CDLG: A Tool for the Generation of Event Logs with Concept Drifts

TUTORIAL

This tutorial provides a step-by-step demonstration of how to use the CDLG tool. The tool itself can be accessed through our repository: https://gitlab.uni-mannheim.de/processanalytics/cdlg_tool. Before using the tool, clone project and follow installation guidance provided by the repository.

Reference: "*CDLG: A Tool for the Generation of Event Logs with Concept Drifts"* by Justus Grimm, Alexander Kraus, and Han van der Aa, submitted to the demo track of BPM 2022.

Execution modes

The CDLG tool can be used in three different execution modes: terminal mode, parameter-file mode, and via a python package.

1. Terminal mode

The terminal mode is an advanced mode that provides the highest degree of customization to a user. To demonstrate the flexibility of the tool, we consider a step-by-step demonstration of the tool for a drift scenario that is based on a single sudden drift.

1. To start the terminal mode, run the python file "start_generator_terminal.py" via a terminal:

```
cdlg_tool>python start_generator_terminal.py
```

2. Specify if there are models that should be imported or if the tool should generate random models:

```
Import models or use randomly generated models [import, random]: random
```

The tool can work with user given block-structured BPMN or Petri net models or it can randomly generate required number of models that are needed to generate a target drift scenario.

3. Select if the new model after the drift should be an evolution of the randomly generated one or if it should be another randomly generated model:

```
Evolution of one randomly generated model or use of two randomly generated models [one_model, two_models]: one_model
```

A model evolution allows to control the changes in terms of activities and control flow operators that will be affected by the drift.

4. Choose if the generation of a random model should be controlled or not:

```
Do you want to adjust the various settings/parameters for the process tree, which will be used to generate the model randomly [yes, no]?
```

The tool allows to control the generation of a random process tree using the PTandLogGenerator tool.

```
Do you want to adjust the various settings/parameters for the process tree, which will be used to generate the model randomly [yes, no]? yes
MODE: most frequent number of visible activities (default 20): 20
MIN: minimum number of visible activities (default 10): 10
MAX: maximum number of visible activities (default 30): 30
SEQUENCE: probability to add a sequence operator to tree (default 0.25): 0.25
CHOICE: probability to add a choice operator to tree (default 0.25): 0.25
PARALLEL: probability to add a parallel operator to tree (default 0.25): 0.25
LOOP: probability to add a loop operator to tree (default 0.25): 0.25
OR: probability to add an or operator to tree (default 0): 0
SILENT: probability to add silent activity to a choice or loop operator (default 0.25): 0.25
DUPLICATE: probability to duplicate an activity label (default 0): 0
LT_DEPENDENCY: probability to add a random dependency to the tree (default 0): 0
INFREQUENT: probability to make a choice have infrequent paths (default 0.25): 0.25
NO_MODELS: number of trees to generate from model population (default 10): 10
UNFOLD: whether or not to unfold loops in order to include choices underneath in dependencies: 0=False, 1=True
        if lt_dependency <= 0: this should always be 0 (False)
        if lt_dependency > 0: this can be 1 or 0 (True or False) (default 10): 0
MAX_REPEAT: maximum number of repetitions of a loop (only used when unfolding is True) (default 10): 10
```

A detailed description of all parameters can be found in the scientific paper 'PTandLogGenerator: A Generator for Artificial Event Data' (BPM Demos, 2016). The option “no” allows to skip these configurations and to use a default setting that lead to a simple, middle, or complex model.

```
Do you want to adjust the various settings/parameters for the process tree, which will be used to generate the model randomly [yes, no]? no
Complexity of the process tree to be generated [simple, middle, complex]: middle
```

5. Next, provide input for a target drift scenario by specifying the drift type, the total number of trace, and the moment, when the drift should occur:

```
--- INPUT DRIFT ---
Type of concept drift [sudden, gradual, recurring, incremental]: sudden
Number of traces in the event log (x >= 100): 20000
Starting point of the drift (0 < x < 1): 0.5
```

Depending on the drift type (sudden, gradual, recurring, incremental), different parameters need to be provided.

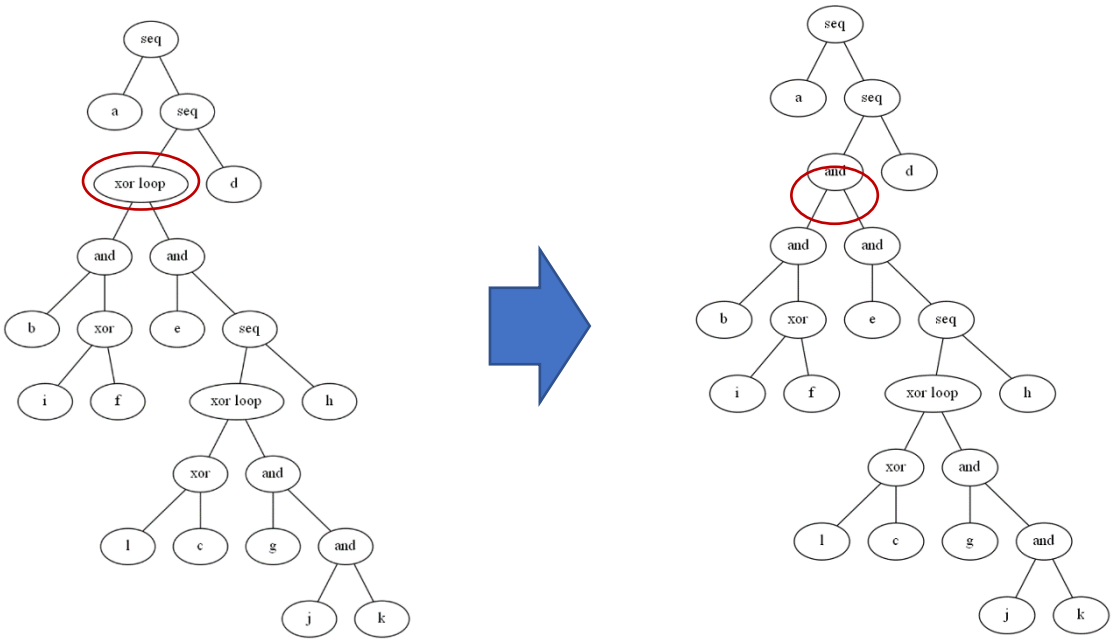
6. Select the model evolution option:

```
Controlled or random evolution of the process tree version [controlled, random]: controlled
```

If needed for the concept drift scenario, the terminal mode provides a full control over the evolution of a process model using the four main transformation steps: change activity, change operator, change tree fragment, delete silent activities. Using these steps, the user can iteratively modify any generate process or an imported process model that is converted to a process tree (The imported model should be in a block-structured format). This is done using a visualization of a process tree and an interactive dialog with the tool through the terminal.

For example, the diagonal below transforms the process tree on the left-hand side to the one on the right-hand side by replacing the “xor loop” operator with an “and“ operator.

```
Operator for the 1. intermediate evolution of the process tree version [change_activity, change_operator, change_tree_fragment, delete_silent_ac]: change operator
Procedure for changing operator/s in the process tree version [replace_op, swap_ops]: replace_op
Existing operator to be replaced [xor, seq, or, xor loop, and]: xor loop
Depth of the node with the operator to be changed (int): 2
New operator [xor, seq, or, xor loop, and]: and
Do you want to continue with the evolution of the version [yes, no]:
```



The user can choose a random process evolution and specify the proportion of activities that should be affected by a random evolution:

```
Controlled or random evolution of the process tree version [controlled, random]: random
Proportion of the activities in the process tree to be affected by random evolution (0 < x < 1): 0.2
```

7. Select if another drift should be added to the log or not

```
Do you want to add an additional drift to the event log [yes, no]?
```

8. The final configuration allows to add noise to the event log

```
Do you want to add noise to the event log [yes, no]? yes

--- INFORMATION FOR THE INTRODUCTION OF NOISE ---
The noise will be randomly distributed in the sector to be determined.
The proportion of noise for this sector can also be predefined.
This gives the possibility to either disguise the drift by placing the noise around/inside the drift or to fake another drift by placing the noise away from the drift.
The following two types of noise exist:
        changed_model: the initial model version is changed randomly, whereby the proportion of changes in the version can be specified.
        random_model: a completely new model is used, which means that there is little or no similarity to the other model versions.
The created model, which will appear as an image, will be used to generate traces introduced in the event log as noise.

Type of model for generating the noise [changed_model, random_model]: random_model
Start point of noise in the event log (0 <= x <= 1): 0.7
End point of noise in the event log (0 <= x <= 1): 0.8
Proportion of noise in the set sector of the event log (0 < x < 0.5): 0.1
```

The noise can be created from an existing model or from a completely new randomly generated model. Then, the user can select a time moment when the noise should be inserted. This can happen during a drift period, directly after or before, or it can be inserted independent from the drift period.

Afterwards, the created event log is stored in the folder “data/generated_logs/” with a certain id.

```
Resulting event log stored as data/generated_logs\terminal_log_1653481470.xes
```

All relevant drift information is stored as a log attributed in the generated xes-file.

2. Parameter-file mode

3. Python package

BUGS!!!

```
Import models or use randomly generated models [import, random]: random
Evolution of one randomly generated model or use of two randomly generated models [one_model, two_models]: one_model
Do you want to adjust the various settings/parameters for the process tree, which will be used to generate the model randomly [yes, no]? yes
MODE: most frequent number of visible activities (default 20):
Wrong input! It has to be an integer.
MODE: most frequent number of visible activities (default 20): 20
MIN: minimum number of visible activities (default 10):
```

```
Import models or use randomly generated models [import, random]: random
Evolution of one randomly generated model or use of two randomly generated models [one_model, two_models]: one_model
Do you want to adjust the various settings/parameters for the process tree, which will be used to generate the model randomly [yes, no]? yes
MODE: most frequent number of visible activities (default 20):
Wrong input! It has to be an integer.
MODE: most frequent number of visible activities (default 20): 20
MIN: minimum number of visible activities (default 10): 10
MAX: maximum number of visible activities (default 30): 20
SEQUENCE: probability to add a sequence operator to tree (default 0.25):
Default: 0.5
CHOICE: probability to add a choice operator to tree (default 0.25):
Default: 0.5
PARALLEL: probability to add a parallel operator to tree (default 0.25):
Default: 0.5
LOOP: probability to add a loop operator to tree (default 0.25):
Default: 0.5
OR: probability to add an or operator to tree (default 0):
Default: 0.5
SILENT: probability to add silent activity to a choice or loop operator (default 0.25):
Default: 0.5
DUPLICATE: probability to duplicate an activity label (default 0):
Default: 0.5
LT_DEPENDENCY: probability to add a random dependency to the tree (default 0):
Default: 0.5
INFREQUENT: probability to make a choice have infrequent paths (default 0.25):
Default: 0.5
```

```
Import models or use randomly generated models [import, random]: random
Evolution of one randomly generated model or use of two randomly generated models [one_model, two_models]: one_model
Do you want to adjust the various settings/parameters for the process tree, which will be used to generate the model randomly [yes, no]? yes
MODE: most frequent number of visible activities (default 20): 20
MIN: minimum number of visible activities (default 10): 10
MAX: maximum number of visible activities (default 30): 30
SEQUENCE: probability to add a sequence operator to tree (default 0.25): 0.25
CHOICE: probability to add a choice operator to tree (default 0.25): 0.25
PARALLEL: probability to add a parallel operator to tree (default 0.25): 0.25
LOOP: probability to add a loop operator to tree (default 0.25): 0.25
OR: probability to add an or operator to tree (default 0): 0
SILENT: probability to add silent activity to a choice or loop operator (default 0.25): 0.25
DUPLICATE: probability to duplicate an activity label (default 0): 0
LT_DEPENDENCY: probability to add a random dependency to the tree (default 0): 0
INFREQUENT: probability to make a choice have infrequent paths (default 0.25): 0.25
NO_MODELS: number of trees to generate from model population (default 10): 10
UNFOLD: whether or not to unfold loops in order to include choices underneath in dependencies: 0=False, 1=True
        if lt_dependency <= 0: this should always be 0 (False)
        if lt_dependency > 0: this can be 1 or 0 (True or False) (default 10):
```

```
NO_MODELS: number of trees to generate from model population (default 10): 1
UNFOLD: whether or not to unfold loops in order to include choices underneath in dependencies: 0=False, 1=True
        if lt_dependency <= 0: this should always be 0 (False)
        if lt_dependency > 0: this can be 1 or 0 (True or False) (default 10):
Wrong input! It has to be an integer.
```