EDU-ProM: ProM for the Classroom

Yossi Dahari, Avigdor Gal, Arik Senderovich

Technion – Israel Institute of Technology
{dahari,avigal,sariks@}technion.ac.il

Abstract. We present EDU-ProM, an extension of the ProM framework, which was designed to serve as a classroom version of ProM for educational purposes. EDU-ProM is designed to work in a non-interactive work mode, allowing the execution of a set of mining tasks against numerous event logs without user intervention. It is mostly suitable for process mining researchers and students who are interested in exploring and extending existing ProM algorithms. EDU-ProM is a standalone open-source project, which requires simple setup to achieve a rich development environment for executing, creating, and extending mining techniques. The demo is aimed at academics who teach process mining in class, while using ProM as the basis for class and home assignments. In addition, the demo will interest researchers and practitioners who would like to run batch experiments using ProM, without the need to perform the often cumbersome dependency management required.

1 Introduction

ProM is an extensible framework that supports a wide variety of process mining techniques in the form of plug-ins [1]. The framework provides a popular tool for process mining, allowing users to add new event logs, define experimental settings, and add new mining techniques. Along with this rich functionality, ProM faces several challenges when introduced in a classroom.

The main challenge involves the necessity to use a graphical user interface (GUI), as presented in Figure 1. Running ProM via a GUI turns out to be an inefficient work mode. Specifically, when using ProM's GUI one is unable to *simultaneously* select a set of process mining methods that would be executed based on *multiple* event logs; only a single method and a single log are considered for every execution. Running several algorithms on multiple event logs is an important time-saving feature in many of the experimental settings that students face in their homework, and when researchers work on their scientific publications. In addition, mining techniques are configurable only via the GUI (*e.g.*, setting noise thresholds), making it impossible to alter existing mining techniques by overriding their functionalities. Finally, ProM development mode relies on plug-ins, making it easy for experienced users to add functionality to the tool, yet difficult to quickly setup a new working environment by combining several mining techniques in an interesting way.

Fig. 1: ProM GUI

One solution that attempts to solve the issues we raise above is ProM CLI[1]. The tool enables the submission of mining scripts that would allow batch mining with multiple event logs. However, beside the usual inefficiencies of working with scripts (*e.g.*, lack of debug tools, lack of abstraction capabilities, passing parameters, *etc.*), the scripting approach does not support compile-time error detection and the understanding of which mining techniques are supported by ProM CLI.

To alleviate these challenges, while avoiding the limitations of ProM CLI, we present EDU-ProM, a tool which was designed to serve the specific needs of students in classroom as well as researchers. EDU-ProM uses the ProM framework and its main plug-ins to support executing mining task non-interactively. EDU-ProM enables batch execution programmatically, creating an efficient work mode when conducting multiple experiments or testing results against benchmarks. The proposed tool provides a unified programmatic access to all main plug-ins, while being easy to set up. Once setup is completed, overriding or extending functionalities of all existing process mining techniques that ProM offers is enabled.

Table 1: A qualitative comparison of the four tools.

Tool	Batch Exec.	Extensibility	Multiple Plug-Ins
ProM GUI	No	No	No
ProM CLI	Partially Inefficient and unsafe	No	No
ProM Plug-ins/Open Source	Partially Per plug-in only	Yes	No
EDU-ProM	Yes	Yes	Yes

Table 1 provides a comparison of EDU-ProM and three other solutions, namely ProM GUI, ProM CLI, and ProM Open Source, across three dimensions: (1) safe

[1] `https://dirksmetric.wordpress.com/2015/03/11/tutorial-automating-process-mining-with-proms-command-line-interface/`

batch execution: the ability to execute several mining tasks in batch, (2) extensibility: writing novel solutions or extending existing ones, and (3) multiple plug-ins development mode. We observe that ProM GUI is incompetent across the three dimensions. The other two solutions (ProM CLI and ProMProM Plug-ins/Open Source) offer partial functionality, whereas EDU-ProM, provides a competent solution across all three dimensions.

The rest of this work is as organized follows. In Section 2, we review the tool architecture, followed by a demonstration of the tool (Section 3), and a conclusion (Section 4).

2 Architecture

EDU-ProM is a standalone Java project with several external dependencies (see below), supporting the reuse of existing mining tasks from the ProM framework and plugins. Figure 2 introduces the main components of EDU-ProM and its environment. For artifacts management and build we use Apache Maven alongside with Gradle. EDU-ProM configuration contains a list of external dependencies, which are fetched upon the initialization of the tool. The external dependencies are then compiled with EDU-ProM core and miners. EDU-ProM offers an abstraction layer for miners. Therefore, all common ProM utilities such as conformance checking, visualization, conversion *etc.*, reside in EDU-ProM core. EDU-ProM miners can then focus on specific mining tasks using EDU-ProM's core context.

EDU-ProM was designed to work in a non-interactive mode, using algorithms that can safely run without prompting the GUI. Such modus operandi supports the need to test benchmarks versus existing mining techniques. Testing may result in enormous executions that incorporate several event logs and mining algorithms. EDU-ProM provides a time preserving alternative to conducting experiments manually directly from ProM. It also avoids re-implementation and compilation of each of the desired algorithms and the time consuming task of setting up the experimental environment.

EDU-ProM was designed for extensibility: it enables accessing the most common ProM plug-ins such as InductiveMiner, HeuristicMiner, EvolutionaryTreeMiner *etc.* To simplify the extension, EDU-ProM is configured with more than 40 plug-ins. This helps in situations where students or researchers attempt to extend EDU-ProM, while all dependencies are in place and no environment change is needed. Moreover, EDU-ProM offers an abstraction layer where new mining techniques benefit from all existing features, including image exporting and conformance analysis, mainly for mining algorithms that result in Petri nets. A newly introduced mining algorithm requires only the implementation of the mining phase, while all other capabilities are available from the context. It is important to note that modern Integrated Development Environments (IDEs) have significantly evolved giving seamless access to decompilation tools, hence making every ProM component visible and reusable.

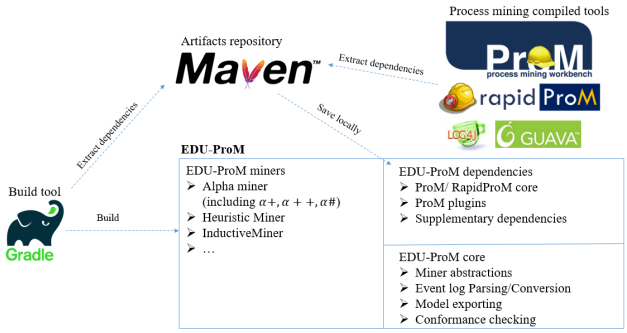


Fig. 2: EDU-ProM Architecture

3 Demonstration details

In the demonstration, we will show the usage of EDU-ProM in a classroom. First, we shall demonstrate the execution of mining techniques available in EDU-ProM (Figure 3) and how to evaluate their results (Figure 4). We will continue by showing examples of how EDU-ProM can be extended based on ProM implementation or by implementing a new prototype of a discovery algorithm. Specifically, we shall read an event log from a relative path *"EventLogs\gamma2.csv"*, from which a Petri net will be constructed (Figure 4a). The Petri net is then be exported into an image file, followed by an execution of a conformance checking phase. The diagnoses of the latter will also be presented (Figure 4b).

```
String filename = "EventLogs\\gamma2.csv";
IMiner miner = new InductiveMiner(filename);
miner.Mine(); // Reads the event log, and constructs a model
miner.Export(); // Creates a new PNG image file at /Output folder
miner.Evaluate(); // Saves conformance and alignment information
```

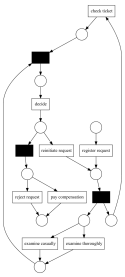
Fig. 3: Mining a Petri net with inductive miner using EDU-ProM

Implementing a new miner or extending an existing one requires mapping the parsed event log to a process model. For example, consider the integration of InductiveMiner into EDU-ProM (Figure 5) where the actual implementation only requires returning a Petri net. All other features of visualization, evaluation, *etc.* are taken care by the EDU-ProM context.

A screencast of the proposed demo is available at EDU-ProM- Process mining tool for educational usage. This video illustrates several examples and provides a brief explanation of the tool settings and main use-cases. The public repository and getting started instructions are available at: EDU-ProM public repository.

4 Conclusion

We presented EDU-ProM, an open source tool, which introduces a simple setup to achieve a rich development environment for executing, creating or extend-



(a) The Mined Petri Net

```
Calculation Time (ms), value= 2.5
Raw Fitness Cost, value= 0.0
Num. States, value= 12.5
Trace Fitness, value= 1.0
Move-Model Fitness, value= 1.0
Move-Log Fitness, value= 1.0
Trace Length, value= 7.0
Queued States, value= 34.0
precision 0.837302
generalization 0.825340
```

(b) Alignment and conformance information, as written to the log

Fig. 4: Result of executing InductiveMiner as presented in Section 3

```
@Override
protected PetrinetWithMarkings TrainPetrinet() throws Exception {
    logger.info( s: "Started mining a petri nets using inductive miner");
    Object[] res = IMPetriNet.minePetriNet(_log, _parameters, _canceller);
    PetrinetWithMarkings pn = new PetrinetWithMarkings();
    pn.petrinet = (PetrinetImpl)res[0];
    pn.initialMarking = (Marking)res[1];
    pn.finalMarking = (Marking)res[2];

    return pn;
}
```

Fig. 5: Inductive miner EDU-ProM implementation.

ing mining techniques. Many process mining techniques are easily accessible based on their ProM implementation, and can be safely executed without interaction. Access to such tools in a classroom enables students to focus on the task at hand rather than struggling with tedious, unnecessary setup and package management. Using EDU-ProM based batch execution simplifies the procedure of conducting experiments that involve multiple event logs and several mining techniques. Currently, EDU-ProM is being used by students who take an introductory course to business process management & mining at the Technion. It also supports graduate students who conduct research in the area of process mining.

Bibliography

[1] van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P. In: The ProM Framework: A New Era in Process Mining Tool Support. Springer Berlin Heidelberg, Berlin, Heidelberg (2005) 444–454