Ebi - a stochastic process mining tool

0.1.0

Manual

The BPM group at RWTH Aachen University
Sander J.J. Leemans, Tian Li

August 27, 2024

Contents

1 Commands

1.1 `Ebi analyse all-traces`

Alias: `Ebi ana all`.
Find all traces. Models containing loops are not supported.

Parameter	
<FILE>	The file with an object that has deterministic stochastic semantics. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* stochastic deterministic finite automaton (.sdfa), compressed event log (.xes.gz), stochastic labelled Petri net (.slpn), event log (.xes) and finite stochastic language (.slang).
`-o` or `--output` <FILE>	The finite stochastic language (.slang) file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
`-a` or `--approximate`	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: finite stochastic language.

1.2 Ebi analyse completeness

Alias: `Ebi ana comp`.
Estimate the completeness of a finite language.
More information: [3].

Parameter	
<FILE>	The event log. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* compressed event log (.xes.gz) and event log (.xes).
-o or --output <FILE>	The fraction file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
-a or --approximate	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: fraction.

1.3 Ebi analyse medoid

Alias: `Ebi ana med`.
Find the traces with the lowest average normalised Levenshtein distance to the other traces; ties are resolved arbritrarily.

Parameter	
<FILE>	The finite stochastic language. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* finite stochastic language (.slang), compressed event log (.xes.gz) and event log (.xes).
<NUMBER_OF_TRACES>	The number of traces that should be extracted. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* integer.
-o or --output <FILE>	The finite language (.lang) file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
-a or --approximate	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: finite language.

1.4 Ebi analyse minimum-probability-traces

Alias: `Ebi ana minprob`.
Find all traces that have a given minimum probability. Please be aware of models containing livelocks: these may cause the computation to never finish. Will return an error if there are no such traces.

Parameter	
<FILE>	The file with an object that has deterministic stochastic semantics. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* stochastic labelled Petri net (.slpn), compressed event log (.xes.gz), event log (.xes), stochastic deterministic finite automaton (.sdfa) and finite stochastic language (.slang).
<MINIMUM_PROBABILITY>	The minimum probability that a trace should have to be included. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* fraction.
`-o` or `--output` <FILE>	The finite stochastic language (.slang) file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
`-a` or `--approximate`	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: finite stochastic language.

1.5 Ebi analyse mode

Find the trace with the highest probability; ties are resolved arbritrarily. Equivalent to 'Ebi evaluate mostlikely 1'.

Parameter	
<FILE>	The file with an object that has deterministic stochastic semantics. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* stochastic labelled Petri net (.slpn), compressed event log (.xes.gz), finite stochastic language (.slang), stochastic deterministic finite automaton (.sdfa) and event log (.xes).
`-o` or `--output` <FILE>	The finite stochastic language (.slang) file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no

`-a` or `--approximate`	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: finite stochastic language.

1.6 Ebi analyse most-likely-traces

Alias: `Ebi ana mostlikely`.
Find the traces with the highest probabilities; ties are resolved arbitrarily. Please be aware of models containing livelocks: these may cause the computation to never finish.

Parameter	
<FILE>	The file with an object that has deterministic stochastic semantics. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* finite stochastic language (.slang), compressed event log (.xes.gz), stochastic labelled Petri net (.slpn), stochastic deterministic finite automaton (.sdfa) and event log (.xes).
<NUMBER_OF_TRACES>	The number of traces that should be extracted. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* integer.
`-o` or `--output` <FILE>	The finite stochastic language (.slang) file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
`-a` or `--approximate`	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: finite stochastic language.

1.7 Ebi analyse-non-stochastic cluster

Alias: `Ebi anans clus`.
Apply k-medoid clustering: group the traces into a given number of clusters, such that the average distance of each trace to its closest medoid is minimal. The computation is random and does not take into account how often each trace occurs.
More information: [7].

Parameter	
<FILE>	The finite stochastic language. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line.

	Accepted values: event log (.xes), finite stochastic language (.slang), compressed event log (.xes.gz) and finite language (.lang).
<NUMBER_OF_CLUSTERS>	The number of clusters. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* integer.
-o or --output <FILE>	The finite language (.lang) file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
-a or --approximate	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: finite language.

1.8 Ebi analyse-non-stochastic medoid

Alias: `Ebi anans med`.
Find the traces with the lowest average normalised Levenshtein distance to the other traces; ties are resolved arbritrarily. The computation is random and does not take into account how often each trace occurs.

Parameter	
<FILE>	The finite stochastic language. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* event log (.xes), finite language (.lang), compressed event log (.xes.gz) and finite stochastic language (.slang).
<NUMBER_OF_TRACES>	The number of traces that should be extracted. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* integer.
-o or --output <FILE>	The finite language (.lang) file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
-a or --approximate	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: finite language.

1.9 Ebi association all-trace-attributes

Alias: `Ebi asso atts`.
Compute the association between the process and trace attributes; 500 samples are taken.
More information: [5].

Parameter	
<FILE>	The event log for which association is to be computed. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* compressed event log (.xes.gz) and event log (.xes).
-s or –number-of-samples	Take a number of samples. *Mandatory:* no
-o or --output <FILE>	The text file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
-a or --approximate	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: text.

1.10 Ebi association trace-attribute

Alias: Ebi asso att.
Compute the association between the process and a given trace attribute; 500 samples are taken.
More information: [5].

Parameter	
<FILE>	The event log for which association is to be computed. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* compressed event log (.xes.gz) and event log (.xes).
<ATTRIBUTE>	The trace attribute for which association is to be computed. The trace attributes of a log can be found using 'Ebi info'. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* text.
-s or –number-of-samples	Take a number of samples. *Mandatory:* no
-o or --output <FILE>	The root file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
-a or --approximate	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: root.

1.11 Ebi conformance entropic-relevance

Alias: `Ebi conf er`.
Compute entropic relevance (uniform).
More information: Section efsec:er.

Parameter	
<FILE_1>	A finite stochastic language (log) to compare. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* finite stochastic language (.slang), event log (.xes) and compressed event log (.xes.gz).
<FILE_2>	A queriable stochastic language (model) to compare. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* finite stochastic language (.slang), event log (.xes), stochastic deterministic finite automaton (.sdfa), stochastic labelled Petri net (.slpn) and compressed event log (.xes.gz).
`-o` or `--output` <FILE>	The logarithm file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
`-a` or `--approximate`	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: logarithm.

1.12 Ebi conformance jensen-shannon

Alias: `Ebi conf jssc`.
Compute Jensen-Shannon stochastic conformance.

Parameter	
<FILE_1>	A finite stochastic language to compare. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* event log (.xes), finite stochastic language (.slang) and compressed event log (.xes.gz).
<FILE_2>	A queriable stochastic language to compare. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* compressed event log (.xes.gz), finite stochastic language (.slang), stochastic labelled Petri net (.slpn), stochastic deterministic finite automaton (.sdfa) and event log (.xes).

`-o` or `--output` <FILE>	The rootlog file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no

Output: rootlog.

1.13 Ebi conformance jensen-shannon-sample

Alias: `Ebi conf jssc-sample`.
Compute Jensen-Shannon stochastic conformance with sampling.

Parameter	
<FILE_1>	A queriable stochastic language to compare. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* compressed event log (.xes.gz), finite stochastic language (.slang), stochastic deterministic finite automaton (.sdfa), stochastic labelled Petri net (.slpn) and event log (.xes).
<FILE_2>	A queriable stochastic language to compare. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* stochastic labelled Petri net (.slpn), compressed event log (.xes.gz), finite stochastic language (.slang), stochastic deterministic finite automaton (.sdfa) and event log (.xes).
<NUMBER_OF_TRACES>	Number of traces to sample. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* integer.
`-o` or `--output` <FILE>	The rootlog file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no

Output: rootlog.

1.14 Ebi conformance unit-earth-movers-stochastic-conformance

Alias: `Ebi conf uemsc`.
Compute unit-earth movers' stochastic conformance.
More information: [6].

Parameter	
<FILE_1>	A finite stochastic language (log) to compare.

	Mandatory: yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* finite stochastic language (.slang), compressed event log (.xes.gz) and event log (.xes).
<FILE_2>	A queriable stochastic language (model) to compare. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* event log (.xes), finite stochastic language (.slang), stochastic deterministic finite automaton (.sdfa), compressed event log (.xes.gz) and stochastic labelled Petri net (.slpn).
-o or --output <FILE>	The fraction file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
-a or --approximate	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: fraction.

1.15 Ebi convert finite-stochastic-language

Alias: Ebi conv slang.
Convert an object to a finite stochastic language.

Parameter	
<FILE>	Any file supported by Ebi that can be converted. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* finite stochastic language (.slang), compressed event log (.xes.gz) and event log (.xes).
-o or --output <FILE>	The finite stochastic language (.slang) file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
-a or --approximate	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: finite stochastic language.

1.16 Ebi convert labelled-Petri-net

Alias: Ebi conv lpn.
Convert an object to a labelled Petri net.

Parameter	
<FILE>	Any file supported by Ebi that can be converted. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* Petri net markup language (.pnml), stochastic deterministic finite automaton (.sdfa), labelled Petri net (.lpn), stochastic labelled Petri net (.slpn) and directly follows model (.dfm).
-o or --output <FILE>	The file to which the results must be written. Based on the file extension, Ebi will output either a labelled Petri net (.lpn) or a Petri net markup language (.pnml). If the parameter is not given, the results will be written to STDOUT as a labelled Petri net (.lpn). *Mandatory:* no
-a or --approximate	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: labelled Petri net.

1.17 Ebi convert stochastic-finite-deterministic-automaton

Alias: Ebi conv sdfa.
Convert an object to a finite stochastic language.

Parameter	
<FILE>	Any file supported by Ebi that can be converted. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* finite stochastic language (.slang), stochastic deterministic finite automaton (.sdfa), event log (.xes) and compressed event log (.xes.gz).
-o or --output <FILE>	The stochastic deterministic finite automaton (.sdfa) file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
-a or --approximate	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: stochastic deterministic finite automaton.

1.18 Ebi discover occurrence

Alias: Ebi disc occ.
Give each transition a weight that matches the occurrences of its label; silent transitions get a weight of 1.

More information: [2].

Parameter	
`<FILE_1>`	A finite stochastic language (log) to get the occurrences from. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* compressed event log (.xes.gz), event log (.xes) and finite stochastic language (.slang).
`<FILE_2>`	A labelled Petri net with the control flow. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* Petri net markup language (.pnml), labelled Petri net (.lpn), directly follows model (.dfm) and stochastic labelled Petri net (.slpn).
`-o` or `--output <FILE>`	The stochastic labelled Petri net (.slpn) file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
`-a` or `--approximate`	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: stochastic labelled Petri net.

1.19 `Ebi discover uniform`

Alias: `Ebi disc uni`.
Give each transition a weight of 1.

Parameter	
`<LPN_FILE>`	A labelled Petri net. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* stochastic labelled Petri net (.slpn), labelled Petri net (.lpn), directly follows model (.dfm) and Petri net markup language (.pnml).
`-o` or `--output <FILE>`	The stochastic labelled Petri net (.slpn) file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
`-a` or `--approximate`	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: stochastic labelled Petri net.

1.20 `Ebi information`

Alias: `Ebi info`.
Show information about an object.

Parameter	
<FILE>	Any file supported by Ebi. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* event log (.xes), Petri net markup language (.pnml), directly follows model (.dfm), labelled Petri net (.lpn), stochastic deterministic finite automaton (.sdfa), compressed event log (.xes.gz), stochastic labelled Petri net (.slpn), finite language (.lang) and finite stochastic language (.slang).
`-o` or `--output` <FILE>	The text file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
`-a` or `--approximate`	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: text.

1.21 `Ebi latex-help graph`

Print the graph of Ebi.

Parameter	
`-o` or `--output` <FILE>	The text file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no

Output: text.

1.22 `Ebi latex-help manual`

Alias: `Ebi latex man`.
Print the automatically generated parts of the manual of Ebi in Latex format.

Parameter	
`-o` or `--output` <FILE>	The text file to which the result must be written. If the parameter is not given, the results will be written to STDOUT.

	Mandatory: no

Output: text.

1.23 `Ebi probability model`

Alias: `Ebi prob mod`.
Compute the probability that a queriable stochastic language (stochastic model) produces any trace of the model.
More information: [4].

Parameter	
`<FILE_1>`	The queriable stochastic language (model). *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* compressed event log (.xes.gz), stochastic labelled Petri net (.slpn), event log (.xes), finite stochastic language (.slang) and stochastic deterministic finite automaton (.sdfa).
`<FILE_2>`	The finite language (log). *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* finite language (.lang), event log (.xes), finite stochastic language (.slang) and compressed event log (.xes.gz).
`-o` or `--output <FILE>`	The fraction file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
`-a` or `--approximate`	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: fraction.

1.24 `Ebi probability trace`

Alias: `Ebi prob trac`.
Compute the probability of a trace in a queriable stochastic language (model).
More information: [4].

Parameter	
`<FILE>`	The queriable stochastic language (model). *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line.

	Accepted values: compressed event log (.xes.gz), stochastic deterministic finite automaton (.sdfa), finite stochastic language (.slang), event log (.xes) and stochastic labelled Petri net (.slpn).
<TRACE>	The trace. *Mandatory:* yes
-o or --output <FILE>	The fraction file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
-a or --approximate	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: fraction.

1.25 Ebi sample

Alias: `Ebi sam`.
Sample traces randomly. Please note that this may run forever if the model contains a livelock.

Parameter	
<FILE>	The stochastic semantics (model). *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* event log (.xes), finite stochastic language (.slang), compressed event log (.xes.gz), stochastic deterministic finite automaton (.sdfa) and stochastic labelled Petri net (.slpn).
<NUMBER_OF_TRACES>	The number of traces to be sampled. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* integer.
-o or --output <FILE>	The finite stochastic language (.slang) file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
-a or --approximate	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: finite stochastic language.

1.26 Ebi test log-categorical-attribute

Alias: `Ebi tst lcat`.
Test the hypothesis that the sub-logs defined by the categorical attribute are derived from identical processes.; 500 samples are taken.
More information: [5].

Parameter	
<FILE>	The event log for which the test is to be performed. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* compressed event log (.xes.gz) and event log (.xes).
<ATTRIBUTE>	The trace attribute for which the test is to be performed. The trace attributes of a log can be found using 'Ebi info'. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* text.
-s or –number-of-samples	Take a number of samples. *Mandatory:* no
-p or –p-value	Use threshold p-value. *Mandatory:* no
-o or --output <FILE>	The text file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
-a or --approximate	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: text.

1.27 Ebi validate

Alias: Ebi vali.
Attempt to parse any file supported by Ebi. If you do not know the type the file should have, try 'Ebi info'.

Parameter	
<TYPE>	The type for which parsing should be attempted. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* the file extension of any file type supported by Ebi (lang, lpn, slpn, slang, sdfa, xes, xes.gz, dfm or pnml).
<FILE>	The file to be parsed. *Mandatory:* yes
-o or --output <FILE>	The text file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
-a or --approximate	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: text.

1.28 `Ebi visualise svg`

Visualise an object as scalable vector graphics.

Parameter	
`<FILE>`	Any file that can be visualised as a graph. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* stochastic labelled Petri net (.slpn), labelled Petri net (.lpn), Petri net markup language (.pnml), stochastic deterministic finite automaton (.sdfa) and directly follows model (.dfm).
`-o` or `--output <FILE>`	The text file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no
`-a` or `--approximate`	Use approximate arithmetic instead of exact arithmetic. *Mandatory:* no

Output: text.

1.29 `Ebi visualise text`

Alias: `Ebi vis txt`.
Visualise an object as text.

Parameter	
`<FILE>`	Any file that can be visualised textually. *Mandatory:* yes, though it can be given on STDIN by giving an '-' on the command line. *Accepted values:* labelled Petri net (.lpn), stochastic labelled Petri net (.slpn), finite language (.lang), finite stochastic language (.slang), compressed event log (.xes.gz), directly follows model (.dfm), stochastic deterministic finite automaton (.sdfa), Petri net markup language (.pnml) and event log (.xes).
`-o` or `--output <FILE>`	The text file to which the result must be written. If the parameter is not given, the results will be written to STDOUT. *Mandatory:* no

Output: text.

2 Supported files

Please note that Ebi does not consider input file extensions. For plug-ins with multiple output file formats, the file extension is used to determine the output format.

2.1 finite language (.lang)

Import as objects: finite language.
Import as traits: finite language.
Input to commands: `Ebi analyse-non-stochastic cluster`, `Ebi analyse-non-stochastic medoid`, `Ebi information`, `Ebi probability model`, `Ebi validate`, `Ebi visualise text`.

2.2 labelled Petri net (.lpn)

Import as objects: labelled Petri net.
Import as traits: labelled Petri net, semantics.
Input to commands: `Ebi convert labelled-Petri-net`, `Ebi discover occurrence`, `Ebi discover uniform`, `Ebi information`, `Ebi validate`, `Ebi visualise svg`, `Ebi visualise text`.

2.3 stochastic labelled Petri net (.slpn)

Import as objects: stochastic labelled Petri net, labelled Petri net.
Import as traits: queriable stochastic language, stochastic semantics, stochastic deterministic semantics, labelled Petri net, semantics.
Input to commands: `Ebi analyse all-traces`, `Ebi analyse minimum-probability-traces`, `Ebi analyse mode`, `Ebi analyse most-likely-traces`, `Ebi conformance entropic-relevance`, `Ebi conformance jensen-shannon`, `Ebi conformance jensen-shannon-sample`, `Ebi conformance unit-earth-movers-stochastic-conformance`, `Ebi convert labelled-Petri-net`, `Ebi discover occurrence`, `Ebi discover uniform`, `Ebi information`, `Ebi probability model`, `Ebi probability trace`, `Ebi sample`, `Ebi validate`, `Ebi visualise svg`, `Ebi visualise text`.

2.4 finite stochastic language (.slang)

Import as objects: finite stochastic language.
Import as traits: finite language, finite stochastic language, queriable stochastic language, iterable stochastic language, stochastic semantics, stochastic deterministic semantics.
Input to commands: `Ebi analyse all-traces`, `Ebi analyse medoid`, `Ebi analyse minimum-probability-traces`, `Ebi analyse mode`, `Ebi analyse most-likely-traces`, `Ebi analyse-non-stochastic cluster`, `Ebi analyse-non-stochastic medoid`, `Ebi conformance entropic-relevance`, `Ebi conformance jensen-shannon`, `Ebi conformance jensen-shannon-sample`, `Ebi conformance unit-earth-movers-stochastic-conformance`, `Ebi convert finite-stochastic-language`, `Ebi convert stochastic-finite-deterministic-automaton`, `Ebi discover occurrence`, `Ebi information`, `Ebi probability model`, `Ebi probability trace`, `Ebi sample`, `Ebi validate`, `Ebi visualise text`.

2.5 stochastic deterministic finite automaton (.sdfa)

Import as objects: stochastic deterministic finite automaton.
Import as traits: queriable stochastic language, stochastic deterministic semantics, stochastic semantics, semantics.
Input to commands: `Ebi analyse all-traces`, `Ebi analyse minimum-probability-traces`, `Ebi analyse mode`, `Ebi analyse most-likely-traces`, `Ebi conformance entropic-relevance`, `Ebi conformance jensen-shannon`, `Ebi conformance jensen-shannon-sample`, `Ebi conformance unit-earth-movers-stochastic-conformance`, `Ebi convert labelled-Petri-net`, `Ebi convert stochastic-finite-deterministic-automaton`, `Ebi information`, `Ebi probability model`, `Ebi probability trace`, `Ebi sample`, `Ebi validate`, `Ebi visualise svg`, `Ebi visualise text`.

2.6 event log (.xes)

Import as objects: event log.
Import as traits: finite language, finite stochastic language, queriable stochastic language, iterable stochastic language, event log, stochastic semantics, stochastic deterministic semantics.
Input to commands: `Ebi analyse all-traces`, `Ebi analyse completeness`, `Ebi analyse medoid`, `Ebi analyse minimum-probability-traces`, `Ebi analyse mode`, `Ebi analyse most-likely-traces`, `Ebi analyse-non-stochastic cluster`, `Ebi analyse-non-stochastic medoid`, `Ebi association all-trace-attributes`, `Ebi association trace-attribute`, `Ebi conformance entropic-relevance`, `Ebi conformance jensen-shannon`, `Ebi conformance jensen-shannon-sample`, `Ebi conformance unit-earth-movers-stochastic-conformance`, `Ebi convert finite-stochastic-language`, `Ebi convert stochastic-finite-deterministic-automaton`, `Ebi discover occurrence`, `Ebi information`, `Ebi probability model`, `Ebi probability trace`, `Ebi sample`, `Ebi test log-categorical-attribute`, `Ebi validate`, `Ebi visualise text`.

2.7 compressed event log (.xes.gz)

Import as objects: event log.
Import as traits: finite language, finite stochastic language, queriable stochastic language, iterable stochastic language, event log, stochastic deterministic semantics.
Input to commands: `Ebi analyse all-traces`, `Ebi analyse completeness`, `Ebi analyse medoid`, `Ebi analyse minimum-probability-traces`, `Ebi analyse mode`, `Ebi analyse most-likely-traces`, `Ebi analyse-non-stochastic cluster`, `Ebi analyse-non-stochastic medoid`, `Ebi association all-trace-attributes`, `Ebi association trace-attribute`, `Ebi conformance entropic-relevance`, `Ebi conformance jensen-shannon`, `Ebi conformance jensen-shannon-sample`, `Ebi conformance unit-earth-movers-stochastic-conformance`, `Ebi convert finite-stochastic-language`, `Ebi convert stochastic-finite-deterministic-automaton`, `Ebi discover occurrence`, `Ebi information`, `Ebi probability model`, `Ebi probability trace`, `Ebi sample`, `Ebi test log-categorical-attribute`, `Ebi validate`, `Ebi visualise text`.

2.8 directly follows model (.dfm)

Import as objects: directly follows model, labelled Petri net.
Import as traits: labelled Petri net.

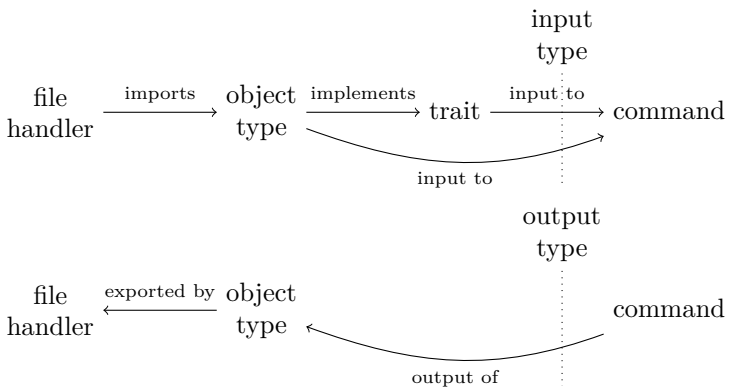


Figure 1: Traits and object types in Ebi.

Input to commands: `Ebi convert labelled-Petri-net`, `Ebi discover occurrence`, `Ebi discover uniform`, `Ebi information`, `Ebi validate`, `Ebi visualise svg`, `Ebi visualise text`.

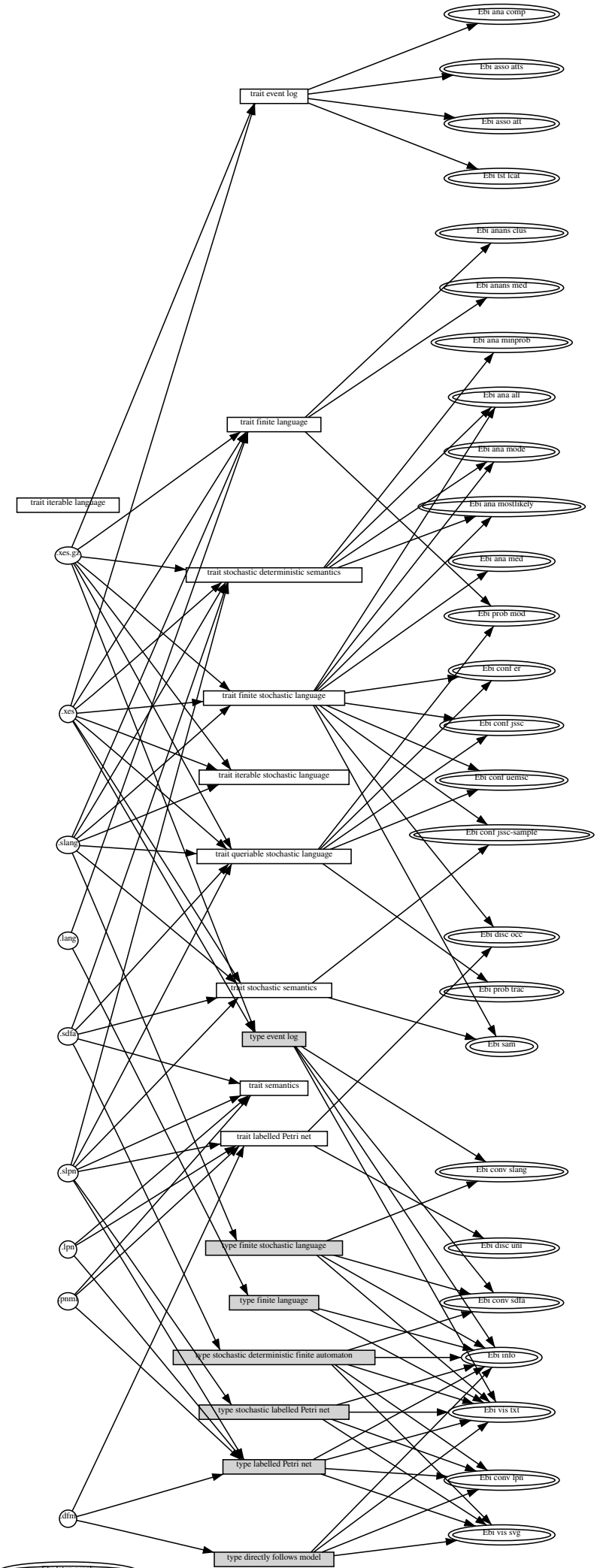
2.9 Petri net markup language (.pnml)

Import as objects: labelled Petri net.
Import as traits: labelled Petri net, semantics.
Input to commands: `Ebi convert labelled-Petri-net`, `Ebi discover occurrence`, `Ebi discover uniform`, `Ebi information`, `Ebi validate`, `Ebi visualise svg`, `Ebi visualise text`.

3 Architecture

3.1 Files and objects

Figure 1 shows the main architectural concepts of Ebi with regards to files and objects. It's a bit verbose, but it enables commands to be completely detached from any I/O, and reduces overhead when adding new functionality, and guarantees the documentation is up-to-date.


Ebi ana comp
Ebi asso atts
Ebi asso att
Ebi tst lcat
Ebi anans clus
Ebi anans med
Ebi ana minprob
Ebi ana all
Ebi ana mode
Ebi ana mostlikely
Ebi ana med
Ebi prob mod
Ebi conf er
Ebi conf jssc
Ebi conf uemsc
Ebi conf jssc-sample
Ebi disc occ
Ebi prob trac
Ebi sam
Ebi conv slang
Ebi disc uni
Ebi conv sdfa
Ebi info
Ebi vis txt
Ebi conv lpn
Ebi vis svg
trait event log
trait finite language
trait iterable language
trait stochastic deterministic semantics
trait finite stochastic language
trait iterable stochastic language
trait queriable stochastic language
trait stochastic semantics
trait semantics
trait labelled Petri net
type event log
type finite stochastic language
type finite language
type stochastic deterministic finite automaton
type stochastic labelled Petri net
type labelled Petri net
type directly follows model
xes.gz
xes
slang
lang
sdfa
slpn
lpn
pnml
dfm

Command A command is a function of Ebi that users can call. A command declares what types of inputs it needs and what type of output it generates. Preferably, commands take traits as inputs rather than object types.

Commands can circumvent the Ebi object handling on their inputs by declaring an `cli_command` function, which gives direct access to the command line interface building process. This is for instance used in the `validate` command, where we want to obtain the parsing error for a particular object type.

The actual input for a command gets wrapped in the `EbiInput` enum, and the output must be wrapped in the `EbiOutput` enum. The Ebi machinery verifies whether the output of a command matches the declared output type at run time.

Commands define a number of inputs, and for each input a list of acceptable types. However, these types need to be similar: a command cannot ask for either a file and a fraction, for instance.

Ebi has the following commands:

- `Ebi analyse all-traces` or `Ebi ana all`
- `Ebi analyse completeness` or `Ebi ana comp`
- `Ebi analyse medoid` or `Ebi ana med`
- `Ebi analyse minimum-probability-traces` or `Ebi ana minprob`
- `Ebi analyse mode` or `Ebi ana mode`
- `Ebi analyse most-likely-traces` or `Ebi ana mostlikely`
- `Ebi analyse-non-stochastic cluster` or `Ebi anans clus`
- `Ebi analyse-non-stochastic medoid` or `Ebi anans med`
- `Ebi association all-trace-attributes` or `Ebi asso atts`
- `Ebi association trace-attribute` or `Ebi asso att`
- `Ebi conformance entropic-relevance` or `Ebi conf er`
- `Ebi conformance jensen-shannon` or `Ebi conf jssc`
- `Ebi conformance jensen-shannon-sample` or `Ebi conf jssc-sample`
- `Ebi conformance unit-earth-movers-stochastic-conformance` or `Ebi conf uemsc`
- `Ebi convert finite-stochastic-language` or `Ebi conv slang`
- `Ebi convert labelled-Petri-net` or `Ebi conv lpn`
- `Ebi convert stochastic-finite-deterministic-automaton` or `Ebi conv sdfa`
- `Ebi discover occurrence` or `Ebi disc occ`
- `Ebi discover uniform` or `Ebi disc uni`
- `Ebi information` or `Ebi info`

- `Ebi latex-help graph` or `Ebi latex graph`
- `Ebi latex-help manual` or `Ebi latex man`
- `Ebi probability model` or `Ebi prob mod`
- `Ebi probability trace` or `Ebi prob trac`
- `Ebi sample` or `Ebi sam`
- `Ebi test log-categorical-attribute` or `Ebi tst lcat`
- `Ebi validate` or `Ebi vali`
- `Ebi visualise svg` or `Ebi vis svg`
- `Ebi visualise text` or `Ebi vis txt`

Input type An input type is a declaration of what a command needs as input. An input type can be:

- a trait;
- an object type;
- a basic data type such as fraction, integer or string;
- the "any object" input type, which denotes that any object type will be accepted;
- a "file handler", which indicates the (name of) a file handler (for instance used in the `validate` command).

The actual input for a command gets wrapped in the `EbiInput` enum.

Trait A "trait" is like an interface: it provides some functions but does not say how these are implemented. For instance "finite stochastic language" is a trait, which allows to iterate over traces and their probabilities, and to obtain the number of traces. Preferably, commands should require traits as inputs.

Ebi has the following traits:

- event log
- iterable language
- finite language
- finite stochastic language
- iterable stochastic language
- queriable stochastic language
- semantics

- stochastic deterministic semantics
- stochastic semantics
- labelled Petri net

Object type An object type is a struct or class. It is a specific implementation with a fixed data structure. The output of a command is an object of a specific object type. The input of a command rather not uses object types, but sometimes this is unavoidable. Objects are wrapped in an `EbiObject` enum to ensure type safety by the compiler.

Ebi has the following object types:

- directly follows model
- event log
- finite language
- finite stochastic language
- labelled Petri net
- stochastic deterministic finite automaton
- stochastic labelled Petri net

File handler A file handler is an importer for a particular file format.

Ebi has the following file handlers:

- finite language (.lang)
- labelled Petri net (.lpn)
- stochastic labelled Petri net (.slpn)
- finite stochastic language (.slang)
- stochastic deterministic finite automaton (.sdfa)
- event log (.xes)
- compressed event log (.xes.gz)
- directly follows model (.dfm)
- Petri net markup language (.pnml)

3.2 Exact Computation

In Ebi, several computations are performed without rounding. To this end, it uses positive fractions:

$$\mathbb{Q} = \{\frac{x}{y} \mid x, y \in \mathbb{N}\}$$

Fractions are well-supported by standard libraries. Even for seemingly small fractions, the natural numbers that compose the fraction can get *huge*. Therefore, simple 32 or 64 bit integers do not suffice for x and y, and a large, unbounded, integer library is necessary.

By default, Ebi uses exact arithmetic, unless it is disabled by a function explicitly. This can be done with `Fraction::set_exact_globally(exact: bool)`. Please note that exact and approximate (double precision) arithmetic cannot be combined, thus functions should set this function as early as possible.

3.2.1 Square roots

Square roots can only be expressed in $\mathbb{N}$ or in $\mathbb{I}$. Thus, we need to represent square roots symbolically. For this, we use the following operations:

$$\begin{aligned}\frac{a}{b} &= \sqrt{\frac{a^2}{b^2}} \\ \sqrt{a}\sqrt{b} &= \sqrt{ab} \\ \frac{\sqrt{a}}{\sqrt{b}} &= \sqrt{\frac{a}{b}}\end{aligned}$$

Thus, we can represent all of these square root operations with the square root of a fractional number.

3.2.2 Logdiv

Logarithms can often not be expressed in $\mathbb{Q}$. For instance, $\log_2(3) \notin \mathbb{Q}$ is not a rational number. To perform computations exactly and to delay rounding to the very last moment, Ebi uses *log division* objects. That is, computations that involve logarithms are performed symbolically in order not to have to evaluate the logarithm.

A log division object (logdiv) is a base-2 logarithm of a positive fraction, divided by a natural number:

$$\begin{aligned}\text{logdiv} &\colon \mathbb{Q} \times \mathbb{N} \to \mathbb{I} \\ \text{logdiv}(\frac{a}{b}, c) &= \frac{\log(\frac{a}{b})}{c} \\ &= \log \sqrt[c]{\frac{a}{b}}\end{aligned}$$

From a fraction Any non-negative fraction can be written as a logdiv:

$$\begin{aligned}\frac{a}{b} &= \frac{\log(2^a)}{b} \\ &= \text{logdiv}(\frac{2^a}{1}, b)\end{aligned}$$

Sum & Difference Logdivs are closed under addition:

$$\begin{aligned}\operatorname{logdiv}(\frac{a}{b}, c) + \operatorname{logdiv}(\frac{d}{e}, f) &= \frac{\log(\frac{a}{b})}{c} + \frac{\log(\frac{d}{e})}{f} \\ &= \frac{f\log(\frac{a}{b}) + c\log(\frac{d}{e})}{cf} \\ &= \frac{\log(\frac{a^f}{b^f}) + \log(\frac{d^c}{e^c})}{cf} \\ &= \frac{\log(\frac{a^f d^c}{b^f e^c})}{cf} \\ &= \operatorname{logdiv}(\frac{a^f d^c}{b^f e^c}, cf)\end{aligned}$$

As a, b, c, d, e and f are all positive integers, the sum of two logdivs is also a logdiv. Similarly, for subtraction:

$$\begin{aligned}\operatorname{logdiv}(\frac{a}{b}, c) - \operatorname{logdiv}(\frac{d}{e}, f) &= \frac{\log(\frac{a}{b})}{c} - \frac{\log(\frac{d}{e})}{f} \\ &= \frac{\log(\frac{a^f}{b^f}) - \log(\frac{d^c}{e^c})}{cf} \\ &= \frac{\log(\frac{a^f}{b^f} / \frac{d^c}{e^c})}{cf} \\ &= \operatorname{logdiv}(\frac{a^f e^c}{b^f d^c}, cf)\end{aligned}$$

As a, b, d and e are all positive integers, the difference between two logdivs is also a logdiv.

Additive identity (0) The additive identity of logdiv is $\operatorname{logdiv}(\frac{1}{1}, 1) = 0$.

While there are infinitely many logdivs that equal 0, such as $\operatorname{logdiv}(\frac{5}{5}, 100) = 0$, only a second argument of 1 guarantees that addition and subtraction work as intended.

$n \log n$ Given a fraction $\frac{a}{b} \in \mathbb{Q}$, the value $\frac{a}{b} \log \frac{a}{b}$ can be represented by

$$\begin{aligned}\frac{a}{b} \log \frac{a}{b} &= \frac{a\log(\frac{a}{b})}{b} \\ &= \frac{\log(\frac{a^a}{b^a})}{b} \\ &= \operatorname{logdiv}(\frac{a^a}{b^a}, b)\end{aligned}$$

Approximation In order to provide readable results that can be used in other tools, at the end of the computation chain, logdivs need to be approximated to a fraction. To this end, we consider that:

$$\text{logdiv}(\frac{a}{b}, c) = \frac{\log(\frac{a}{b})}{c}$$

Thus, we need to compute $\log(q)$ with $q \in \mathbb{Q}$. As the libraries we use do not support computing logarithms on fractions with arbitrary large representations, we must implement this ourselves.

Taylor expansion Using Taylor expansion, we translate the approximation of the logarithm base 2 to q to smaller and smaller values, until we end up with a $q' \leq 2$. That is, we repeatedly divide q by two and add one to the result, until q drops below 2. Then, we can approximate $\log_2(q)$ with a Taylor series on the natural logarithm[1].

$$\log(q) = \begin{cases} 1 + \log(\frac{q}{2}) & \text{if } q > 2 \\ \frac{\ln(q)}{\ln(2)} & \text{if } 0 < q \leq 2 \end{cases}$$

$$\ln(q) = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{(z-1)^k}{k} \text{ for } 0 < q \leq 2$$

We then approximate $\ln(q)$ up until the terms become small enough for the required precision.

Bits As even standard multiplications are expensive for fractions, we need a more efficient strategy to bring the approximation to acceptable speed:

- If $\frac{a}{b} < \frac{1}{2}$:

$$\begin{aligned} \log(\frac{a}{b}) &= \log(a) - \log(b) \\ &= -(\log(b) - \log(a)) \\ &= -\log(\frac{b}{a}) \end{aligned}$$

  In which $\frac{b}{a}$ is larger than 2.

  Complexity: pointer swap.

- While $\frac{a}{b} > 1$:

$$\log(\frac{a}{b}) = \begin{cases} 1 + \log(\frac{a}{2b}) & \text{if } a \text{ is odd} \\ 1 + \log(\frac{a/2}{b}) & \text{if } a \text{ is even} \end{cases}$$

  Complexity: for each iteration a left or right bit shift plus a comparison (right bit shift).

[1] `https://en.wikipedia.org/wiki/Logarithm`

- Then, $\frac{1}{2} < \frac{a}{b} < 1$. As a and b may be *huge*, we cannot easily translate our fraction to a standard floating-point number. Therefore, we use the following procedure to obtain as many digits as necessary for the required precision (where 3.3 bits yield one decimal):

$$\log(\frac{a}{b})_n = \begin{cases} 1 & \text{if } a = b \\ \log(\frac{a^2}{2b^2})_{n+1} - \frac{1}{2^n} & \text{if } \frac{a^2}{b^2} > 2 \qquad \text{report a binary 1} \\ \log(\frac{a^2}{b^2})_{n+1} & \text{otherwise} \qquad \text{report a binary 0} \end{cases}$$

  In which n denotes the iteration.

  Complexity: two multiplications and at most one bit shift per bit precision. As multiplications are prohibitively expensive, and squaring roughly doubles the number of bits each iteration, we truncate a and b by the same number of bit-shifts every iteration, such that in each number enough bits are left.

3.2.3 RootLogDiv

A RootLogDiv is the following:

$$\begin{aligned} \text{RootLogDiv}(\frac{a}{b}, c) &= \sqrt{\text{logdiv}(\frac{a}{b}, c)} \\ &= \sqrt{\frac{\log \frac{a}{b}}{c}} \end{aligned}$$

There is an option to perform the 1- operation on a RootLogDiv.

4 Entropic Relevance

Entropic relevance is computed as follows:

Definition 1 (Entropic Relevance [1])**.** *Let L be a finite stochastic language and let M be a queriable stochastic langauge. Let Λ be the set of all activities appearing in the traces of L. Then, the* entropic relevance (ER) of M to L *is defined as follows:*

$$ER(L, M) = H_0\left(\sum_{\sigma \in \bar{L},\, M(\sigma) > 0} L(\sigma)\right) + \sum_{\sigma \in \bar{L}} L(\sigma) J(\sigma, M)$$

$$J(\sigma, M) = \begin{cases} -\log_2 M(\sigma) & M(\sigma) > 0 \\ (1 + |\sigma|) \log_2(1 + |\Lambda|)) & \textit{otherwise} \end{cases}$$

$$H_0(x) = -x \log_2 x - (1 - x) \log_2 (1 - x) \ \textit{with}\ H_0(0) = H_0(1) = 0$$

References

[1] Hanan Alkhammash, Artem Polyvyanyy, Alistair Moffat, and Luciano García-Bañuelos. Entropic relevance: A mechanism for measuring stochastic process models discovered from event data. *Inf. Syst.*, 107:101922, 2022.

[2] Adam Burke, Sander J. J. Leemans, and Moe Thandar Wynn. Stochastic process discovery by weight estimation. In Sander J. J. Leemans and Henrik Leopold, editors, *Process Mining Workshops - ICPM 2020 International Workshops, Padua, Italy, October 5-8, 2020, Revised Selected Papers*, volume 406 of *Lecture Notes in Business Information Processing*, pages 260–272. Springer, 2020.

[3] Martin Kabierski, Markus Richter, and Matthias Weidlich. Addressing the log representativeness problem using species discovery. In *ICPM*, pages 65–72. IEEE, 2023.

[4] Sander J. J. Leemans, Fabrizio Maria Maggi, and Marco Montali. Enjoy the silence: Analysis of stochastic petri nets with silent transitions. *Information Systems*, in press, 2024.

[5] Sander J. J. Leemans, James M. McGree, Artem Polyvyanyy, and Arthur H. M. ter Hofstede. Statistical tests and association measures for business processes. *IEEE Trans. Knowl. Data Eng.*, 35(7):7497–7511, 2023.

[6] Sander J. J. Leemans, Anja F. Syring, and Wil M. P. van der Aalst. Earth movers' stochastic conformance checking. In Thomas T. Hildebrandt, Boudewijn F. van Dongen, Maximilian Röglinger, and Jan Mendling, editors, *Business Process Management Forum - BPM Forum 2019, Vienna, Austria, September 1-6, 2019, Proceedings*, volume 360 of *Lecture Notes in Business Information Processing*, pages 127–143. Springer, 2019.

[7] Erich Schubert and Peter J. Rousseeuw. Fast and eager k-medoids clustering: O(k) runtime improvement of the pam, clara, and CLARANS algorithms. *Inf. Syst.*, 101:101804, 2021.