Entropia 1.7

User's Guide

Hanan Alkhammash*[1] and Anandi Karunaratne†[1]

[1] *The University of Melbourne*
{halkhammash, anandik}@student.unimelb.edu.au

Entropia is an open-source command-line tool that implements a family of conformance checking measures. This guide is intended to explains through a series of simple steps how to use the *Entropia* tool to measure precision and recall between a model and an event log using classical non-deterministic and stochastic conformance checking approaches. This guide covers the following topics:

1. Getting Started: find out how to get *Entropia* and prerequisites required to run the tool.
2. Classical Non-deterministic Conformance Checking Measures: The *Entropia* commands for the are provided where each command-line instruction is illustrated with an example, making it easy to find the information you are looking for. The section covers the following measures:
    1. Exact Matching Precision and Recall [8],
    2. Partial Matching Precision and Recall [5]; and
    3. Controlled Partial Matching Precision and Recall [2],
3. The *Entropia* commands for stochastic conformance checking measures. Here, you can find out a detailed description you need to apply different stochastic conformance checking approaches. The section includes the following measures:
    1. Stochastic Precision and Recall [4]; and
    2. Entropic Relevance [6].
4. Bootstrap Generalization: A section detailing the *Entropia* commands and processes for applying the Bootstrap Generalization approach to assess the generalization of a discovered process model described as a directly-follows-graph (DFG).
5. Log Representativeness Measures: Learn about the measures used to evaluate log representativeness and how to apply them using *Entropia*, ensuring that your logs adequately reflect the underlying process.

*Documentation completed for version 1.5 on August 22, 2020.
†Documentation updated for version 1.7 on August 22, 2024.

Getting Started with Entropia

This section will quickly get you started in using the *Entropia* tool.

a) Checking for prerequisite

Prior to running the *Entropia* commands, you will need to ensure that the **JDK** (Java Development Kit) is in place.

b) Downloading and Running Entropia

1. Clone or download the JPBT library to your local machine: https://github.com/jbpt/codebase.
2. Navigate to the **jbpt-pm/entropia/** folder in your terminal.
3. Issue the following command to verify that the *Entropia* tool is properly downloaded and display its version number as shown in the output screen:

```
>java -jar jbpt-pm-entropia-1.5.jar -v
```

Output Screen:

```
>java -jar jbpt-pm-entropia-1.5.jar -v
1.5
```

Use the option (`-h`) to display the help message that shows information of the core and specific options of the *Entropia* tool.

```
>java -jar jbpt-pm-entropia-1.5.jar -h
```

Output Screen:

```
>java -jar jbpt-pm-entropia-1.5.jar -h
================================================================================
Tool to compute quality measures for Process Mining and Process Querying ver. 1.5.
For support, please contact us at jbpt.project@gmail.com.
================================================================================
PNML format:    http://www.pnml.org/
XES format:     https://xes-standard.org/
================================================================================

usage: java -jar jbpt-pm.jar <options>
 -dent,--diluted-entropy                      compute entropy measure (for "diluted" traces)
 -doent,--diluted-optimized-entropy           compute entropy measure (for "diluted" traces
                                              with optimization)
 -ent,--entropy                               compute entropy measure (for exact traces)
 -h,--help                                    print help message
 -popr,--partial-optimized-precision-recall   compute entropy-based precision and recall (partial
                                              trace matching with optimization)
 -ppr,--partial-precision-recall              compute entropy-based precision and recall (partial
                                              trace matching)
 -pr,--precision-recall                       compute entropy-based precision and recall (exact
                                              trace matching)
 -rel,--relevant <file path>                  model that describes relevant traces (XES or PNML)
 -ret,--retrieved <file path>                 model that describes retrieved traces (XES or PNML)
 -s,--silent                                  print the results only
 -sk,--skips                                  add specified amount of skips to traces
 -skrel,--skrelevant <number of skips>        add specified amount of skips to relevant traces
 -skret,--skretrieved <number of skips>       add specified amount of skips to retrieved traces
 -v,--version                                 get version of this tool
================================================================================
```

Models and Event Log

For the first two sections, which cover conformance checking measures for precision and recall, the tutorial will use example files that are provided with the tool, which are the event log coded in XES format (log.xes), process model modelled as a Petri Net (model.pnml) and stochastic process model as an SDFA coded in JSON format (automaton.json). Table 1, Figure 1 and Figure 2 represent the three files, respectively. For the last two sections where the bootstrap generalization measure and log representativeness measures are explained, we will use Figure 3 (system3.json) as the system, Figure 4 (model3.json) as the model, Table 2 (log3.xes) as the event log generated from the system in Figure 3.

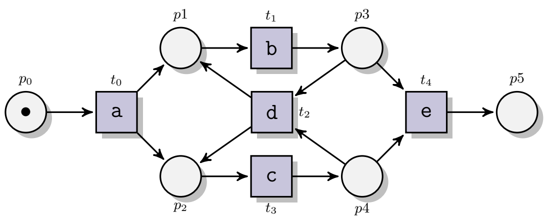

Figure 1: Process Model.

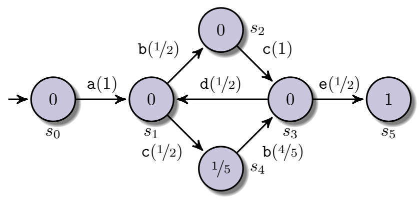

Figure 2: Stochastic Process Model.

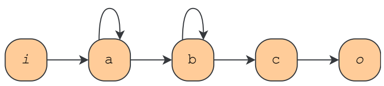

Figure 3: A ystem represented as a DFG.

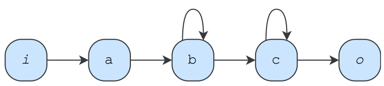

Figure 4: A process model represented as a DFG.

Trace
⟨a, b, c, e⟩
⟨a, b, c, d, c, b, e⟩
⟨a, b, d, c, b, e⟩
⟨a, c, e⟩
⟨b, c, e⟩
⟨b, c, e⟩
⟨a, a, a, c, b, e⟩

Table 1: An Event Log.

L
$< a, b, c >^{15}$
$< a, b, b, c >^{10}$
$< a, b, b, b, c >^{4}$
$< a, b, c, c >^{3}$
$< a, a, b, c >^{1}$

Table 2: An Event Log generated by system in Figure 3.

All files are located in the folder **jbpt-pm/entropia/examples/**. If the folder does not contain the files, you can download them from `https://github.com/jbpt/codebase/tree/master/jbpt-pm/entropia/examples`. It is recommended to use the event log and models provided to understand how to run the *Entropia* commands. Then, try it with your own data. Refer to Table III in the demo paper for more details about file types and formats supported by the tool.

Classical Non-Deterministic Conformance Checking Measures

a) Matching Precision and Recall Measures

To compute the exact matching precision value between the event log (log.xes) and process model (model.pnml), use the option (`-emp`) as follows.

```
>java -jar jbpt-pm-entropia-1.5.jar -emp -rel=log.xes -ret=model.pnml
```

Note that in the command the paths to the event log and process model files are assigned to the relevant (-rel=<path>) and retrieved (-ret=<path>) models respectively.

Output Screen:

```
>java -jar jbpt-pm-entropia-1.5.jar -emp -rel=log.xes -ret=model.pnml
================================================================================
Tool to compute quality measures for Process Mining and Process Querying ver. 1.5.
For support, please contact us at jbpt.project@gmail.com.
================================================================================
PNML format:    http://www.pnml.org/
XES format:     https://xes-standard.org/
================================================================================

Computing eigenvalue-based precision and recall based on exact matching of traces.
The technique is described in:
Artem Polyvyanyy, Andreas Solti, Matthias Weidlich, Claudio Di Ciccio,
Jan Mendling. Monotone Precision and Recall for Comparing Executions and
Specifications of Dynamic Systems.
ACM Transactions on Software Engineering and Methodology (TOSEM) (2020)

Loading the retrieved model from C:\Users\halkhammash\git\codebase-master\jbpt-pm\model.pnml.
The retrieved modeln loaded in                          80 ms.
Loading the relevant model from C:\Users\halkhammash\git\codebase-master\jbpt-pm\log.xes.
The relevant model loaded in                            40 ms.
The boundedness of the retrieved model checked in       222 ms.
The boundedness of the relevant model checked in        0 ms.
Constructing automaton RET that encodes the retrieved model.
Automaton RET constructed in                            6 ms.
Automaton RET has 6 states and 7 transitions.
Constructing automaton REL that encodes the relevant model.
Automaton REL constructed in                            1 ms.
Automaton REL has 10 states and 14 transitions.

===================Calculating precision and recall=============================

The intersection INT of RET and REL constructed in      3 ms.
Automaton INT has 8 states and 8 transitions.

A largest eigenvalue of the adjacency matrix of REL is 1.3899267936242778.
A largest eigenvalue of the adjacency matrix of REL computed in     85 ms.

A largest eigenvalue of the adjacency matrix of RET is 1.4371560431001367.
A largest eigenvalue of the adjacency matrix of RET computed in     2 ms.

A largest eigenvalue of the adjacency matrix of INT is 1.1147977972610885.
A largest eigenvalue of the adjacency matrix of INT computed in     2 ms.

Precision computed in                                               4 ms.
Precision: 0.7756971155730045.
```

When the option (-s) is added to commands, the tool runs in the silent mode. The following command is an example of using the (-s) option:

```
>java -jar jbpt-pm-entropia-1.5.jar -emp -s -rel=log.xes -ret=model.pnml
```

The tool, in the silent mode, only prints the result, in this case the exact matching precision, omitting the debug information and execution data. The expected output of the command will be as the following.

Output Screen:

```
>java -jar jbpt-pm-entropia-1.5.jar -emp -s -rel=log.xes -ret=model.pnml
0.7756971155730045.
```

By replacing the option (-emr) with (-emp), the tool computes the exact matching recall value between the event log and process model.

```
>java -jar jbpt-pm-entropia-1.5.jar -emr -rel=log.xes -ret=model.pnml
```

Output Screen:

```
>java -jar jbpt-pm-entropia-1.5.jar -emr -rel=log.xes -ret=model.pnml
================================================================================
Tool to compute quality measures for Process Mining and Process Querying ver. 1.5.
For support, please contact us at jbpt.project@gmail.com.
================================================================================
PNML format:    http://www.pnml.org/
XES format:     https://xes-standard.org/
================================================================================

Computing eigenvalue-based precision and recall based on exact matching of traces.
The technique is described in:
Artem Polyvyanyy, Andreas Solti, Matthias Weidlich, Claudio Di Ciccio,
Jan Mendling. Monotone Precision and Recall for Comparing Executions and
Specifications of Dynamic Systems.
ACM Transactions on Software Engineering and Methodology (TOSEM) (2020)

Loading the retrieved model from C:\Users\halkhammash\git\codebase-master\jbpt-pm\model.pnml.
The retrieved modeln loaded in                              80 ms.
Loading the relevant model from C:\Users\halkhammash\git\codebase-master\jbpt-pm\log.xes.
The relevant model loaded in                                40 ms.
The boundedness of the retrieved model checked in           222 ms.
The boundedness of the relevant model checked in            0 ms.
Constructing automaton RET that encodes the retrieved model.
Automaton RET constructed in                                6 ms.
Automaton RET has 6 states and 7 transitions.
Constructing automaton REL that encodes the relevant model.
Automaton REL constructed in                                1 ms.
Automaton REL has 10 states and 14 transitions.

==================Calculating precision and recall============================

The intersection INT of RET and REL constructed in          3 ms.
Automaton INT has 8 states and 8 transitions.

A largest eigenvalue of the adjacency matrix of REL is 1.3899267936242778.
A largest eigenvalue of the adjacency matrix of REL computed in     85 ms.

A largest eigenvalue of the adjacency matrix of RET is 1.4371560431001367.
A largest eigenvalue of the adjacency matrix of RET computed in     2 ms.

A largest eigenvalue of the adjacency matrix of INT is 1.1147977972610885.
A largest eigenvalue of the adjacency matrix of INT computed in     2 ms.

Recall computed in                                                  87 ms.
Recall: 0.8020550451827885.
```

b) Partial Matching Precision and Recall Measures

To measure the partial matching precision value between the event log and model, use the option (`-pmp`) on the command line followed by the paths to log (`-rel=<path>`) and model files (`-ret=<path>`), as follows:

```
>java -jar jbpt-pm-entropia-1.5.jar -pmp -s -rel=log.xes -ret=model.pnml
```

Output Screen:(in the silent mode).

```
>java -jar jbpt-pm-entropia-1.5.jar -pmp -s -rel=log.xes -ret=model.pnml
0.8675873674841651.
```

When you replace the option (`-pmp`) with (`-pmr`), the tool measures the partial matching recall value.

```
>java -jar jbpt-pm-entropia-1.5.jar -pmr -rel=log.xes -ret=model.pnml
```

Note that the option (`-s`) is removed as the debug information and execution data are placed on the output screen.
Output Screen:

```
>java -jar jbpt-pm-entropia-1.5.jar -pmr -rel=log.xes -ret=model.pnml
================================================================================
Tool to compute quality measures for Process Mining and Process Querying ver. 1.5.
For support, please contact us at jbpt.project@gmail.com.
================================================================================
PNML format:    http://www.pnml.org/
XES format:     https://xes-standard.org/
================================================================================

Computing eigenvalue-based precision and recall based on partial matching of traces.
The technique is described in:
Artem Polyvyanyy, Anna Kalenkova. Monotone Conformance Checking for Partially
Matching Designed and Observed Processes. ICPM 2019: 81-88.
https://doi.org/10.1109/ICPM.2019.00022

Loading the retrieved model from C:\Users\halkhammash\git\codebase-master\jbpt-pm\model.pnml.
The retrieved modeln loaded in                           78 ms.
Loading the relevant model from C:\Users\halkhammash\git\codebase-master\jbpt-pm\log.xes.
The relevant model loaded in                             40 ms.
The boundedness of the relevant model checked in         0 ms.

The boundedness of the retrieved model checked in        228 ms.

Constructing automaton RET that encodes the retrieved model.
Automaton RET constructed in                             6 ms.
Automaton RET has 6 states and 7 transitions.
Constructing automaton REL that encodes the relevant model.
Automaton REL constructed in                             1 ms.
Automaton REL has 10 states and 14 transitions.
Minimizing automaton REL.
Automaton REL was minimized in                           0 ms.
The minimized version of REL has 10 states and 14 transitions.
Determinizing the minimized version of REL.
The minimized version of REL was determinized in         0 ms.
The determinized version of the minimized version of REL has 11 states and 36 transitions.
Minimizing deterministic version of automaton REL.
The deterministic version of REL was minimized in        1 ms.
The minimized deterministic version of automaton REL has 9 states and 28 transitions.
Minimizing automaton RET.
Automaton RET was minimized in                           1 ms.
The minimized version of RET has 6 states and 7 transitions.
Determinizing the minimized version of RET.
The minimized version of RET was determinized in         0 ms.
The determinized version of the minimized version of RET has 5 states and 17 transitions.
Minimizing deterministic version of automaton RET.
The deterministic version of RET was minimized in        0 ms.
The minimized deterministic version of automaton RET has 3 states and 9 transitions.
The intersection INT of RET and REL constructed in       3 ms.
Automaton INT has 8 states and 23 transitions.

A largest eigenvalue of the adjacency matrix of REL is 3.9664313615514613.
A largest eigenvalue of the adjacency matrix of REL computed in     95 ms.

A largest eigenvalue of the adjacency matrix of RET is 4.4944928370554145.
A largest eigenvalue of the adjacency matrix of RET computed in     15 ms.

A largest eigenvalue of the adjacency matrix of INT is 3.899365208677344.
A largest eigenvalue of the adjacency matrix of INT computed in     7 ms.

Recall computed in                                                  102 ms.
Recall: 0.983091563483432.
```

c) Controlled Partial Matching Precision and Recall Measures

In order to measure controlled partial matching precision and recall values, the options (`-cpmp`) and (`-cpmr`) are used, respectively. Both options should be followed by the paths to log (`-rel<path>`) and model files (`-ret<path>`); and (`-srel=<num>`) and (`-sret=<num>`) options to specify the number of allowed skips in relevant and retrieved traces.

The following command computes the controlled partial matching precision value between the event log and model, where (-cpmp) is applied with a maximum of 3 allowed skipped actions in a trace described by each of the event log and model.

```
>java -jar jbpt-pm-entropia-1.5.jar -cpmp -srel=3 -sret=3 -rel=log.xes -ret=model.pnml
```

Output Screen:

```
 >java -jar jbpt-pm-entropia-1.5.jar -cpmp -srel=3 -sret=3 -rel=log.xes -ret=model.pnml
===============================================================================
Tool to compute quality measures for Process Mining and Process Querying ver. 1.5.
For support, please contact us at jbpt.project@gmail.com.
===============================================================================
PNML format:    http://www.pnml.org/
XES format:     https://xes-standard.org/
===============================================================================

Computing eigenvalue-based precision and recall based on exact matching of traces.
The technique is described in:
Artem Polyvyanyy, Andreas Solti, Matthias Weidlich, Claudio Di Ciccio,
Jan Mendling. Monotone Precision and Recall for Comparing Executions and
Specifications of Dynamic Systems.
ACM Transactions on Software Engineering and Methodology (TOSEM) (2020)

Loading the retrieved model from C:\Users\halkhammash\git\codebase-master\jbpt-pm\model.pnml.
The retrieved modeln loaded in                          78 ms.
Loading the relevant model from C:\Users\halkhammash\git\codebase-master\jbpt-pm\log.xes.
The relevant model loaded in                            40 ms.
The boundedness of the retrieved model checked in       223 ms.
The boundedness of the relevant model checked in        0 ms.
Constructing automaton RET that encodes the retrieved model.
Automaton RET constructed in                            6 ms.
Automaton RET has 6 states and 7 transitions.
Constructing automaton REL that encodes the relevant model.
Automaton REL constructed in                            1 ms.
Automaton REL has 10 states and 14 transitions.

===================Calculating precision and recall=============================

Number of states in : REL is 17
Construction time of : REL is 4 ms.
Number of states in : RET is 18
Construction time of : RET is 1 ms.
The intersection INT of RET and REL constructed in      4 ms.
Automaton INT has 18 states and 48 transitions.

A largest eigenvalue of the adjacency matrix of REL is 3.4859606099099087.
A largest eigenvalue of the adjacency matrix of REL computed in     77 ms.

A largest eigenvalue of the adjacency matrix of RET is 3.089125399438049.
A largest eigenvalue of the adjacency matrix of RET computed in     9352 ms.

A largest eigenvalue of the adjacency matrix of INT is 2.895675670596543.
A largest eigenvalue of the adjacency matrix of INT computed in     16 ms.

Precision computed in                                               9368 ms.
Precision: 0.9373771848573396.
```

Similarly, in the following command, (-cpmr) is used to measure the controlled partial matching recall value, where the maximal number of allowed skipped actions in traces in the event log (-srel) and model (-sret=<num>) are 2 and 3, respectively.

```
>java -jar jbpt-pm-entropia-1.5.jar -cpmr -srel=2 -sret=3 -rel=log.xes -ret=model.pnml
```

Output Screen:

```
>java -jar jbpt-pm-entropia-1.5.jar -cpmr -srel=2 -sret=3 -rel=log.xes -ret=model.pnml
===============================================================================
Tool to compute quality measures for Process Mining and Process Querying ver. 1.5.
For support, please contact us at jbpt.project@gmail.com.
===============================================================================
PNML format:    http://www.pnml.org/
```

```
XES format:     https://xes-standard.org/
================================================================================

Computing eigenvalue-based precision and recall based on exact matching of traces.
The technique is described in:
Artem Polyvyanyy, Andreas Solti, Matthias Weidlich, Claudio Di Ciccio,
Jan Mendling. Monotone Precision and Recall for Comparing Executions and
Specifications of Dynamic Systems.
ACM Transactions on Software Engineering and Methodology (TOSEM) (2020)

Loading the retrieved model from C:\Users\halkhammash\git\codebase-master\jbpt-pm\model.pnml.
The retrieved modeln loaded in                              78 ms.
Loading the relevant model from C:\Users\halkhammash\git\codebase-master\jbpt-pm\log.xes.
The relevant model loaded in                                40 ms.
The boundedness of the retrieved model checked in           223 ms.
The boundedness of the relevant model checked in            0 ms.
Constructing automaton RET that encodes the retrieved model.
Automaton RET constructed in                                6 ms.
Automaton RET has 6 states and 7 transitions.
Constructing automaton REL that encodes the relevant model.
Automaton REL constructed in                                1 ms.
Automaton REL has 10 states and 14 transitions.

===================Calculating precision and recall=============================

Number of states in : REL is 17
Construction time of : REL is 4 ms.
Number of states in : RET is 18
Construction time of : RET is 1 ms.
The intersection INT of RET and REL constructed in          4 ms.
Automaton INT has 18 states and 48 transitions.

A largest eigenvalue of the adjacency matrix of REL is 3.4859606099099087.
A largest eigenvalue of the adjacency matrix of REL computed in     77 ms.

A largest eigenvalue of the adjacency matrix of RET is 3.089125399438049.
A largest eigenvalue of the adjacency matrix of RET computed in     9352 ms.

A largest eigenvalue of the adjacency matrix of INT is 2.895675670596543.
A largest eigenvalue of the adjacency matrix of INT computed in     16 ms.

Recall computed in                                                  84 ms.
Recall: 0.9851977981870396.
```

Stochastic Conformance Checking Measures

a) Stochastic Precision and Recall Measures

To compute the stochastic precision value between the event log and the process model, use the option (-sp) as follows.

```
>java -jar jbpt-pm-entropia-1.5.jar -sp  -rel=log.xes -ret=automaton.json
```

Output Screen:

```
output
```

Use the option (-sr) instead of (-sp) in order to get the stochastic recall value between the event log and process model.

```
>java -jar jbpt-pm-entropia-1.5.jar -sr  -rel=log.xes -ret=automaton.json
```

Output Screen:

```
output
```

b) Entropic Relevance Measure

You can measure relevance of a stochastic process model to an event log using the option (-r), as the following command shows. Note that the retrieved model is specified to the stochastic process model, i.e. the stochastic deterministic finite automaton (SDFA), in JSON format.

```
>java -jar jbpt-pm-entropia-1.5.jar -r -rel=log.xes -ret=automaton.json
```

Output Screen:

```
>java -jar jbpt-pm-entropia-1.5.jar -r -rel=log.xes -ret=automaton.json
{coverage=0.2857142857142857, numberOfTransitions=7, numberOfNonFittingTraces=5, numberOfTraces=7,
relevance=11.653256727657693, numberOfStates=6, costOfBackgroundModel=67.20902501875005}
```

Bootstrap Generalization Measure

To compute the generalization of a process model with bootstrap generalization approach [7], use the option (-bgen) as follows.

```
>java -jar jbpt-pm-entropia-1.5.jar -bgen -rel=log3.xes -ret=model3.json
```

Output Screen:

```
Computing bootstrap generalization.
The technique is described in:
Artem Polyvyanyy, Alistair Moffat, Luciano Garcia-Bonuelos. Bootstrapping
Generalization of Process Models Discovered from Event Data. CAiSE 2022

===========================Calculating generalization===========================

Sample Size = 264
Number of Log Generations = 16
Crossover Subtrace Length = 2
Breeding Probability = 1.0
Threshold for confidence interval of bootstrap samples = 0.01

Model-log precision and recall calculated for bootstrap sample 1: 0.8640223021976667, 0.948122502132
Model-log precision and recall calculated for bootstrap sample 2: 0.846835226895293, 0.929856208666 3
Model-log precision and recall calculated for bootstrap sample 3: 0.8789360974064723, 0.970371448763
Model-log precision and recall calculated for bootstrap sample 4: 0.8640223021901551, 0.948151229748
Model-log precision and recall calculated for bootstrap sample 5: 0.8399359447535413, 0.927314126686
Model-log precision and recall calculated for bootstrap sample 6: 0.8468352269011715, 0.941841673297
Model-log precision and recall calculated for bootstrap sample 7: 0.8789360974064715, 0.937715554591
Model-log precision and recall calculated for bootstrap sample 8: 0.8789360974064718, 0.942662351920
Model-log precision and recall calculated for bootstrap sample 9: 0.8789360974064719, 0.937715554591
Model-log precision and recall calculated for bootstrap sample 10: 0.8640223021081593, 0.93682372879
Model-log precision and recall calculated for bootstrap sample 11: 0.8789360974064686, 0.95287533400

===========================Calculated generalization===========================

Generalization calculated in 6296 ms with 11 samples.
Model-system precision: 0.8654867083707584 +/- 0.009628352340204586
Model-system recall: 0.9430408830179853 +/- 0.007616113385628391
```

Note that the model you want to analyze should be provided in JSON format. Each DFG node has a string label for the activity name, a numerical frequency for activity execution, and a unique identifier. Arcs connect nodes by their source and target numbers, along with their occurrence frequency. The start and end nodes should be labeled `INPUT` and `OUTPUT` to indicate process boundaries. Refer to Listing 1 for a sample representation.

Listing 1: Sample JSON representation.

```
{ "nodes": [
    { "id": 1, "label": "INPUT",  "freq": 30 },
    { "id": 2, "label": "A",      "freq": 30 },
    { "id": 3, "label": "B",      "freq": 30 },
    { "id": 4, "label": "OUTPUT", "freq": 30 }
  ],
  "arcs": [
    { "from": 1, "to": 2, "freq": 30 },
    { "from": 2, "to": 3, "freq": 30 },
    { "from": 3, "to": 4, "freq": 18 }
  ]
}
```

You can adjust the underlying bootstrapping process with several parameters for bootstrap generalization estimation, including sample size (`n`), number of samples (`m`), number of log generations (`g`), crossover subtrace length (`k`), breeding probability (`p`), and threshold for confidence interval of bootstrap samples (`ep`). These parameters, detailed in [7], are optional for tool usage.

```
>java -jar jbpt-pm-entropia-1.5.jar -bgen -rel=log3.xes -ret=model3.json -s -m=1000
-ep=0.005
```

Output Screen:

```
> java -jar jbpt-pm-entropia-1.7.jar -bgen -rel=examples/log3.xes -ret=examples/model3.json -s -m=1
0.8646947781195624, 0.9443517823270182
```

Here, bootstrapping continues until 1,000 samples are reached, unless the confidence interval for both precision and recall of the bootstrap samples falls below 0.005, triggering early termination. And this calculation will be done in silent mode, showing only the final result.
Another configuration might be as follows:

```
>java -jar jbpt-pm-entropia-1.5.jar -bgen -rel=log3.xes -ret=model3.json -n=1000 -m=20
-p=0.5
```

Output Screen:

```
Computing bootstrap generalization.
The technique is described in:
Artem Polyvyanyy, Alistair Moffat, Luciano Garcia-Bonuelos. Bootstrapping
Generalization of Process Models Discovered from Event Data. CAiSE 2022

==========================Calculating generalization==========================

Sample Size = 1000
Number of Samples = 20
Number of Log Generations = 16
Crossover Subtrace Length = 2
Breeding Probability = 0.5

Model-log precision and recall calculated for bootstrap sample 1: 0.8640223023960756, 0.93160785144
Model-log precision and recall calculated for bootstrap sample 2: 0.8882953280705439, 0.94160554280
Model-log precision and recall calculated for bootstrap sample 3: 0.8640223022539603, 0.92180544515
Model-log precision and recall calculated for bootstrap sample 4: 0.8640223022681677, 0.94815122983
Model-log precision and recall calculated for bootstrap sample 5: 0.8640223022654226, 0.93682372896
Model-log precision and recall calculated for bootstrap sample 6: 0.864022302362209, 0.948151229936
Model-log precision and recall calculated for bootstrap sample 7: 0.8640223022423997, 0.96679395265
Model-log precision and recall calculated for bootstrap sample 8: 0.864022302265727, 0.948151229830
Model-log precision and recall calculated for bootstrap sample 9: 0.8640223022506948, 0.93160785128
Model-log precision and recall calculated for bootstrap sample 10: 0.86402230226115, 0.948151229825
Model-log precision and recall calculated for bootstrap sample 11: 0.8789360974064712, 0.97037144876
```

```
Model-log precision and recall calculated for bootstrap sample 12: 0.8640223022574599, 0.94815122982
Model-log precision and recall calculated for bootstrap sample 13: 0.8789360974064726, 0.93764421014
Model-log precision and recall calculated for bootstrap sample 14: 0.8789360974064709, 0.94266235192
Model-log precision and recall calculated for bootstrap sample 15: 0.8640223022271509, 0.95390617776
Model-log precision and recall calculated for bootstrap sample 16: 0.864022302276446, 0.92180428496
Model-log precision and recall calculated for bootstrap sample 17: 0.878936097406473, 0.95287509495
Model-log precision and recall calculated for bootstrap sample 18: 0.8640223022513908, 0.94815122981
Model-log precision and recall calculated for bootstrap sample 19: 0.8789360974064728, 0.93771555454
Model-log precision and recall calculated for bootstrap sample 20: 0.8640223022648623, 0.95390617780

===========================Calculated generalization===========================

Generalization calculated in 12226 ms with 20 samples.
Model-system precision: 0.8689644023473011 +/- 0.0036445059613284285
Model-system recall: 0.9445018526122011 +/- 0.005750771071582731
```

This configuration generates 20 bootstrap samples, each containing 1,000 traces, with new traces being generated 50% of the time, and existing traces used the remaining 50%.

Log Representativeness Measures

Entropia allows users to evaluate an event log with respect to its generative system based on completeness, coverage, [1] and log representativeness approximation (LRA) [3]. This analysis can be done for specific event data aspects: activities, directly-follows relations, or traces.

a) Completeness

You can calculate the log completeness [1] with the following command.

```
>java -jar jbpt-pm-entropia-1.5.jar -l=log3.xes -com
```

Output Screen:

```
Computing event log statistics
Completeness and Coverage are described in:
M. Kabierski, M. Richter, and M. Weidlich, "Addressing the
log representativeness problem using species discovery," in ICPM 2023

============================= Event Log Statistics =============================

--------------------------- Activity-based Analysis ----------------------------
Activities: [A, B, C]
Number of distinct activities: 3
Number of total activities: 121
Completeness: 1.0

------------------- Directly-follows-relation-based Analysis -------------------
Number of distinct DF-relations: 5
Number of total DF-relations: 88
Completeness: 1.0

----------------------------- Trace-based Analysis -----------------------------
Number of distinct traces: 5
Number of total traces: 33
Completeness: 1.0
```

b) Coverage

To calculate the log coverage for activities and directly-follows relations only, use the following command.

```
>java -jar jbpt-pm-entropia-1.5.jar -l=log3.xes -cov -act -dfr
```

Output Screen:

```
Computing event log statistics
Completeness and Coverage are described in:
M. Kabierski, M. Richter, and M. Weidlich, "Addressing the
log representativeness problem using species discovery," in ICPM 2023

============================ Event Log Statistics ============================

-------------------------- Activity-based Analysis ----------------------------
Activities: [A, B, C]
Number of distinct activities: 3
Number of total activities: 121
Coverage: 1.0

------------------ Directly-follows-relation-based Analysis -------------------
Number of distinct DF-relations: 5
Number of total DF-relations: 88
Coverage: 1.0
```

c) LRA

To analyze the trace-based log representativeness approximation in silent mode, execute this command.

```
>java -jar jbpt-pm-entropia-1.5.jar -l=log3.xes -lra -tr -s
```

Output Screen:

```
> java -jar jbpt-pm-entropia-1.7.jar -l=examples/log3.xes -lra -tr -s
0.8484848484848485
```

References

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

[2] Anna Kalenkova and Artem Polyvyanyy. A spectrum of entropy-based precision and recall measurements between partially matching designed and observed processes. In *ICSOC*, LNCS. Springer, 2020. In Press.

[3] Anandi Karunaratne, Artem Polyvyanyy, and Alistair Moffat. The role of log representativeness in estimating generalization in process mining. IEEE, 2024. to appear.

[4] Sander J. J. Leemans and Artem Polyvyanyy. Stochastic-aware conformance checking: An entropy-based approach. In *Advanced Information Systems Engineering*, volume 12127 of *LNCS*, pages 217–233. Springer, 2020.

[5] Artem Polyvyanyy and Anna Kalenkova. Monotone conformance checking for partially matching designed and observed processes. In *International Conference on Process Mining, ICPM 2019, Aachen, Germany, June 24-26, 2019*, pages 81–88. IEEE, jun 2019.

[6] Artem Polyvyanyy, Alistair Moffat, and Luciano García-Bañuelos. An entropic relevance measure for stochastic conformance checking in process mining. *CoRR*, abs/2007.09310, 2020.

[7] Artem Polyvyanyy, Alistair Moffat, and Luciano García-Bañuelos. Bootstrapping generalization of process models discovered from event data. pages 36–54, 2022.

[8] Artem Polyvyanyy, Andreas Solti, Matthias Weidlich, Claudio Di Ciccio, and Jan Mendling. Monotone precision and recall measures for comparing executions and specifications of dynamic systems. *ACM Trans. Softw. Eng. Methodol.*, 29(3), June 2020.