged similar to brackets in most programming languages, i.e. for a sequence of split nodes there must always be a reversed sequence of join nodes of the same types.

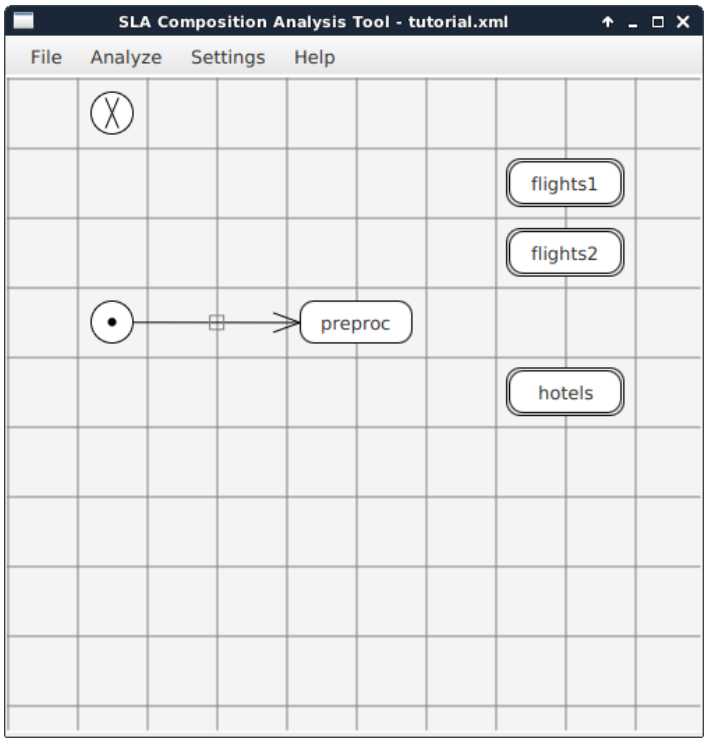


Figure 10: Three subservices

4.1.3 Analyzing a Network

Analysis results The result of an analysis are the bounds for delay and service of the network.
A point (x, y) of an upper/lower delay curve means that if x customers are present their delay will be at least/at most y time units.
A point (x, y) of an upper/lower service curve means that in x time units at least/at most y customers have or need to be served.

Overloaded systems If the arrivals as defined in the `source node` can not be handled by the system, the analysis will output the upper delay bound $[0, inf, inf]$, indicating the overload-situation.

Mathematical details For mathematical details on the analysis of SLA compositions, see [2].

4.2 Node types

Source The source node has exactly one outgoing connection and represents the entry point for arrivals to the system. Bounds for these arrivals can be set by right clicking the node and choosing `Edit arrival bounds`. A menu as shown in Fig. 19 will be shown.

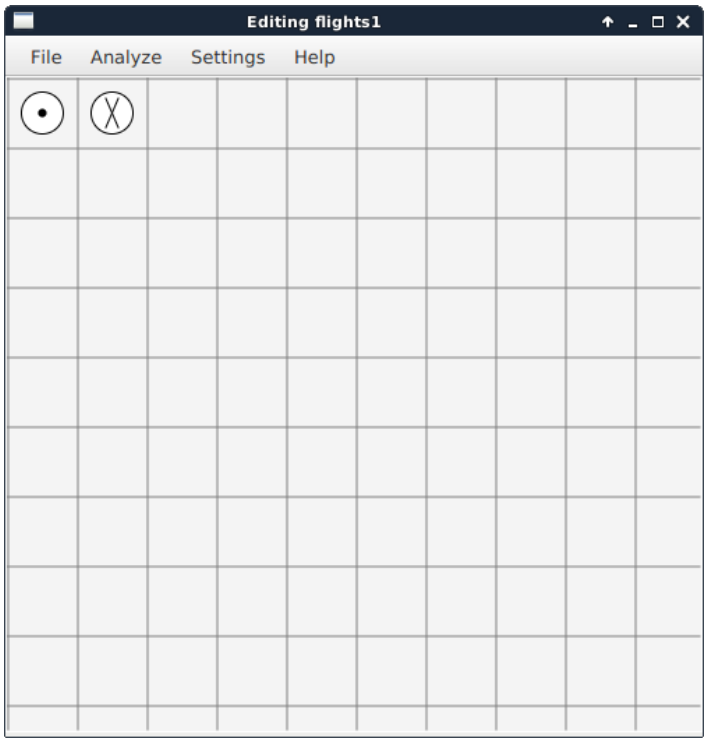


Figure 11: Editing a subservice

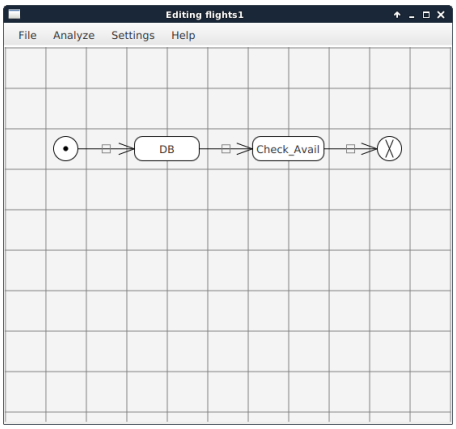


Figure 12: Structure of `flights1`

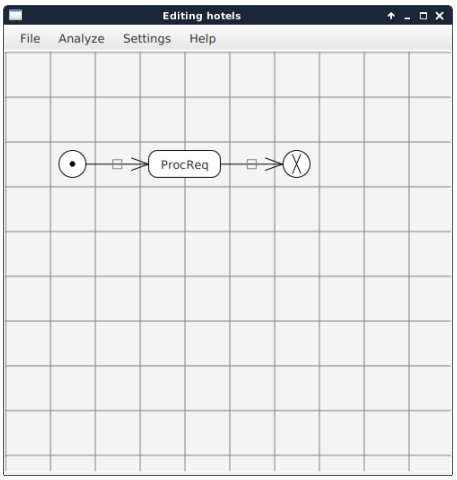


Figure 13: Structure of `hotels`

This option is not available in a subservice of a hierarchical composition, since the arrivals for those subservices are specified by the parent composition.

Sink The sink is the node at which all arrivals leave the system. It needs exactly one incoming connection and offers no settings.

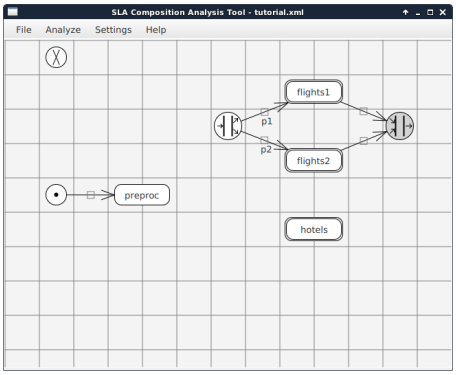


Figure 14: The or split and join node

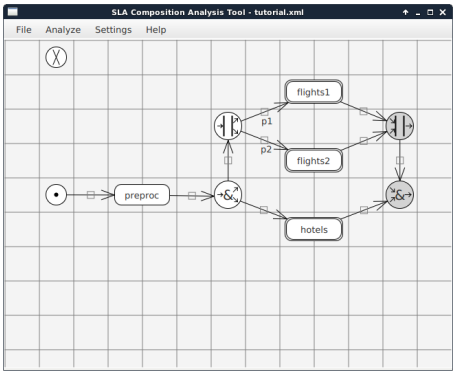


Figure 15: The and split and join node

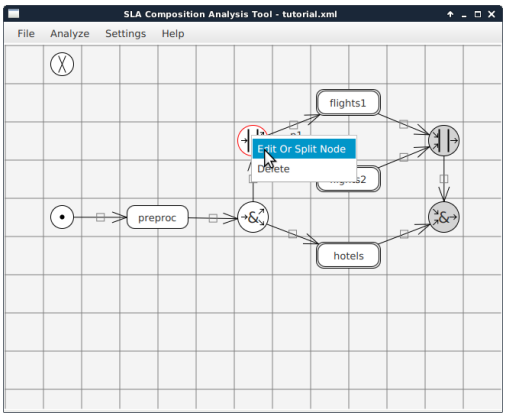


Figure 16: Opening the or split nodes configuration dialog

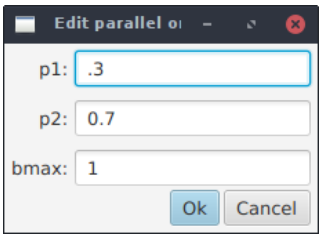


Figure 17: The `or split node` edit dialog with inserted values

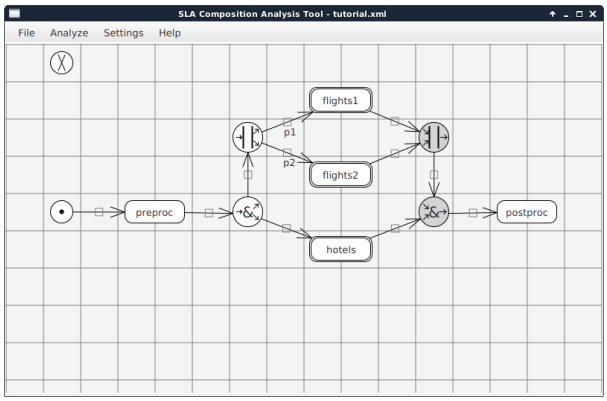


Figure 18: The added `postproc` service

Service Service nodes represent stations that handle arrivals and delay them according to their delay bounds configuration. They have one incoming and one outgoing connection. The configuration for each service consists of `arrival bounds`, `delay bounds` and `service bounds` (c.f. Fig. 5). Most of the dialogs settings (e.g. zoom factors or translation) are stored with the composition, so they are restored for anyone using the same saved file.

Subservice Subservices contain an entire composition and can even be used inside of another subservice. They allow for an encapsulation of function groups in compositions.

And split `And split nodes` can be used to have incoming arrivals traverse both branches leaving the split in parallel. They have one incoming and two outgoing connections.

And join To mark that two branches of an `and split` merge, the `and join` is used. It has two incoming connections and one outgoing connection.

Or split `Or split nodes` represent a branch where every incoming arrival has to proceed using one of the two branches. Which branch should be used is determined by the probabilities `p1` and `p2`. Another parameter of this node is `bmax`. It is the lowest possible unit of separation for arrival units. The `or split node` has two outgoing and one incoming connections.

Or join `Or join nodes` are used to merge the two branches of an `or split used`. Like the `And split node`, they require two incoming and one outgoing connection.

4.3 Main Canvas Window

The main canvas window (shown in Fig. 1) is used to create the composition as a graph. To add different nodes to a composition, the right click menu on an empty spot in the

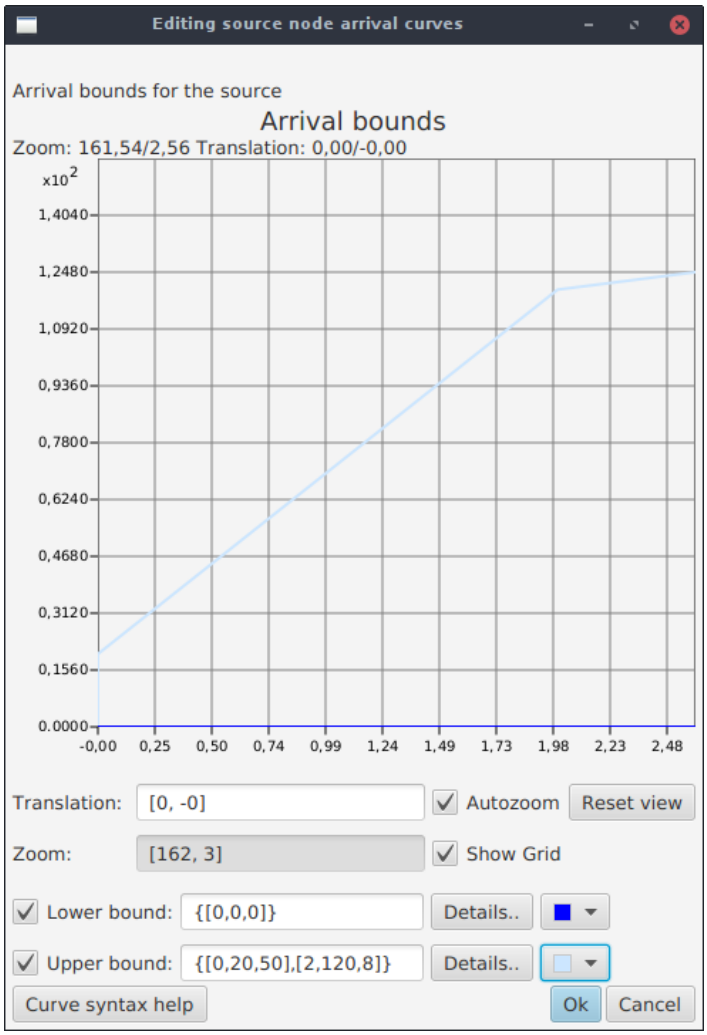


Figure 19: Setting the arrival bounds for the source node

composition is available. The graph can also be zoomed in with the scroll wheel. The menu bar at the top allows for further options and is discussed in another section.

Almost the same window is also used to edit `subservices`, only the menu bar differs in some items.

4.4 SLA bounds

`SLA bounds` are pairs of curves that are specified by piecewise linear segments. Each bound consists of a lower and an upper bound specification.

4.4.1 Curve syntax

SLA bounds are described by curves which can be specified as an array of three-dimensional vectors like e.g. `{[0, 0, 1], [10, 10, 5]}`. Each vector of a curve consists of three values x, y and m which describe one linear segment of the

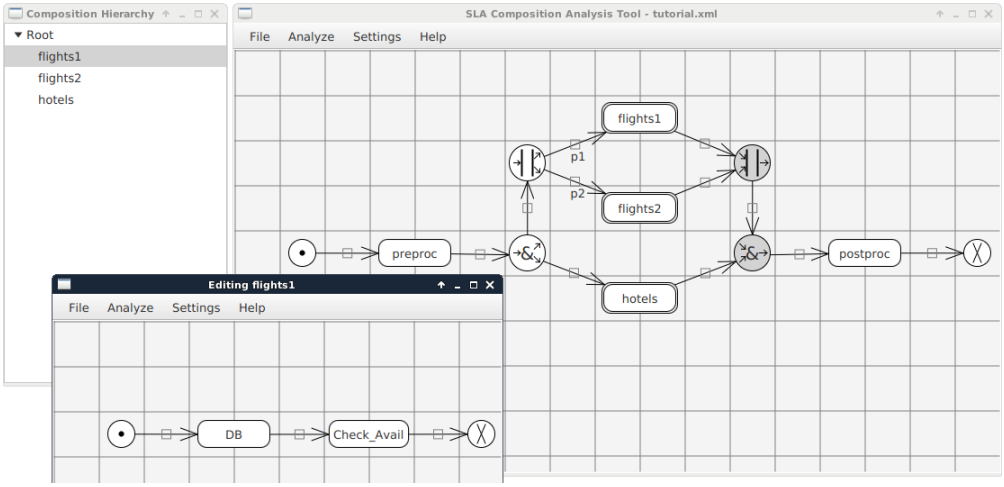


Figure 20: Example network of services

curve. (x, y) is the point where the segment starts, while m is the slope of the segment. The tool expects curve segments in the array to be ordered ascending by their x-values.

4.4.2 Bounds display

The tool uses the same kind of block of components to display all `SLA bounds` (c.f. Fig. 24).

The name for the bounds is shown at the top, below of that is some information on the zoom and translation of the curves.

In the middle of the block a coordinate system displays the upper and lower curves. By clicking the left mouse button and dragging the cursor on the coordinate system, the displayed part of the graph can be changed. The zoom factor can be changed by scrolling or by using the right mouse button to create a box to zoom into.

Below additional control entries are shown. The translation and zoom text fields allow to precisely define the values as separate `[x, y]`-values. The `Autozoom` check box activates the auto-zoom feature, which automatically calculates the correct zoom factor to display all parts of the shown curves. `Show Grid` allows to show or hide the grid of the coordinate system. The `Reset view` button resets the translation and zoom values to their default values, if the auto-zoom is activated, the zoom factor will be reset to the calculated value.

The last two rows are used for the textual description of the curves. Two check boxes can be used to hide one of the curves. The text fields contain the textual description of the curves as explained in Sect. 4.4.1. On the right of both rows, there are color selectors, which control the color of the curves in the coordinate system.

The two `Details..` buttons open another dialog, which allows to edit a single curve. This dialog is explained in detail in the next section.

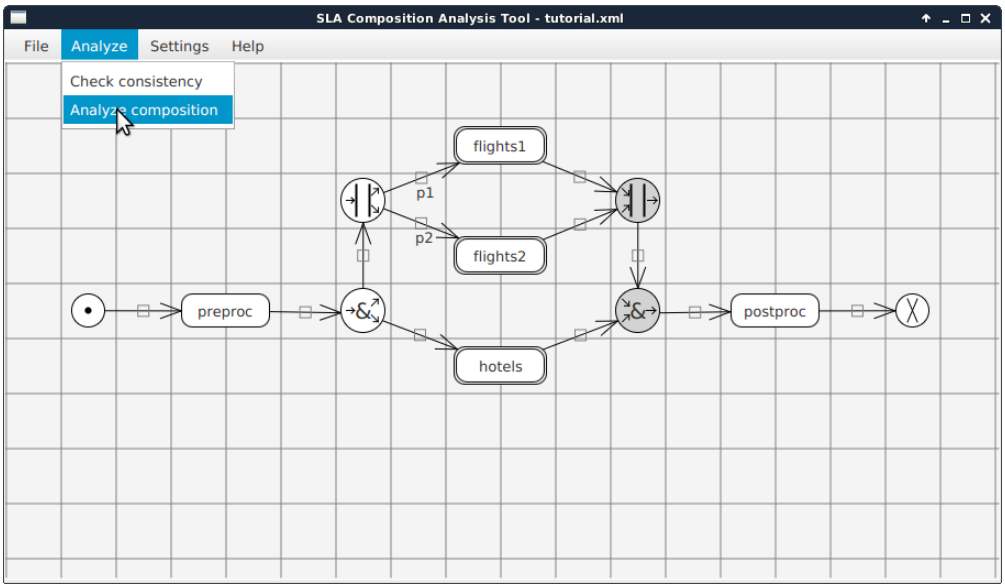


Figure 21: Starting the composition analysis

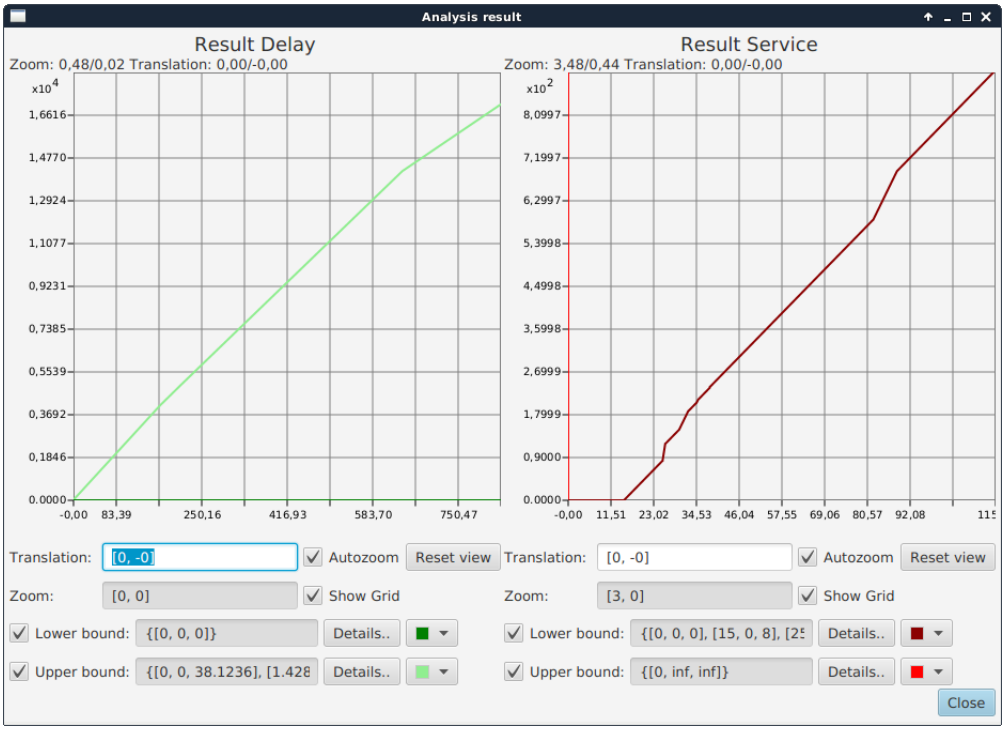


Figure 22: Analysis result for the travel agency example

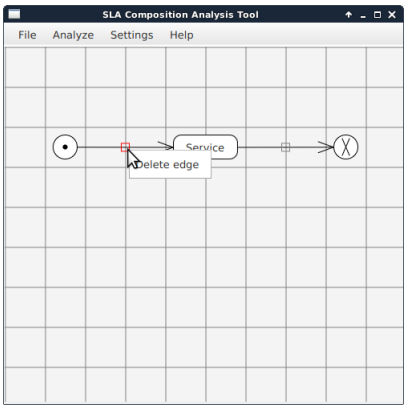


Figure 23: Deleting a connection between two nodes.

4.4.3 The details dialog

The details dialog can be opened from the bounds display and is used to edit a single curve (cf. Fig. 25). At the top the coordinate system and options for translation and zoom are shown as already explained in Sect. 4.4.2. Below is a text field which contains the curve description, which can be useful to read long curve descriptions. The bottom row of the dialog offers options to substitute the original curve specification (button *Substitute by concave function* for upper bounds and button *Substitute by convex function* for lower bounds). The original curve description is substituted by a concave or convex bound, which can be useful since e.g. upper arrival and delay bounds must be concave for the analysis.

The check box *Show full text* can be used for result curves that were calculated by the tool. If being checked, the full decimal representation of the curve parameters will be shown instead of rounded values.

4.5 Hierarchy Window

The hierarchy window gives an overview of the hierarchical structure of the composition (c.f. Fig. 26). It uses a tree view to display the hierarchy of subservices. By clicking on the elements in the tree, the corresponding subservice editing window will open.

By right clicking the elements in the window, their names can be changed. This also applies for the root composition node. The root name is used for labeling auto saves in the `Autosave` menu.

4.6 SLA Tool's Menu Structure

4.6.1 The File Menu

New Composition This button creates a new composition and closes the old one.

Load Composition Loads an XML composition file. In subservices this item is

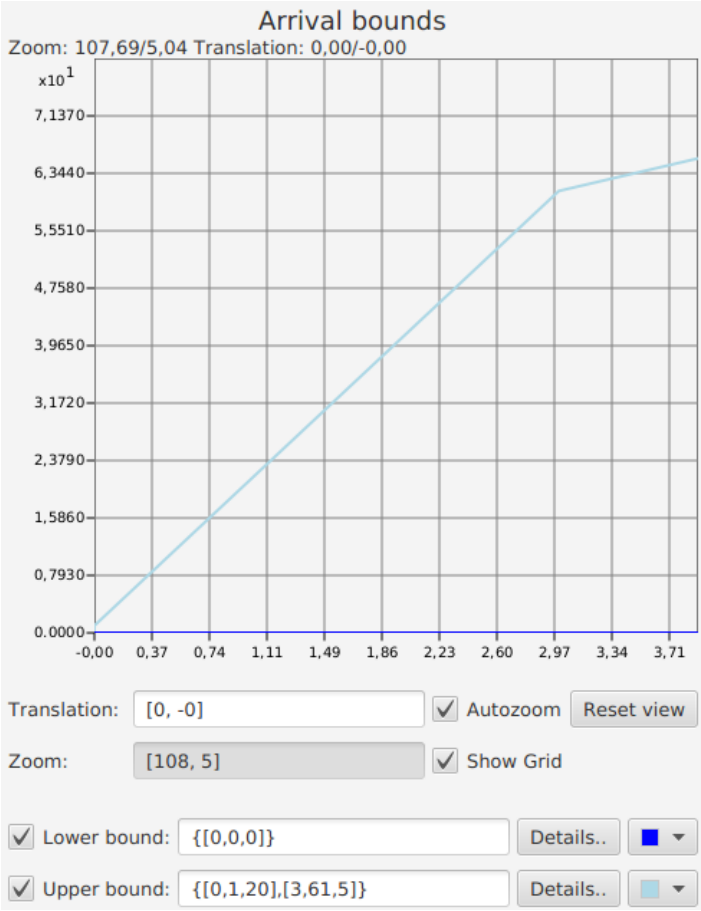


Figure 24: Bounds display

replaced by `Import`, which loads an XML file into the subservice node.

Load Autosave Loads an auto save, which is created when the tool is closed and after every 5 minutes. Works similar to import for subservices.

Recently used Recently used loads files that were recently saved or used. It can also be used to import files in subservices.

Save composition This item is only available when a filename is already known, i.e. the composition was loaded from some file or saved using `Save composition as` before. It saves to the same location again.

For subservices it exports the content of the subservice as an XML file of its own.

Save composition as Saves the current composition in a file selected by the user. In subservices it says `Export as..` and only stores the content of the subservice.

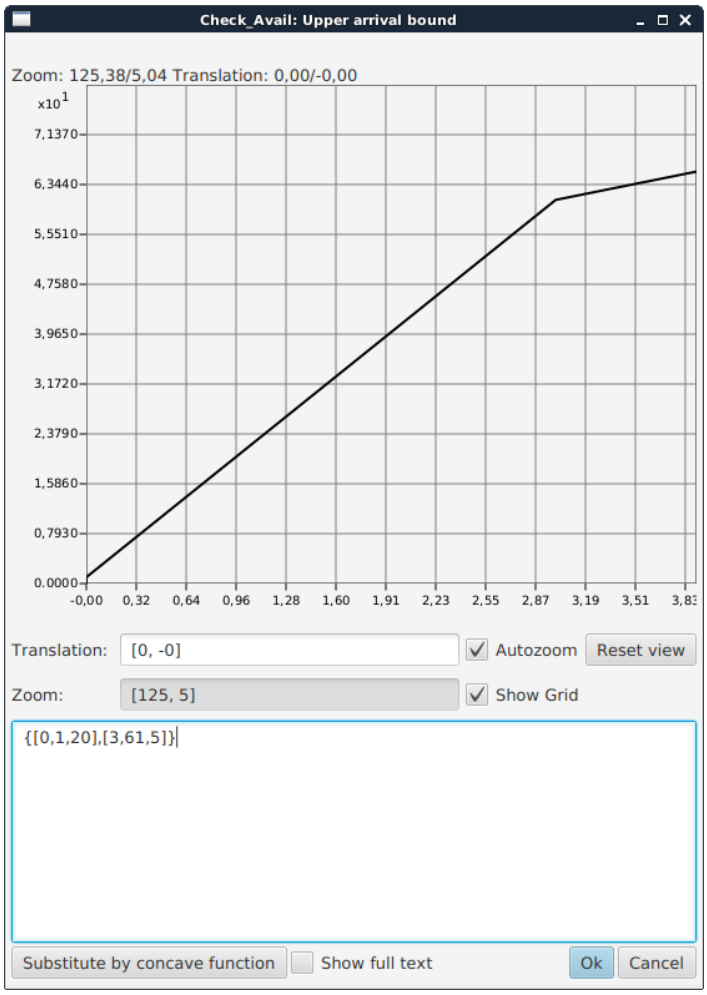


Figure 25: The details dialog.

Show hierarchy This option is only visible when the `Hierarchy window` has been closed before and will open the hierarchy windows again.

Quit Quits the tool.

4.6.2 The Analyze Menu

Analyze Checks the structural validity of the composition and starts the analysis if it is valid. The analysis may take some seconds, in which the tool will not be responsive. The results of the analysis are then displayed in a dialog.

Verify Only checks the structure of the composition, i.e. whether all nodes satisfy the connection rules described in Section 4.2. The result is displayed in a message box.

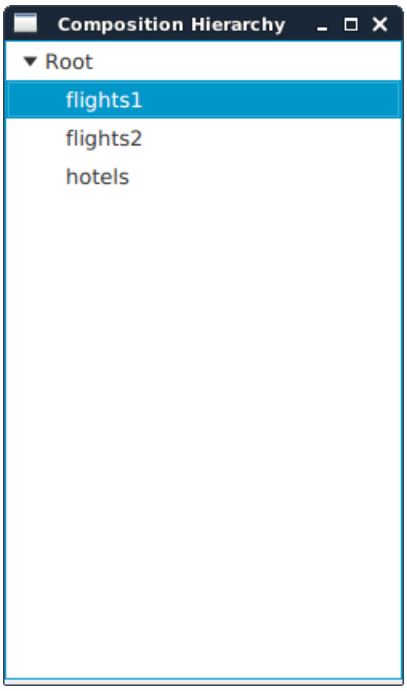


Figure 26: The hierarchy window.

4.6.3 Settings

The settings are divided into two sections. The `Composition settings` are specific for every composition and are also stored in the corresponding XML-file. `Tool settings` on the other hand are settings that only apply to the local system.

Both buttons in this menu open the same dialog but preselect different tabs. The `Save as default` button at the bottom of this dialog stores the Composition settings as default, so they will be used for new compositions.

Composition The option *Magnetic grid* enables or disables nodes in composition graph snapping to the next grid position when dragged and dropped.

Show grid can be used to hide the grid in the background of the composition graph.

A composition might still be valid if some nodes have no connection to the subgraph connected to the source, since it may happen that different nodes can be swapped for an experiment. The warning that is shown for unconnected nodes can be disabled with the *Show warning on unconnected nodes* option.

Tool *Maximum fraction digits* is the maximum number of digits in the fraction part of a number, if the number is shown rounded.

The zoom behavior of the SLA Tool is not linear. Instead a fixed number of zoom steps for each decade of zoom is used, i.e. the intervals $[0, 10]$, $[10, 100]$ and $[100, 1000]$ all contain the same number of zoom

steps. The option *Steps per zoom decade* can be used to change the steps per decade.

In case your Octave installation is not available via the `PATH` variable, the option *Octave executable* allows to set a command to execute for octave.

By default the SLA Tool will show a reminder when you exit the tool with unsaved changes or when you try to load another file. This reminder can be disabled with the option *Show save reminder*.

To prevent malicious code in composition files to be executed, the *Strict Octave syntax check* checks every input before passing it to Octave. In case this behavior causes any problems, it can be disabled using the corresponding checkbox.

4.6.4 Help

Show log This option is only available when the log window is not visible. The log contains information about all calls to Octave and which of them failed. The log might be useful if something goes wrong while using the tool.

Help Opens this file using the systems default application.

About Shows a dialog with some information on the tool.

A Samples

The SLA Tool contains several samples containing compositions for experimentation with the features in the tool. The samples can be found inside the `samples` directory inside the directory of the SLA Tool.

A.1 Travel agency

The `travel_agency.xml` file contains the sample described in 3.

A.2 Online shop

`online_shop.xml` models a very simple checkout process in an online shop. It assumes that every customer has to take a routine to begin the checkout, while the payment is then provided by an external contractor.

Possible additions to this sample would be multiple eligible payment providers or modeling the whole checkout process in finer steps.

A.3 Support hotline

The example in the file `support_hotline.xml` models a support hotline using multiple escalation levels of service. Every calling customer is at first redirected to the first support level, afterwards some customers may still need support of the next level. An Or Split node is utilized to model this split.

Since both leaving edges of the node need to have a some sort of service before arriving at the join, a special Skip node was created. The Skip node can handle unlimited arrivals while create no delay at all, so it has no effect on the incoming arrivals.

References

[1] F. Bause, P. Buchholz, and J. May. A Tool Supporting the Analytical Evaluation of Service Level Agreements. In *Proc. of ICPE 2017: International Conference on Performance Engineering, April 22-26, 2017, L'Aquila , Italy*, 2017.

[2] Peter Buchholz and Sebastian Vastag. An introduction to SLA calculus for the analytical validation of SLAs. Technical report, Informatik 4, TU Dortmund, 2015. http://ls4-www.cs.tu-dortmund.de/download/buchholz/sla.pdf.

Index