DALG: The Data Aware Event Log Generator

Manual

Trier University
Department IV - Computer Science
Chair of Business Informatics II

David Jilg
jilg@uni-trier.de
jilg.david@gmail.com
https://github.com/DavidJilg/

Contents

1 Manual **1**

1.1 Introduction . . . 1

1.2 Operating system compatibility . . . 1

1.2.1 Writing and Reading Permissions . . . 2

1.3 Installation . . . 2

1.3.1 Windows . . . 2

1.3.2 Linux . . . 2

1.4 Software Overview and operation . . . 2

1.4.1 Program Overview . . . 2

1.4.2 Program Operation . . . 4

1.4.2.1 Command Line and Terminal Operation . . . 5

1.5 Configuration . . . 5

1.5.1 General Configuration . . . 5

Output Directory . . . 5

Filename . . . 6

Number of Event Logs . . . 6

Include Configuration in Output Directory . . . 6

Include Metadata . . . 6

1.5.2 Simulation Configuration . . . 7

Simulation Mode . . . 7

Trace Name . . . 8

Number of Traces . . . 8

Maximum Number of Traces . . . 8

Minimum Trace Length . . . 8

Maximum Trace Length . . . 8

Maximum Loop iterations (Marking) . . . 8

Maximum Loop iterations (Transitions) . . . 9

Maximum Number of Duplicate Traces . . . 9

Duplicate Detection with Considering Variables . . . 9
Duplicate Detection with Considering Invisible Transitions . . . 9
Include only Valid Ending Traces . . . 9
Include Partial Traces . . . 9
Timestamp Anchor . . . 9
UTC Offset . . . 9
Fixed Timestamp Anchor . . . 10
Merge Intervals if possible . . . 10
Limit Variable Values . . . 10
Perform Trace Estimation . . . 10
1.5.3 Variable Configuration . . . 10
Include Values in Origin Event . . . 10
Use Initial Value . . . 10
Generate Initial Value . . . 11
Initial Value . . . 11
Fixed Variable . . . 11
Trace Variable . . . 11
Minimum Value . . . 11
Maximum Value . . . 11
Which Information is Used for the Value Generation . . . 11
Values . . . 11
Intervals . . . 12
Distribution . . . 12
Include Inverse Intervals . . . 12
Dependencies . . . 13
Self Reference . . . 13
Self Reference Maximum Deviation . . . 14
1.5.4 Transition configuration . . . 14
Include Invisible Transitions in Event Log . . . 14
Include Microseconds in Timestamps . . . 14
Average Transition Time Delay . . . 14
Average Transition Lead Time . . . 14
Activity Name . . . 14
Weight . . . 15
Invisible Transition . . . 15
Use General Configuration . . . 15

No Time Forwarding 15
Time Intervals 15
Add Time Interval Variance 15
Maximum Variance 15
Included Variables 15

Figures

1.1 Screenshot of welcome screen that is shown when you first start the program 2
1.2 Screenshot of the GUI with the distinct parts labelled for easier referencing 3
1.3 Screenshot of the two tabs of the top menu bar 4
1.4 Screenshot of the Terminal and the command line with the command to open DALG in the command line mode 5
1.5 Example of how the trace names are written to the event log 8

1. Manual

1.1 Introduction

Welcome to the manual for DALG: The Data Aware Event Log Generator. David Jilg developed DALG as part of his Master Thesis 'Generating Synthetic Procedural Multi-Perspective Electronic Healthcare Treatment Cases' at the Chair of Business Information Systems II at the University of Trier. DALG allows to generate event logs that conform to the data and the control flow perspective of a provided Data Petri net through token-based simulation. The software is open-source, and access is provided via a Github repository[1]. This document provides information that is needed to operate the software. However, it should be noted that the users are expected to have basic knowledge of process mining and Data Petri nets. The document is structured as follows. First, section 1.2 provides information on the compatibility of this software with different operating systems. Subsequently, section 1.3 will provide information about the installation of the program. Then section 1.4 gives an overview of the software and explains how it is operated. Lastly, section 1.5 provides a detailed explanation of every setting that can be used to configure the generation of the event logs.

1.2 Operating system compatibility

DALG is generally speaking compatible with Windows and Linux-based operating systems. However, the software relies on the QT6.2 framework[2] for the graphical user interface. The following operating systems are the only desktop operating systems officially supported by QT6.2.

1. Windows 10 / 11 / Windows on ARM
2. macOS 10.14, 10.15, 11
3. Generic Linux
4. Ubuntu 20.04
5. SUSE Linux Enterprise Server 15 SP3
6. openSUSE 15.3
7. CentOS Linux 8.2

Due to the limited operating system compatibility of the QT framework, it was decided to only officially support and test DALG for Windows 10, Windows 11, and Ubuntu 20.04. However, just because an operating system is not officially supported does not mean the program can not work on it. It is also possible to run DALG without the graphical user interface. Therefore it should be possible to run it on most Windows and Linux based operating systems by running it from the command line or the terminal. Information on how that can be done is provided in section 1.4.

[1] https://github.com/DavidJilg/DALG
[2] https://www.qt.io/product/qt6

1.2.1 Writing and Reading Permissions

The program needs permission to write and read files from its installation folder and the specified output directory for the program to function. You need to make sure that the program has these permissions. For example, if you install the program in the 'C:/Program Files' folder on Windows 10 you might need to start the program with administrator rights or else it will not work correctly.

1.3 Installation

1.3.1 Windows

The Windows version of DALG is provided as an installer. Therefore the only thing you have to do to install it is to run the installer and follow the provided instructions. The installer will create shortcuts on your desktop that can be used to run the program. Alternatively you can use the portable installation provided for Windows.

1.3.2 Linux

DALG is provided as a portable installation for Linux. Therefore the only thing you have to do to install it is to extract the provided archive to the desired installation path. The 'bin' folder in the installation path contains the 'DALG.run' file that can be used to run the program.

1.4 Software Overview and operation

This section will provide an overview of the program's features and instructions on operating it. It should be noted that all of the screenshots that you will see in the manual are from the Windows 10 version of the program. The Graphical User Interface will look slightly different on a machine using a Ubuntu operating system. However, the function of the program is identical.

1.4.1 Program Overview

When you start the software for the first time, you will be greeted by a welcome screen that can be seen in figure 1.1.

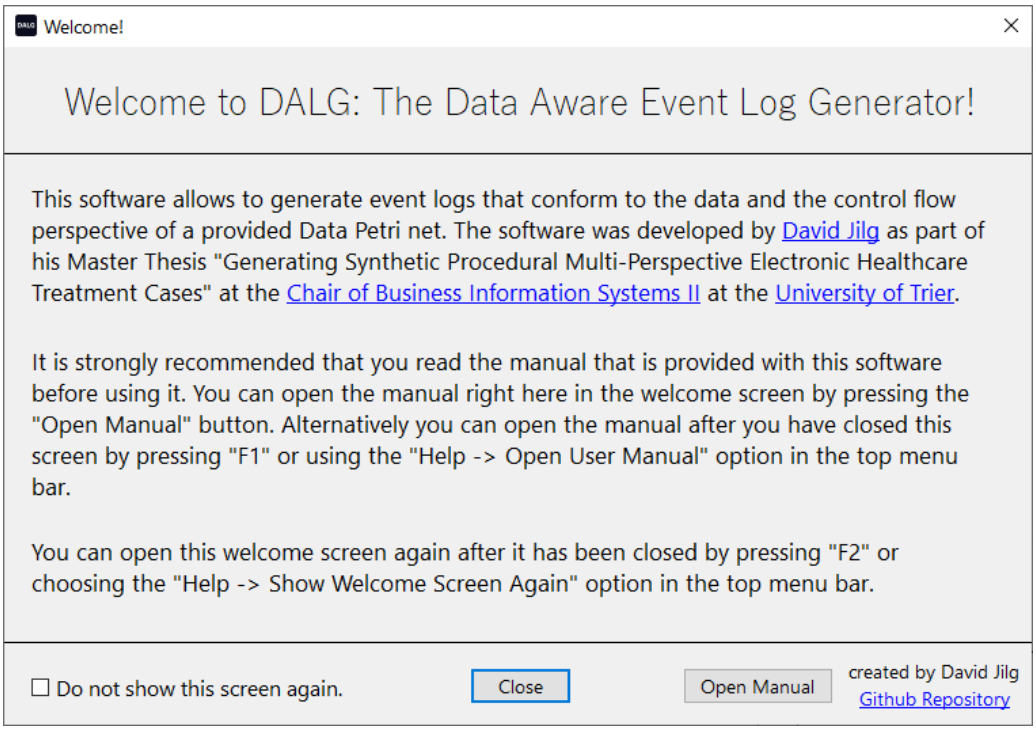

Figure 1.1: Screenshot of welcome screen that is shown when you first start the program

This screen provides some basic information about the software and links to other resources like this manual. If you do not tick the 'Do not show this screen again.' checkbox, this screen will be shown on every startup of the program. The GUI is split into six distinct parts. Figure 1.2 shows these parts, which have been labelled with the letters A through F so that they can be easier referenced in this manual.

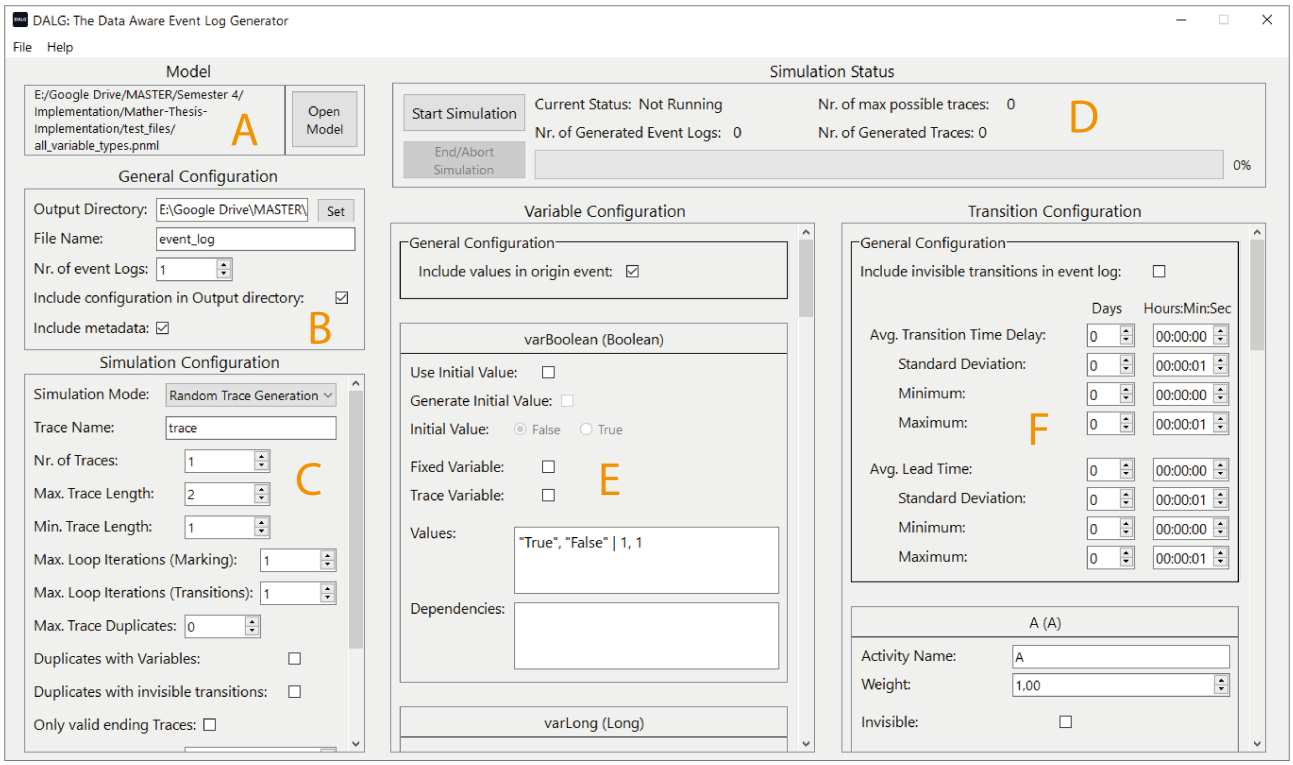

Figure 1.2: Screenshot of the GUI with the distinct parts labelled for easier referencing

In the following, the parts of the GUI will just be called by the letter that they are marked within figure 1.2. Part A is used to display the path of the currently loaded model file. It also contains the 'Open Model' button, which allows you to open a Petri Net model from a Petri Net Markup Language file. Part B contains the settings which allow you to control how the generated event logs are saved. These settings are explained in detail in section 1.5.1. Part C contains all the settings regarding the token-based simulation, except for settings regarding individual variables and transitions. The simulation settings are explained in section 1.5.2. Part D contains GUI elements that allow you to start and stop the simulation and elements that display the current status of the simulation. Part E contains all the simulation settings regarding the variables of the loaded model. When no model is loaded, only the general configuration for the variables is displayed. The settings present in this part are explained din section 1.5.3. Lastly, part F contains all settings regarding the transitions of the loaded model. Just as in part E only the general configuration is shown in part F when no model is loaded. The settings present in this section are explained in section 1.5.4.

In addition to the previously described parts, the GUI also contains a menu bar located at the top left of the program window. The options of the menu bar can be seen in figure 1.3. As seen in the figure, each feature can also be accessed through a shortcut.

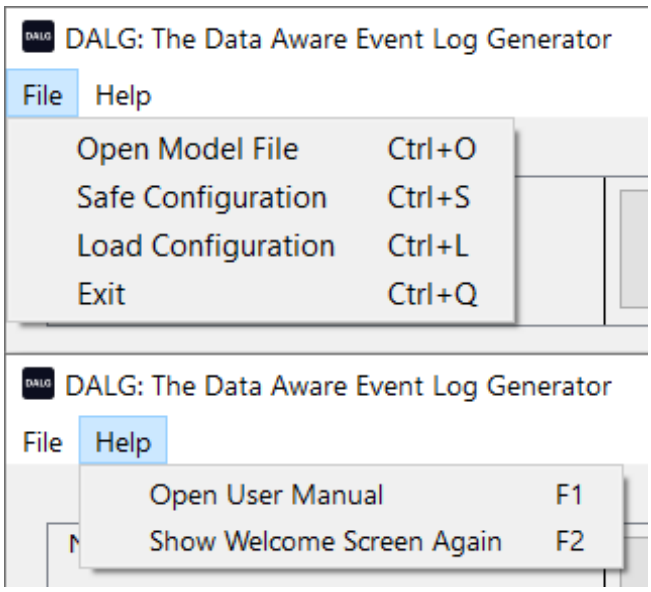

Figure 1.3: Screenshot of the two tabs of the top menu bar

The 'Open Model File' option does the same as the previously described 'Open Model' button. The 'Safe Configuration' and the 'Load Configuration' options allow you to safe/load the current configuration to or from a JSON file. The 'Exit' option closes the program. The 'Help' tab contains an option to open this manual that you are currently reading and an option that lets you see the welcome screen again after it has been closed. It should be noted that the user manual will be opened in the application that is set as the standard for PDF files in your operating system.

1.4.2 Program Operation

This section goes over the steps you need to take to generate event logs from a Petri net model. To start the program on Windows, you simply need to open the 'DALG' shortcut created on the desktop. On Linux, you have to run the RUN file in the installation folder. After starting the program, the first step you need to do is to open a Petri net model. This program only supports models in the Petri Net Markup Language (PNML)[3] (.pnml files). To open a model file, you can either press the 'Open Model' button in the Model section of the GUI or use the menu bar at the top left and choose the 'File -> Open Model' option. You can also access this option by pressing Control + O. Any of those options will open the standard file dialogue of the operating system that you are running on. You can then use that file dialogue to select a PNML file. After you have selected a model file, the program will try to parse this model into the internal model representation. You will see a success message if the model could be loaded successfully. If there were any errors in the model file, an error message will be displayed. After loading a model file, the next step is to configure all the settings to your liking. What settings are available and what these settings do is explained in detail in section 1.5. Once you are happy with the settings, it is time to start the simulation and therefore start generating the event log(s). The simulation can be started by pressing the 'Start simulation' button. The first thing the program does after you have pressed that button is to check the configuration for possible errors. If the configuration is deemed invalid, you will get a warning message asking you if you want to continue anyway. Once the simulation is running, the status will be displayed to the right of the button you used to start the simulation. You can abort the simulation at any time by pressing the 'End/Abort Simulation' button. When you abort the simulation, you will get an option to save the events logs and traces that have been generated so far. Once the simulation finishes, the generated event logs will be saved to the specified output directory, and a success message will be displayed.

[3] https://www.pnml.org/

1.4.2.1 Command Line and Terminal Operation

As previously mentioned, DALG can also be controlled through the Command Line on Windows or the Terminal on Ubuntu. However, this form of controlling the program is not the intended way, and some comfort features are missing. To start the program in this mode, you need to pass a path to a configuration file as an argument. Figure 1.4 shows an example of opening the program in this way on Windows and Ubuntu. (*Note: On Windows, you have to open a particular Exe file in order to run the command-line version of the program. The file is named 'DALG_CMD.exe', and it is present in the installation path.*)

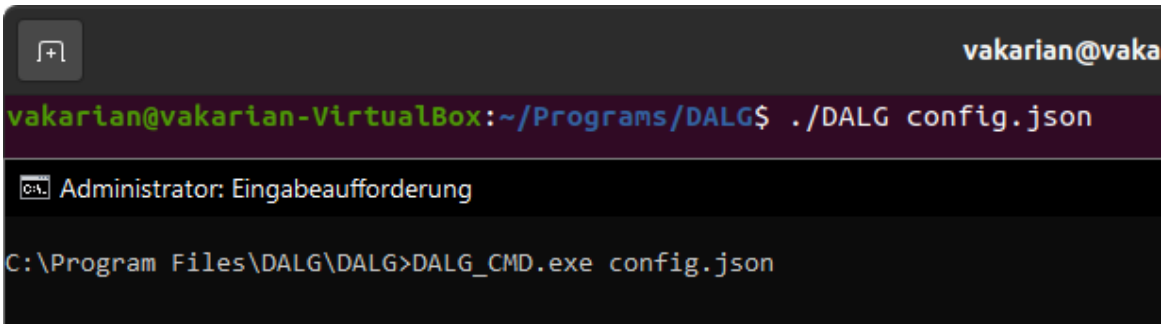

Figure 1.4: Screenshot of the Terminal and the command line with the command to open DALG in the command line mode

After the program has been started through the command line, it immediately loads the model from the path specified in the configuration file. Then it tries to parse the settings in the configuration file and starts the simulation. During the simulation, the current status will be printed in the console. You can abort the simulation at any time with a Keyboard Interrupt. When stopping the simulation, the event logs/traces generated so far are written to the output directory. When the simulation finishes, the generated event logs are written to the output directory specified in the configuration file.

When you use the program without the GUI, you will have to create the configuration file by yourself. For this purpose an example is provided in the `'installationpath'/documentation/examples` folder. This example contains a Data Petri net, a configuration file, and an event log created using the model and configuration. You can use this example configuration file to get to know the structure that the configuration files need to have. It should be noted that all the date values in the configuration files need to be converted to seconds from epoch time (i.e., seconds since 01.01.1970).

1.5 Configuration

This section will describe all the available configuration options for configuring the generation of event logs. The settings are grouped by the different GUI parts they are grouped by. The configuration areas are labelled with the letters B, C, E, and F in figure 1.2.

1.5.1 General Configuration

This section will describe all the settings in the 'General Configuration' area, which is marked with the letter B in figure 1.2.

Output Directory

The output directory setting allows you to set the directory to which the created event logs will be saved. You can either enter the path manual by typing into the text field or you can press the 'Set' button, which opens the standard Ubuntu or Windows file dialogue. If you enter the path to the output directory manually, it has to be in the same format as the following example: '`C:/Users/david/Documents`'

Filename

The filename setting allows you to specify the name of the created '.xes' files. Therefore the text you enter has to be a valid filename in the operating system that you use. For example, the character '/' is not allowed to be in a filename in Windows and Linux. Additionally, it is not necessary to enter the file extension since this program can only create files in the EXtensible Event Stream[4] (.xes) format. If multiple event logs are generated, the filename will be expanded with a number. For example, if you specify the filename 'event_log' and generate 3 event logs the created files will have the names 'event_log1', 'event_log2', and 'event_log3'.

Number of Event Logs

This setting allows you to control how many event logs are created with the specified simulation settings. These event logs will all be different since the seed you enter is only set before creating the first event log. The (pseudo) random number generator is then continuously used for all following event logs.

Include Configuration in Output Directory

This setting allows you to control whether the configuration used to generate event logs is saved as a file in the output directory. If this setting is activated, the current configuration will be saved as the following file in the same directory as the generated event logs: '{filename}_config.json'. This setting is only available when using the graphical user interface and not when the program is used through the command line or the terminal.

Include Metadata

This setting allows you to control whether some basic metadata will be added to the event log. If this is the case, the XES extensions 'MetaData General'[5] and 'MetaData Concept'[6] will be used. The following enumeration list the metadata attributes that will be added.

1. **General metadata**

   Total number of traces

   Total number of events

   Average number of events

   Minimum number of events

   Maximum number of events

   Number of events standard deviation

2. **Metadata Concept**

   Number of different events

   Average number of different events

   Number of different events standard deviation

   Minimum number of different events

   Maximum number of different event

   Total number of named events

   Average number of named events

   Minimum number of named events

   Maximum number of named events

   Number of named events standard deviation

[4] http://www.xes-standard.org/
[5] http://www.xes-standard.org/meta_general.xesext
[6] http://www.xes-standard.org/meta_concept.xesext

1.5.2 Simulation Configuration

This section will describe all the settings in the 'Simulation Configuration' area, which is marked with the letter C in figure 1.2. The available settings differ depending on the simulation mode that is selected. Therefore you will not see all of the following configuration options at all times.

Simulation Mode

The 'Simulation Mode' setting is the setting that has the biggest influence on the simulation. There are three options to choose from, which will be explained in the following paragraphs.

The first simulation mode is called '**Random Trace Generation**'. This is the mode that you should use in most cases. As the name suggests, the simulation continuously generates random traces in this mode until the specified number of traces is reached. The traces are generated by randomly firing one of the enabled transitions in the model until a final marking is reached, a deadlock is reached, or the maximum trace length is exceeded. It should be noted that the simulation will only properly finish if the number of specified traces can be generated from the model. For example, if you specify to generate ten unique traces from a model with only five unique traces, the simulation will end up in an endless loop trying to find more traces. However, this is not a severe problem since the simulation can be ended by pressing the 'End/Abort Simulation' button. In the case of an abort, you will be given the opportunity to save the Event logs/traces generated so far.

The second simulation mode is the '**Random Exploration**' mode. This simulation mode aims to generate as many unique traces as possible and, therefore, explores the Petri net. The exploration is done by randomly generating traces and only saving traces that have not been generated before. This simulation mode has itself two different modes of operations depending on whether the 'Perform Trace Estimation' setting is activated. If this setting is activated, the maximum number of possible traces will be calculated by fully exploring the Petri net while not considering the data perspective. The data perspective can not add any traces to the model. It can only restrict the number of traces with its guard conditions. Therefore the maximum number of traces can be calculated by ignoring the data perspective (but including invisible transitions). If a trace estimation is performed, the simulation might finish on its own if the number of traces that are possible when ignoring the data perspective equals the number of traces possible when considering the data perspective. If no trace estimation is performed or if not all traces of the estimation are possible with the data perspective, the simulation will not finish on its own. In this case, it will continuously try to find new unique traces that have not been found so far until you tell it to stop. However, as previously mentioned, this is not a problem since the simulation can be ended by pressing the 'End/Abort Simulation' button. In the case of an abort, you will be given the opportunity to save the Event logs/traces generated so far.

The third simulation mode is the '**All Traces**'-mode. As the name suggests, this mode tries to find all possible traces of a given Petri net systematically. It should be noted that this simulation mode is a **experimental** mode that does not work for every Petri net. The mode cannot correctly deal with variables that get written to multiple times during a single trace. Therefore if the model contains loops or transitions that fire more than once, this method might return traces that are not valid. It is advised to use this mode with caution. In the following, it is explained how this mode works. Subsequently, advice is given on how to use this mode to achieve the best result.
The first step that is done in this mode is to do a full exploration of the Petri net while only considering the control-flow perspective. However, this full exploration will stop if the specified number of maximum traces is reached. This exploration returns a list of traces that only consists of a sequence of fired transitions and no variable values. The program will then iterate over all found traces and tries to find a variable value combination for all write operations in the trace that satisfies all the guard conditions. Depending on the number of possible combinations, this can take a very long time since it is basically brute forcing valid combinations. Since Data Petri nets can contain variables that are not discrete (e.g., double, long), the number of combinations is practically infinite. Therefore, all variables are transformed into discrete variables by dividing their possible values into intervals. The

intervals are determined based on the values necessary for satisfying guard conditions. However, this can still take a very long time for Petri nets with many variables or many possible values for variables. Therefore this simulation mode should only be used for Petri nets with relatively few variables.

Trace Name

This setting allows you to specify the name of the generated event logs' traces. If more than one trace is generated, the trace name will be expanded with a number just like described previously for the filename of the event logs. The way the trace name will be written in the event log can be seen in figure 1.5

```
<trace>
    <string key="concept:name" value="trace1"/>
    [...]
</trace>
```

Figure 1.5: Example of how the trace names are written to the event log

Number of Traces

This setting is only available when using the 'Random Trace Generation' simulation mode. It determines how many traces the program will try to generate. For the simulation to end on its own, the number of traces needs to be lower or equal to the number of traces that are possible with the current setting. However, like previously mentioned in section 1.4 the simulation can be stopped at any time, and the event logs and traces that have been generated so far can be saved. Therefore it is not a problem when this setting is set too high.

Maximum Number of Traces

This setting is only available in the 'Random Exploration' and 'All Traces' simulation modes. It acts as a limit for the amount of traces that are generated. If the number of traces possible with the current settings exceeds this threshold, the exploration is stopped when this number is reached.

Minimum Trace Length

This setting allows you to set a minimum length for all generated traces. Only traces equal to or exceeding this threshold will be added to the event log. The length of a trace is determined by the number of events it contains. Therefore it is equal to the number of non-invisible transitions that are fired. However, if the setting 'Include invisible transitions in the event log' is activated, the trace length will be determined by all transitions that have been fired.

Maximum Trace Length

This setting allows you to set a maximum length for all generated traces. The way this setting works is analogous to the previously described 'Minimum Trace Length' setting.

Maximum Loop iterations (Marking)

This setting is one of the two settings letting you control the number of loop iterations for every trace. The function that relies on this setting uses the marking of the model to detect loops. Every marking that is seen in a given trace is saved. This allows the program to detect if the current marking of the model has been visited before. Therefore loops that return to a common marking can be detected. However, loops that do not return to a common marking can not be controlled with this setting. It should be noted that the name of this setting is slightly misleading. If this setting is set to one, every marking is only allowed to appear once in any given trace. Therefore the amount of loop iterations allowed is 0.

Maximum Loop iterations (Transitions)

This is the second setting, allowing you to control the number of loop iterations. The function that relies on this setting detects loops by counting the number of times each transition has been fired. Therefore it can detect loops that do not return to a common marking. However, this way of detecting loops can lead to false-positive results during loop detection. A transition that was fired more than once does not necessarily mean that a loop is present in a model. Therefore this setting should be used with caution. It should be noted that the name of this setting is slightly misleading. If this setting is set to one, every transition is only allowed to appear once in any given trace. Therefore the amount of loop iterations allowed is 0.

Maximum Number of Duplicate Traces

This setting allows you to control how many duplicate traces are allowed to appear in the event log. It is only available in the 'Random Trace Generation' mode since the other two modes try to find as many different traces as possible. Which paths are considered identical is determined by the two following setting options.

Duplicate Detection with Considering Variables

This setting allows you to control whether the values of variables should be considered when determining if two traces are identical. If this setting is not activated, only the sequence of fired transitions is considered to determine duplicate traces. It should be noted that if this setting is activated the trace estimation in the 'Random Exploration' simulation mode can not be performed.

Duplicate Detection with Considering Invisible Transitions

This setting allows you to control whether invisible transitions should be considered when determining if two traces are identical. If this setting is not activated, invisible transitions are ignored while determining if two traces are identical.

Include only Valid Ending Traces

This setting allows you to control whether only traces reaching a valid final should be added to the generated event log. The valid final markings are read from the loaded model file. If this setting is not activated, traces that reach a deadlock are also included in the event log.

Include Partial Traces

This setting allows you to control whether the partial traces of generated traces are included in the event log. For example, if the trace 'A, B, C' is generated, the partial traces would be 'A' and 'A, B'. It should be noted that the partial traces are still subjected to all the other restrictions like the minimum trace length.

Timestamp Anchor

This setting allows you to set the timestamp used as a starting time for the simulation. It should be noted that this timestamp is treated as the Universal Standard Time with no offset since the offset can be set with the following setting.

UTC Offset

This setting allows you to set the UTC offset for the timestamp anchor. Therefore this setting allows you to set the timezone of the simulation time.

Fixed Timestamp Anchor

This setting allows you to control whether all traces should start with the same timestamp or whether the traces should follow each other. If this setting is not activated, the simulation time will not be reset between the generation of traces. In this case, the trace's starting time is the previous trace's ending time. Therefore, if you want the traces to run in parallel, you should activate this setting, and if you want only one trace running at any time, you should deactivate this setting.

Merge Intervals if possible

Note: This setting affects a portion of the program that has so far not been explained in the manual and therefore may not be fully understandable yet. The interval setting for variables is explained in section 1.5.3

This setting is exclusive to the 'All Traces' simulation mode. As previously mentioned, this simulation mode tries all possible variable value combinations to find a valid assignment for every possible trace. Therefore the fewer variable values exist, the less computation is needed to find all traces. Therefore this setting allows you to reduce the number of intervals that must be tried for every numeric variable by merging intervals. For example, the interval 'variable > 5' and 'variable > 10' can be both represented by the single interval 'variable > 10'. More on intervals for variable values can be read in section 1.5.3.

Limit Variable Values

This setting is also exclusive to the 'All Traces' simulation mode. It also has the goal to reduce variable value combinations. Therefore this setting allows you to reduce the possible variable values to values needed to satisfy guard transitions. If this setting is activated, the user's input for the possible values of variables is ignored, and the guard conditions are analysed for possible values.

Perform Trace Estimation

This setting is exclusive to the 'Random Exploration' simulation mode. It allows to control whether a trace estimation is performed before the simulation. The only purpose of the trace estimation is to provide an idea of the maximum number of different traces possible for the given model. This setting can help the user to decide when to abort a simulation. Additional, it can even lead to the simulation finishing on its own if all traces have been found. However, it should be noted that the trace estimation can take a very long time if the Petri net is very complex since the number of traces can be exponential to the number of transitions. Therefore it is advised only to use this setting for Petri nets of moderate sizes. Additionally, if this setting is activated, the 'Duplicates with Variables' setting can not be activated since the trace estimation ignores the data perspective.

1.5.3 Variable Configuration

This section will describe all the settings in the 'Variable Configuration' area, which is marked with the Letter E in figure 1.2.

Include Values in Origin Event

This setting allows you to control in which event variable values that are written by a transition first appear. If this setting is activated, the values written by a transition will already appear in the corresponding event. If this setting is not activated, the values will first appear in the following event.

Use Initial Value

This setting allows you to control whether a variable has an initial value. If this setting is activated, the variable already has a value event before any transition has written to it.

Generate Initial Value

This setting allows you to control whether the initial value is generated. If this setting is not activated, the initial value that the user provides is used. If this setting is activated, the initial value will be randomly generated using the provided semantic information (For example, by using a provided distribution).

Initial Value

This setting allows you to supply an initial value used for the variable.

Fixed Variable

This setting allows you to choose whether a variable should have a fixed value. If this setting is activated, the first value written to the variable will not change. Even if another transition writes to it, the value will remain unchanged. If the variable has an initial value, the initial value will be the only value that the variable will have.

Trace Variable

This setting allows you to specify that a variable is a trace variable. If this setting is activated, the variable will not be part of any events, but it will be part of the trace itself. In most cases, it makes sense to also activate the 'Fixed Variable' setting for all trace variables.

Minimum Value

This setting is only available for numeric or date variables. It allows you to set a minimum value for the variable that will be used during the generation of values for that variable.

Maximum Value

This setting is only available for numeric or date variables. It allows you to set a maximum value for the variable that will be used during the generation of values for that variable.

Which Information is Used for the Value Generation

This setting allows you to specify which information is used to generate variable values needed if a transition writes to a variable. This setting is only available for numeric and date variables. There are three options to choose from. First, you can specify to use the values you have provided in the values field. Secondly, you can specify intervals from which the values are generated. The third option is to specify the parameters of a distribution, which is then used to generate the values. The following three paragraphs will cover these options.

Values

As mentioned in the previous paragraph, this input setting allows you to specify values that are used when a transition writes to a variable. However, these values are only used when you have specified to use the values through the setting described in the last paragraph. If this is the case, a random value of the specified values will be chosen every time a transition writes to it. To influence the random decision, you can provide weights for all the specified values. In the following, the syntax for the specification of values is explained.

The syntax consists of values encapsulated in double quotation marks and separated by commas.

"value1", "value2", "value3" ...

It should be noted that all type of values need to be provided in quotation marks and not only values for string variables. The type of value , e.g., integer or date, will be inferred from the variable type. For date variables, the following format must be used 'yyyy-MM-ddThh:mm:ss'. As mentioned before, it is possible to supply weights for each value that you specify. If you supply weights, the number of values must equal the number of weights. The following syntax is used to specify weights.

"value1", "value2", "value3" | 1, 1, 3.5

In the above example, the first two values have the same chance of being chosen, and the third value has a 3.5 times higher chance of being chosen. The weight values can thereby be any positive numbers. However, the sum of the weights must be greater than 0.

You may have noticed that there can already be values in the text field when you load a model. This is due to a program feature that tries to make it easier for the user to input the information. When a model is loaded the guard transitions of the model are analysed to find possible values and intervals for the variables. For example if a model has the transition guard 'variable1 == "value" ' it can be inferred that 'value' is a valid value for variable1.

Intervals

As previously mentioned, it is also possible to specify intervals for the variable value generation. Suppose a transition writes to a variable and you have specified to use the intervals through the 'Used information' setting. In that case, the intervals will be used to generate the value of the write operation. If this is the case, a random interval will be chosen, and a value will be generated from a uniform distribution. The following syntax is used to specify intervals. Each interval consists of an operator and an upper or lower boundary depending on the operator.

(">"; "1"),("<="; "1.0"), ...

In the above example, the first interval is specified between the variable's maximum and one, while one is excluded. The second interval is set to between the minimum of the variable and 1.0 while including the upper boundary. The operators that are allowed for the interval specifications are '<', '>', '<=', and '>='. For date variables the following format must be used for the boundary: 'yyyy-MM-ddThh:mm:ss'

You may have noticed that there can already be intervals in the text field when you load a model. This is due to a program feature that tries to make it easier for the user to input the information. When a model is loaded, the guard transitions of the model are analysed to find possible values and intervals for the variables. For example if a model has the transition guard 'variable1 > 5' it can be inferred that (">"; "5") is a valid interval for variable1.

Distribution

As previously mentioned in the paragraph regarding the 'Used Information' setting, the third option for specifying how to generate variable values for numeric and date variables is to specify a distribution and its parameters. There are three types of distributions, namely *Uniform*, *Normal*, and *Exponential*, that can be chosen. The distributions will be truncated to fit the variable's defined minimum and maximum values. If you choose to use a Normal (i.e., Gaussian) distribution, you must specify the mean and standard deviation. If you choose the Exponential distribution, you only have to specify the standard deviation. It should be noted that the standard deviation is not allowed to be 0.

Include Inverse Intervals

This setting allows you to control whether the inverse intervals of the intervals you specified will be included when generation values for the variable. For example, the inverse interval for (">"; "5") would be ("<="; "5").

Dependencies

This setting allows you to specify dependencies between variables that are considered when generating values for the variables. These decencies can be a helpful tool to increase the realism of the generated event logs. The following example illustrates this usefulness. Let us imagine a model that describes the treatment process of cancer patients in a hospital. The model has a variable for the type of cancer that the patient has. This variable is written to by a transition that resembles the patient's diagnosis. Let us additionally imagine that the model also has a variable that specifies the gender of the patient. Suppose the gender of the patient is 'female'. In that case, it is not realistic that the diagnosis transition would write the value 'prostate cancer' to the variable specifying the type of cancer that the patient has. This sort of behaviour can be prevented by using dependencies. The specification of the dependencies has to be done using the following syntax:

"logical expression" => "constraint", ...

The logical expression must follow the same syntax as the guard transitions in a PNML model. In fact, the method used to evaluate the guard conditions is the same method used to evaluate the logical expressions of the specified dependencies. The constraints of dependencies have to adhere to the following syntax:

" 'operator' ; 'value' "

The operators that are allowed to appear in the constraints are '<', '<=', '>', '>=', '==', and '! ='. The following example for a dependency specification for the variable 'cancer type' tackles the previously described problem regarding the gender of patients:

"gender == 'female' " => " '!=' ; 'prostate cancer' "

The above example states that the value of the variable 'cancer type' is not allowed to be 'prostate cancer' when the patient is a woman.

Self Reference In addition to the previously described dependencies, the dependency input can also be used to access the 'Self Reference' feature. This feature is only available for numeric and date variables. It allows you to have a variable that changes with regard to itself every time it is written to. Let us consider an example to explain this further.

"SELF_REFERENCE" => " '+' ; '5' "

The example above shows which syntax must be used to input a self-reference. In this example, the variable increases by five every time the variable is written to. However, it should be noted that this self-reference only affects the variable if it already has a value. The operators that are allowed for this feature are '+', '-', '/' (Division), and '*' (Multiplication). For date variables, the value that is used in this feature needs to be input in the following format: 'dd:hh:mm:ss'.

In addition to the previously mentioned allowed operators the self reference feature can also be used with the '==' operator. This allows the variable to be equal to other variables. Lets consider the following example.

"SELF_REFERENCE" => " '==' ; 'gender' "

In this example the variable that this dependency is used on would be equal to the variable 'gender'. Additionally the variable can be equal to the last transition that writes to it by using the following syntax.

“SELF_REFERENCE” => “ ‘==’ ; ‘_EVENT_NAME_’ ”

In this case the variable would be equal to the activity name of the last transition that wrote to it.

Self Reference Maximum Deviation

This setting affects the previously described ‘Self Reference’ feature. It allows you to add variation to the change of variables. Again, consider the example previously used to explain the self-reference feature.

“SELF_REFERENCE” => “ ‘+’ ; ‘5’ ”

In this example, the variable increases by five at every write operation. This feature allows you to have some deviation from the fixed value. For example, if you set the maximum deviation to be 2, the variable will be increased by a value from the interval [3, 7]. A random value with a uniform distribution will be generated from that range. If you do not want any deviation from the specified value, you can simply leave this setting at 0.

1.5.4 Transition configuration

This section will describe all the settings in the ‘Transition Configuration’ area, which is marked with the letter F in figure 1.2.

Include Invisible Transitions in Event Log

This setting allows you to control whether the event log will contain events that correspond to invisible transitions. Each transition firing will create an event in the event log if this setting is activated. If this setting is not activated, only the firing of not invisible transitions will create an event in the log.

Include Microseconds in Timestamps

This setting allows you to control whether the timestamps of events include microseconds precision. If this setting is not checked the timestamps will be rounded to the nearest second.

Average Transition Time Delay

This setting allows you to control the delay before a transition is fired. Together with the ‘Average Transition Lead Time’ setting, this setting, therefore, controls the amount of time between two events. The time values will be calculated using a normal distribution with the provided mean, standard deviation, minimum and maximum value. The software provides two options to configure this setting. You can either set global values for this setting, and you can set values for individual transitions.

Average Transition Lead Time

This setting allows you to control the time a transition takes to execute. Together with the previously described setting this setting therefore controls the amount of time between two events. The time values will thereby be calculated by using a normal distribution with the provided mean, standard deviation, minimum and maximum value. The software provides two options to configure this setting. You can set global values for this setting, and you can set values for individual transitions.

Activity Name

This setting allows you to specify what name the event corresponding to this transition should have. Every time a transition is fired, an event is created in the event log. The name of the event represents the activity that triggered this event.

Weight

This setting allows you to influence the random firing of transitions in the two random based simulation modes. A random transition will be chosen and fired if multiple transitions are enabled. This setting allows influence this decision by using a weighted random decision. When a decision needs to be made, all weights for the enabled transitions will be normalised to calculate the percentage chance each transition has to be chosen. For example, if a transition has a weight of five, it has a five times greater chance of being chosen than a transition with 1.

Invisible Transition

This setting allows you to set transitions to be invisible. If this setting is activated the transition will be treated as invisible meaning it will not appear in the event log but it can influence the flow of the model through the data perspective.

Use General Configuration

This setting allows you to enter individual time settings for every transition. If this setting is activated the general configuration that you can edit at the top of the 'Transition Configuration' area is used. If this setting is not activated, you can enter individual settings.

No Time Forwarding

This setting allows you to control whether a firing of the transition forwards the time of the simulation clock. If this setting is checked the transition will have no time delay and a lead time of 0.

Time Intervals

The Time Intervals input allows you to restrict when an activity can be performed. Therefore, a list of time intervals, in which the activity can occur, can be provided. If not Intervals are provided, no restrictions apply. The following syntax has to be used to specify intervals:

Mon,Tue,Wed,Thu,Fri,Sat,Sun | 07:00:00-12:00:00; Mon,Tue,Wed,Thu,Fri | 14:00:00-18:00:00; .....

In the above example the activity can be performed on every day of the week between 07:00 am and 12:00 pm (use the 24 hour time format to specify timestamps). Additionally the activity can be performed Monday through Friday between 02:00 pm and 06:00 pm.

Add Time Interval Variance

This setting allows you to add variance to the timestamps of activities that had to be adjusted to fit into an interval. If the original timestamp calculated for an activity is outside of all time intervals, it gets forwarded to the next available interval. This can result in many events occurring on the start of an interval. This setting allows you to choose whether some variance should be added so that the event do not all start at the beginning of an interval.

Maximum Variance

This setting allows you to set the maximum variance for the previously explained setting. The provided value specifies the maximum amount of variance in minutes.

Included Variables

This setting allows you to control which variables are included in the events created from firing a transition. A variable is included in the event if it has a value, is not a trace variable, and the checkbox next to the variable is checked.