MINERful tutorial

Cecilia Iacometta, c.iacometta@students.uu.nl
Claudio Di Ciccio, c.diciccio@uu.nl
Utrecht University, Netherlands

June 26, 2025

Welcome to the tutorial on MINERful. We assume you have already downloaded the tool from `https://github.com/cdc08x/MINERful/`
To execute the JAR, make sure to have JRE 22+ installed. Any version beyond 11 can work but, in that case, you should recompile the sources to obtain a compatible JAR.

1 What is MINERful?

MINERful is a process mining, simulation, and analysis tool for declarative process specifications.
The main functionalities of MINERful are:

1. Discovery of declarative specifications of process control flows out of event logs. Event logs can be stored as XES, MXML, or text files (a collection of strings, in which every character is considered as an event, every line as a trace);
2. Simulation of declarative specifications of process control flows and creation of synthetic event logs. The input process specification can be given as a JSON file. The event logs can be exported as XES, MXML, or text files;
3. Simplification and inconsistency removal of declarative process specifications;
4. Discovery of declarative specifications of process control flows from windows over event logs.
5. Fitness checker of passed declarative process specifications with event logs.

For more information about MINERful, please visit the Wiki at this address:
`https://github.com/cdc08x/MINERful/wiki`

2 Introduction

The easiest way to launch MINERful if you use a Unix/Linux or a Mac machine is to run the `.sh` files.

1. To launch the miner: `run-MINERful.sh`
2. To create synthetic logs: `run-MINERfulEventLogMaker.sh`
3. To process and simplify specifications: `run-MINERfulSimplifier.sh`
4. To launch the miner on sublogs as windows over an original event log: `run-MINERfulSlider.sh`
5. To check the fitness of a passed declarative process specification with an event log: `run-MINERfulFitnessChecker.sh`

If you use a Windows machine, please run the following command from within a terminal (prompt) locating the current directory in the folder where the MINERful jar lies.

COMMAND LINE: `java -cp ".\lib\*;MINERful.jar" <MAIN_CLASS>`

where `<MAIN_CLASS>` is either of the following, depending on the above functionalities you wish to employ:

1. `minerful.MinerFulMinerStarter`
2. `minerful.MinerFulLogMakerStarter`
3. `minerful.MinerFulSimplificationStarter`
4. `minerful.MinerFulMinerSlider`
5. `minerful.MinerFulFitnessCheckerStarter`

Each of those launchers can be invoked with the `-h` parameter, to get guidance on how to use them. An explanation of all possible parameters that can be passed will be displayed (beware, they are quite an amount so take your time).

For example, MINERful allows the setting of thresholds for both event and trace-based quality measures (confidence, support, and coverage) while discovering constraints from event logs or simplifying existing specifications. The obtained results can be saved in the form of CSV or JSON files. Furthermore, with MINERful we can create a finite state automaton in DOT of the discovered process specification. To open GraphViz DOT files, please check out the GraphViz toolkit (an online viewer is also available at `https://dreampuf.github.io/GraphvizOnline/`).

3 Creating event logs from process specifications

Starting from a given process specification MINERful permits to create synthetic event logs. The input process specification can be given as a JSON file. The event logs can be exported as XES, MXML files or text files (a collection of strings, in which every character is considered as an event, every line as a trace). As an example, we start with the following JSON process specification describing the admission process to a University. Next, you can see the constraints to be fed as an input to MINERful in a JSON format below. Please save the lines below it as `specifications/university.json`

```
{
  "name": "Admission to university process",
  "constraints": [
    {"template": "init", "parameters":[["Create a candidate account"]]},
    {"template": "atmost1", "parameters":[["Create a candidate account"]]},
    {"template": "precedence", "parameters": [["Create a candidate account"],["Register for selection round"]]},
    {"template": "alternatesuccession", "parameters": [["Register for selection round"],["Enter evaluation phase"]]},
    {"template": "precedence", "parameters": [["Upload admission test score"],["Enter evaluation phase"]]},
    {"template": "alternateprecedence", "parameters": [["Enter evaluation phase"],["Receive rejection notification"]]},
    {"template": "alternateprecedence", "parameters": [["Enter evaluation phase"],["Receive admission notification"]]},
    {"template": "notresponse", "parameters": [["Receive admission notification"],["Receive rejection notification"]]},
    {"template": "precedence", "parameters": [["Receive admission notification"],["Pre-enrol in the program"]]},
    {"template": "atmost1", "parameters":[["Pre-enrol in the program"]]},
    {"template": "precedence", "parameters": [["Pay subscription fee"],["Pre-enrol in the program"]]},
    {"template": "chainresponse", "parameters": [["Pay subscription fee"],["Pre-enrol in the program"]]},
    {"template": "precedence", "parameters": [["Pre-enrol in the program"],["Enrol in the program"]]},
    {"template": "precedence", "parameters": [["Upload certificates"],["Enrol in the program"]]},
    {"template": "atmost1", "parameters": [["Enrol in the program"]]}
  ]
}
```

Please find the diagram of that specification below (taken from `http://dx.doi.org/10.1007/978-3-031-08848-3_4`).

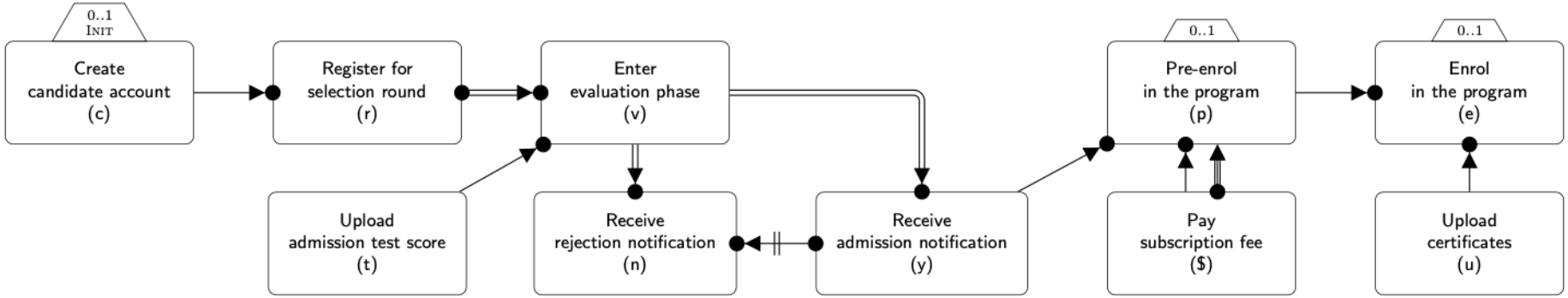

Figure 1: Admission to university specification diagram from `http://dx.doi.org/10.1007/978-3-031-08848-3_4`

3.1 MINERful event log maker and discovery

By running MINERful as an event log maker, we can create a synthetic event log starting from the above process specification with (say) 10000 traces constituted by a number of events between 2 and 24.

COMMAND LINE (Linux/Mac):

```
./run-MINERfulEventLogMaker.sh --input-specification-file specifications/university.json --size 10000 --minlen 2 --maxlen 24
--out-log encoding xes --out-log-file logs/university10000.xes
```

COMMAND LINE (Windows):

```
java -cp ".\lib\*;MINERful.jar" minerful.MinerFulLogMakerStarter --input-specification-file specifications/university.json
--size 10000 --minlen 2 --maxlen 24 --out-log-encoding xes --out-log-file logs/university10000.xes
```

After the creation of the event log, we can proceed with a process discovery task to check whether the simulated log matches the input process specification. Here we set the maximum level of pruning (template-hierarchy based simplification plus conflict and double-pass redundancy check, i.e., `-prune 'hierarchyconflictredundancydouble'`) and the following thresholds: event-based support 0.02, event-based coverage 0.02, event-based confidence 1, trace-based support 0.125, trace-based coverage 0.125, and trace-based confidence 1. Also, we request the output formatted as a CSV file.

COMMAND LINE (Linux/Mac):

```
./run-MINERful.sh -iLF logs/university10000.xes -prune hierarchyconflictredundancydouble
--support 0.02 --confidence 1.0 --coverage 0.02 --trace-support 0.125 --trace-confidence 1.0 --trace-coverage 0.125
-oCSV specifications/university10000.csv
```

COMMAND LINE (Windows):

```
java -cp ".\lib\*;MINERful.jar" minerful.MinerFulMinerStarter -iLF logs/university10000.xes -prune hierarchyconflictredundancydouble
--support 0.02 --confidence 1.0 --coverage 0.02 --trace-support 0.125 --trace-confidence 1.0 --trace-coverage 0.125
-oCSV specifications/university10000.csv
```

Constraint	Template	Activation	Target	Confidence	Coverage	Support	Trace confidence	Trace coverage	Trace support
AtMost1(Create a candidate account)	AtMost1		Create a candidate account	1.000	0.174	0.174	1.000	1.000	1.000
Init(Create a candidate account)	Init		Create a candidate account	1.000	1.000	1.000	1.000	1.000	1.000
NotChainResponse(Create a candidate account, Enrol in the program)	NotChainResponse	Create a candidate account	Enrol in the program	1.000	0.174	0.174	1.000	1.000	1.000
NotChainResponse(Create a candidate account, Pay subscription fee)	NotChainResponse	Create a candidate account	Pay subscription fee	1.000	0.174	0.174	1.000	1.000	1.000
NotChainResponse(Create a candidate account, Pre-enrol in the program)	NotChainResponse	Create a candidate account	Pre-enrol in the program	1.000	0.174	0.174	1.000	1.000	1.000
NotChainResponse(Create a candidate account, Receive admission notification)	NotChainResponse	Create a candidate account	Receive admission notification	1.000	0.174	0.174	1.000	1.000	1.000
NotChainResponse(Create a candidate account, Receive rejection notification)	NotChainResponse	Create a candidate account	Receive rejection notification	1.000	0.174	0.174	1.000	1.000	1.000
AtMost1(Enrol in the program)	AtMost1		Enrol in the program	1.000	0.174	0.174	1.000	1.000	1.000
NotChainResponse(Enter evaluation phase, Pre-enrol in the program)	NotChainResponse	Enter evaluation phase	Pre-enrol in the program	1.000	0.107	0.107	1.000	0.504	0.504
AlternatePrecedence(Register for selection round, Enter evaluation phase)	AlternatePrecedence	Enter evaluation phase	Register for selection round	1.000	0.107	0.107	1.000	0.504	0.504
Precedence(Upload admission test score, Enter evaluation phase)	Precedence	Enter evaluation phase	Upload admission test score	1.000	0.107	0.107	1.000	0.504	0.504
AtMost1(Pay subscription fee)	AtMost1		Pay subscription fee	1.000	0.174	0.174	1.000	1.000	1.000
NotChainSuccession(Pay subscription fee, Enter evaluation phase)	NotChainSuccession	Pay subscription fee	Enter evaluation phase	1.000	0.114	0.114	1.000	0.504	0.504
NotChainSuccession(Pay subscription fee, Register for selection round)	NotChainSuccession	Pay subscription fee	Register for selection round	1.000	0.114	0.114	1.000	0.504	0.504
NotChainSuccession(Pay subscription fee, Upload admission test score)	NotChainSuccession	Pay subscription fee	Upload admission test score	1.000	0.288	0.288	1.000	0.811	0.811
AtMost1(Pre-enrol in the program)	AtMost1		Pre-enrol in the program	1.000	0.174	0.174	1.000	1.000	1.000
AtMost3(Receive admission notification)	AtMost3		Receive admission notification	1.000	0.174	0.174	1.000	1.000	1.000
AtMost3(Receive rejection notification)	AtMost3		Receive rejection notification	1.000	0.174	0.174	1.000	1.000	1.000
AlternateResponse(Register for selection round, Enter evaluation phase)	AlternateResponse	Register for selection round	Enter evaluation phase	1.000	0.107	0.107	1.000	0.504	0.504
NotChainResponse(Register for selection round, Pre-enrol in the program)	NotChainResponse	Register for selection round	Pre-enrol in the program	1.000	0.107	0.107	1.000	0.504	0.504
NotChainResponse(Upload admission test score, Pre-enrol in the program)	NotChainResponse	Upload admission test score	Pre-enrol in the program	1.000	0.280	0.280	1.000	0.811	0.811
NotChainPrecedence(Pay subscription fee, Upload certificates)	NotChainPrecedence	Upload certificates	Pay subscription fee	1.000	0.276	0.276	1.000	0.701	0.701
NotChainResponse(Upload certificates, Pre-enrol in the program)	NotChainResponse	Upload certificates	Pre-enrol in the program	1.000	0.276	0.276	1.000	0.701	0.701

Figure 2: Discovered process specification table with quality measures

3.2 MINERful event log maker and discovery with violated constraints

By running MINERful as an event log maker, we can also create event logs partially satisfying a given process specification. We can create a second synthetic event log with 10000 traces constituted by a number of events between 2 and 24. In this log, we let *AtMost1(Create a candidate account)* be violated in 2000 traces. Next, you can see the constraints to be violated in a JSON format below. Please save the lines below it as `specifications/constraints.json` should look as follows:

```
{
"name": "Admission to university process constraints to be violated",
    "constraints": [
        {"template": "atmost1", "parameters":[["Create a candidate account"]]}
    ]
}
```

COMMAND LINE (Linux/Mac):

```
./run-MINERfulEventLogMaker.sh --input-specification-file specifications/university.json --size 10000 --minlen 2 --maxlen 24
--out-log-encoding xes --out-log-file logs/university10000withviolation.xes --sizeviol 2000 --viol-const-file specifications/constraints.json
```

COMMAND LINE (Windows):

```
java -cp ".\lib\*;MINERful.jar" minerful.MinerFulLogMakerStarter --input-specification-file specifications/university.json
--size 10000 --minlen 2 --maxlen 24 --out-log-encoding xes --out-log-file logs/university10000withviolation.xes
--sizeviol 2000 --viol-const-file specifications/constraints.json
```

After the creation of this second event log with noise, we can proceed with a second process discovery task to check how far the simulated dataset matches the input process specification and what is the effect of the noise in the discovery.

COMMAND LINE (Linux/Mac):

```
./run-MINERful.sh -iLF logs/university10000withviolation.xes -prune hierarchyconflictredundancydouble
--support 0.02 --confidence 1.0 --coverage 0.02 --trace-support 0.125 --trace-confidence 1.0 --trace-coverage 0.125
-oCSV specifications/university10000withviolation.csv
```

COMMAND LINE (Windows):

```
java -cp ".\lib\*;MINERful.jar" minerful.MinerFulMinerStarter -iLF logs/university10000withviolation.xes -prune hierarchyconflictredundancydouble
--support 0.02 --confidence 1.0 --coverage 0.02 --trace-support 0.125 --trace-confidence 1.0 --trace-coverage 0.125
-oCSV specifications/university10000withviolation.csv
```

Constraint	Template	Activation	Target	Confidence	Coverage	Support	Trace confidence	Trace coverage	Trace support
Init(Create a candidate account)	Init		Create a candidate account	1.000	1.000	1.000	1.000	1.000	1.000
NotChainResponse(Create a candidate account, Pre-enrol in the program)	NotChainResponse	Create a candidate account	Pre-enrol in the program	1.000	0.219	0.219	1.000	1.000	1.000
AtMost1(Enrol in the program)	AtMost1		Enrol in the program	1.000	0.156	0.156	1.000	1.000	1.000
NotChainResponse(Enter evaluation phase, Pre-enrol in the program)	NotChainResponse	Enter evaluation phase	Pre-enrol in the program	1.000	0.101	0.101	1.000	0.524	0.524
AlternatePrecedence(Register for selection round, Enter evaluation phase)	AlternatePrecedence	Enter evaluation phase	Register for selection round	1.000	0.101	0.101	1.000	0.524	0.524
Precedence(Upload admission test score, Enter evaluation phase)	Precedence	Enter evaluation phase	Upload admission test score	1.000	0.101	0.101	1.000	0.524	0.524
AtMost1(Pay subscription fee)	AtMost1		Pay subscription fee	1.000	0.156	0.156	1.000	1.000	1.000
NotChainSuccession(Pay subscription fee, Create a candidate account)	NotChainSuccession	Pay subscription fee	Create a candidate account	1.000	0.228	0.228	1.000	1.000	1.000
NotChainSuccession(Pay subscription fee, Enter evaluation phase)	NotChainSuccession	Pay subscription fee	Enter evaluation phase	1.000	0.109	0.109	1.000	0.524	0.524
NotChainSuccession(Pay subscription fee, Register for selection round)	NotChainSuccession	Pay subscription fee	Register for selection round	1.000	0.109	0.109	1.000	0.524	0.524
NotChainSuccession(Pay subscription fee, Upload admission test score)	NotChainSuccession	Pay subscription fee	Upload admission test score	1.000	0.267	0.267	1.000	0.793	0.793
AtMost1(Pre-enrol in the program)	AtMost1		Pre-enrol in the program	1.000	0.156	0.156	1.000	1.000	1.000
AtMost3(Receive admission notification)	AtMost3		Receive admission notification	1.000	0.156	0.156	1.000	1.000	1.000
AlternatePrecedence(Enter evaluation phase, Receive admission notification)	AlternatePrecedence	Receive admission notification	Enter evaluation phase	1.000	0.023	0.023	1.000	0.132	0.132
NotChainPrecedence(Pay subscription fee, Receive admission notification)	NotChainPrecedence	Receive admission notification	Pay subscription fee	1.000	0.023	0.023	1.000	0.132	0.132
NotChainResponse(Receive admission notification, Pre-enrol in the program)	NotChainResponse	Receive admission notification	Pre-enrol in the program	1.000	0.023	0.023	1.000	0.132	0.132
NotResponse(Receive admission notification, Receive rejection notification)	NotResponse	Receive admission notification	Receive rejection notification	1.000	0.023	0.023	1.000	0.132	0.132
AtMost2(Receive rejection notification)	AtMost2		Receive rejection notification	1.000	0.156	0.156	1.000	1.000	1.000
AlternateResponse(Register for selection round, Enter evaluation phase)	AlternateResponse	Register for selection round	Enter evaluation phase	1.000	0.101	0.101	1.000	0.524	0.524
NotChainResponse(Register for selection round, Pre-enrol in the program)	NotChainResponse	Register for selection round	Pre-enrol in the program	1.000	0.101	0.101	1.000	0.524	0.524
NotChainResponse(Upload admission test score, Pre-enrol in the program)	NotChainResponse	Upload admission test score	Pre-enrol in the program	1.000	0.259	0.259	1.000	0.793	0.793
NotChainPrecedence(Pay subscription fee, Upload certificates)	NotChainPrecedence	Upload certificates	Pay subscription fee	1.000	0.261	0.261	1.000	0.713	0.713
NotChainResponse(Upload certificates, Pre-enrol in the program)	NotChainResponse	Upload certificates	Pre-enrol in the program	1.000	0.261	0.261	1.000	0.713	0.713

Figure 3: Discovered process specification table with quality measures

3.3 Considerations

To speed up the post-processing, other techniques than `hierarchyconflictredundancydouble` (say, `hierarchyconflictredundancy`, or just `hierarchy`, the default) can be used. However, this implies less accuracy in the detection (and removal) of the redundant constraints. Generally, the mining per se is very fast. The post-processing takes most of the computation time.