Setup instructions for the usage of the tool "Plan-based Data Aware Declarative Conformance Checker"

Document linked to the paper: "Aligning Data-Aware Declarative Process Models and Event Logs", submitted at BPM 2021, Foundational track

15 march 2021

Setup instructions for the usage of the tool "Plan-based Data Aware Declarative Conformance Checker

Document linked to the paper: "Aligning Data-Aware Declarative Process Models and Event Logs", submitted at BPM 2021, Foundational track

We have developed a planning-based alignment tool in Java that implements the technique discussed in the main paper to find optimal alignments of event logs and data-aware declarative process models. To see the tool in action while running one of the experiments enacted in the range of the paper, **it is strongly suggested** to **watch the associated screencast: HOW_TO_USE_THE_ALIGNER.mp4**

1. PREREQUISITES TO RUN THE TOOL

The tool, which is integrated with the SymBA*-2 planning framework (https://www.ida.liu.se/~TDDD48/labs/2015/planners.en.shtml), and has been developed with Linux Ubuntu (version 12.04 - 32 bit). All features provided by the tool should work without restrictions also under other 32 bit versions of Linux, provided that the following dependencies are satisfied:

- to run the tool, at least Java 1.7 is required
- to run the planner, Python 2.7 or Python >= 3.2 is required.

These dependencies are often satisfied by default on the majority of the development machines. If not, open the terminal of Linux and launch the following command:

➢ Installing Java 1.7:

```
$ sudo apt-get install openjdk-7-jre
```

➢ Installing Phyton:

```
$ sudo apt-get install mercurial g++ make python flex bison cmake
```

2. LAUNCHING THE TOOL

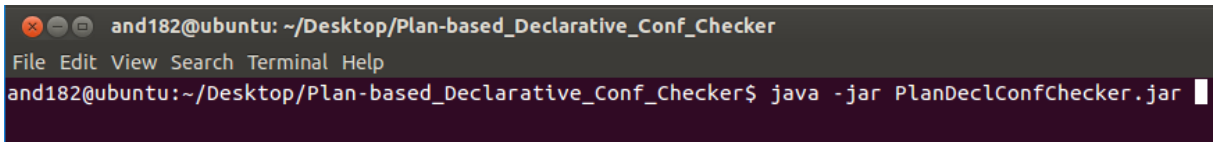
First of all, extract the folder "Plan-based_Data_Aware_Declarative_Conf_Checker" from the ZIP archive and copy it in a whatever folder of the user's Linux machine.

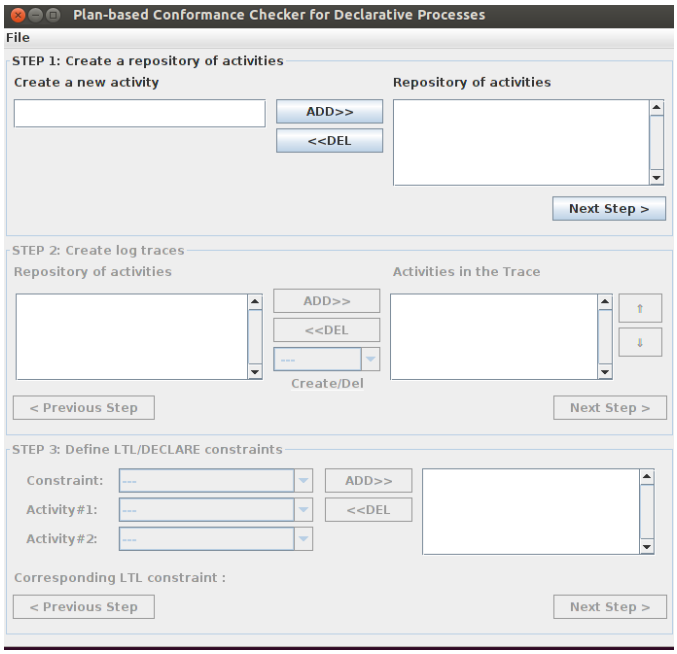
Running the tool requires to execute the JAR file "PlanDeclConfChecker.jar" contained into the " Plan-based_Data_Aware_Declarative_Conf_Checker " folder.

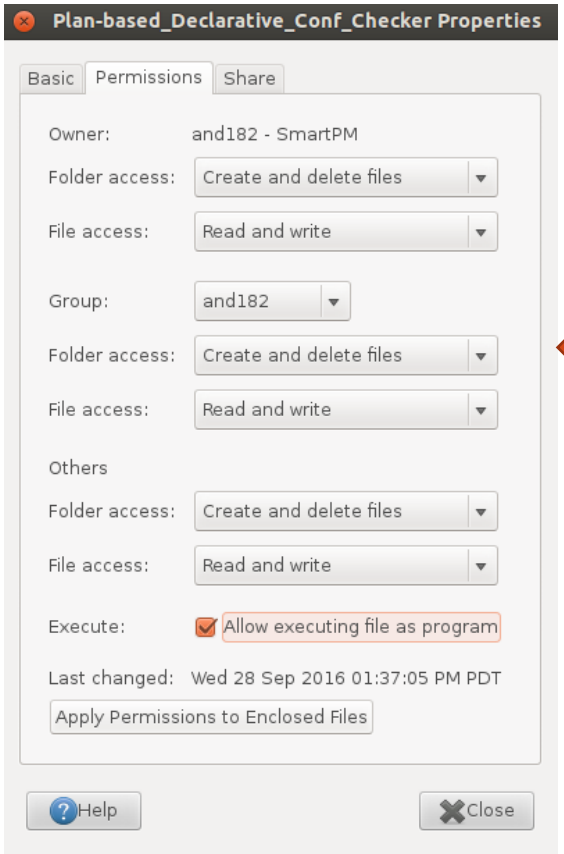
The execution can be performed by opening the Linux terminal, going into the tool's main folder and launching the following instruction:

```
$ java -jar PlanDeclConfChecker.jar
```

As a consequence, the GUI of the tool will be opened.







Check that the main folder/subfolders can be accessed in read/write mode!

ATTENTION: To see the tool in action while running one of the experiments enacted in the range of the BPM '21, paper, **it is strongly suggested to watch the associated screencast: HOW_TO_USE_THE_ALIGNER.mp4**

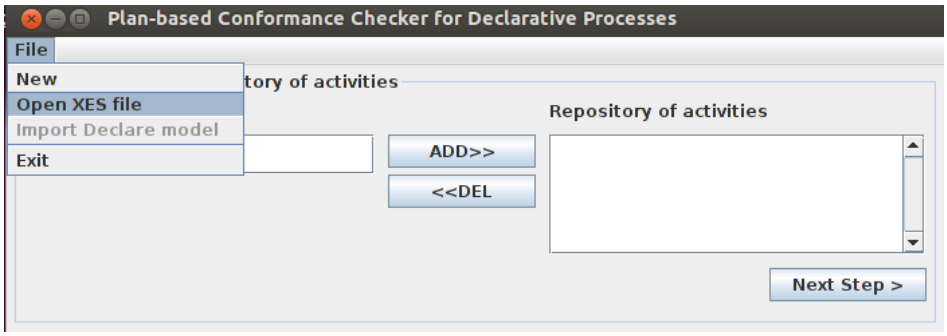
3. TOOL USAGE

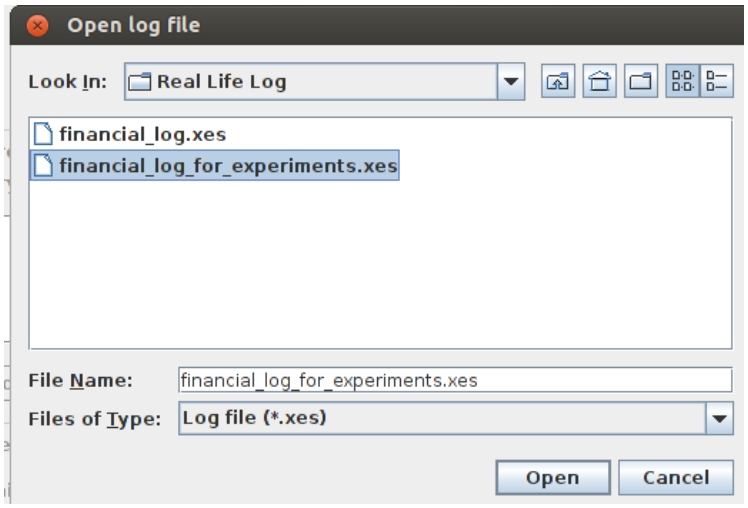
The usage of the tool requires to follow a four-step process driven by the tool's GUI:

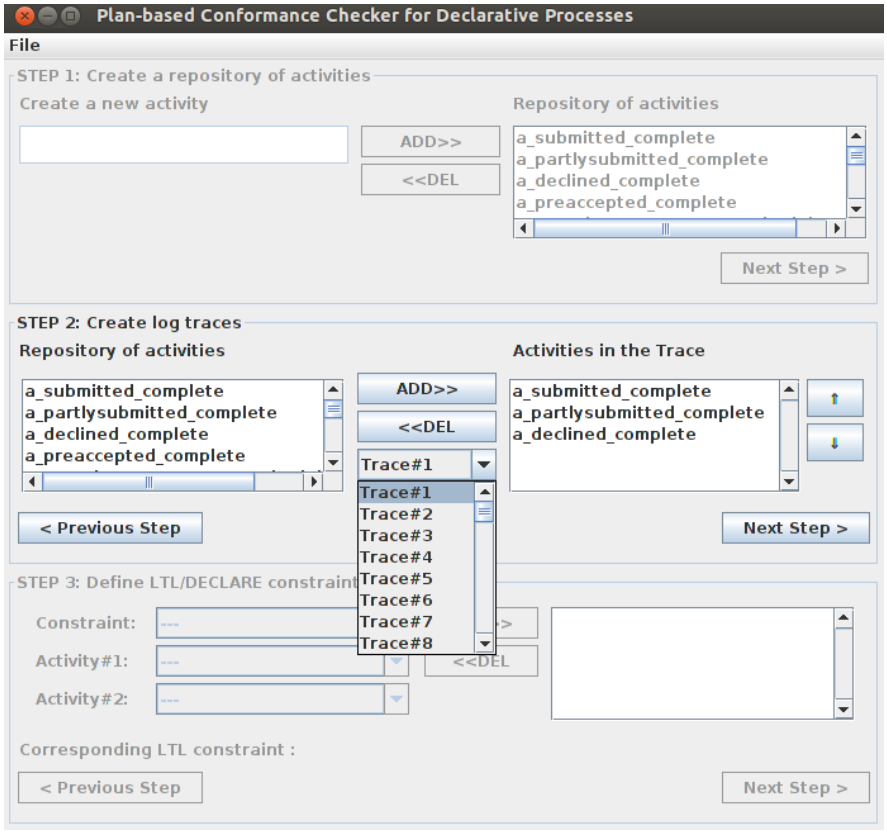
1. Specify an alphabet of activities that will be used for the construction of the traces.
2. Define the structure of every trace that composes the event log under construction.
3. Specify the DECLARE constraints that the log traces have to satisfy.

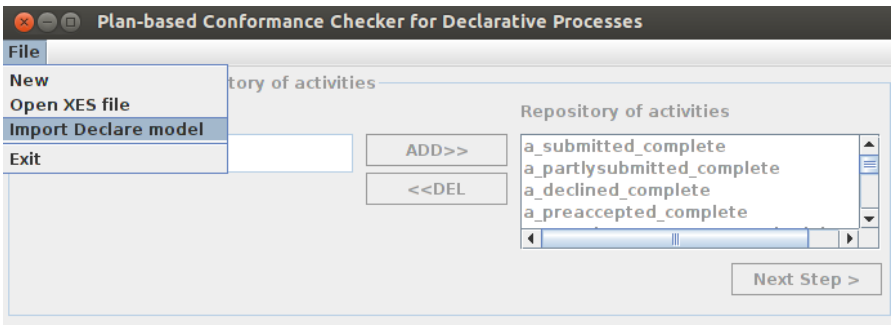
ATTENTION: The tool also enables to **automatize the steps (1-3)** by loading existing event logs formatted with the XES standard (through the item "Open XES File" of the menu called "File") and by importing:

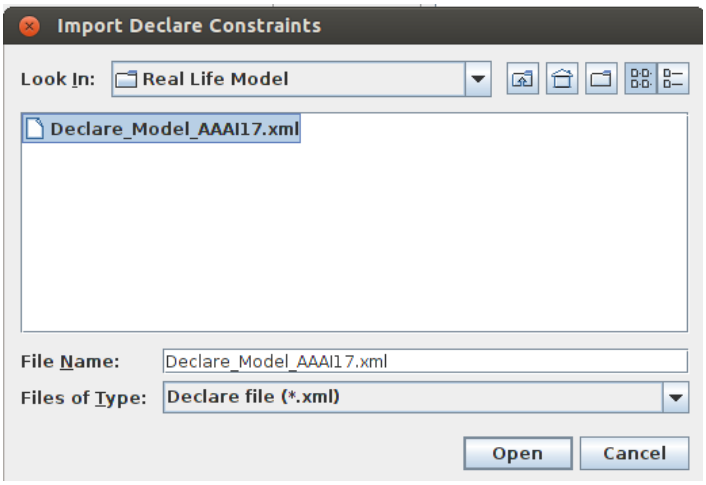
- Declare models (previously designed through a design tool, e.g., with the Declare Designer - http://www.win.tue.nl/declare/) and saved with the XML format.
- Deterministic Finite State automata saved with the DOT format.
- LTL formulas (saved as strings in TXT files) expressed according to the syntax shown in *FM Maggi et al. Runtime Verification of LTL-Based Declarative Process Models. IN: International Conference on Runtime Verification (2011)*





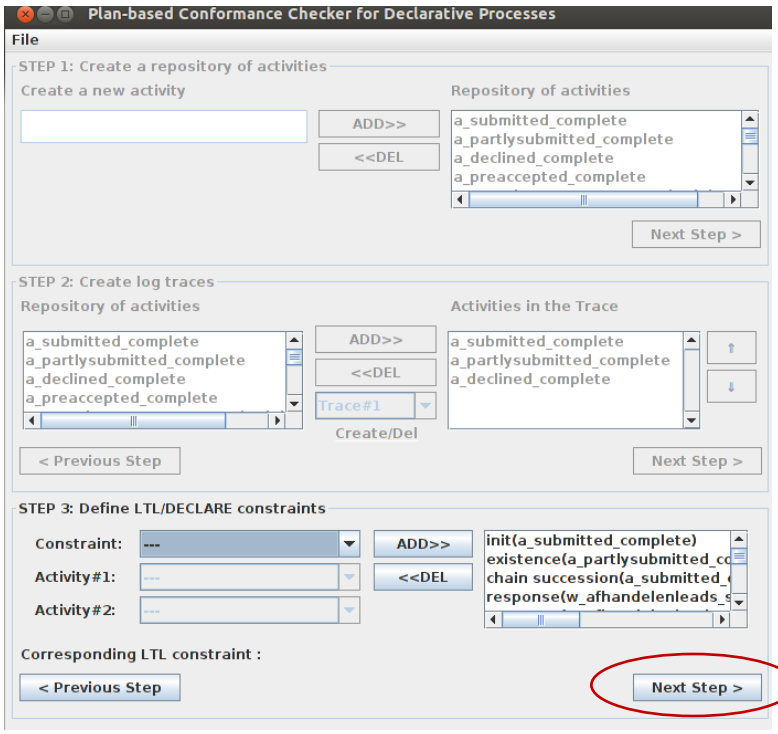

Plan-based Conformance Checker for Declarative Processes
File
STEP 1: Create a repository of activities
Create a new activity
ADD>>
<<DEL
Repository of activities
a_submitted_complete
a_partlysubmitted_complete
a_declined_complete
a_preaccepted_complete
Next Step >
STEP 2: Create log traces
Repository of activities
a_submitted_complete
a_partlysubmitted_complete
a_declined_complete
a_preaccepted_complete
ADD>>
<<DEL
Trace#1
Trace#1
Trace#2
Trace#3
Trace#4
Trace#5
Trace#6
Trace#7
Trace#8
Activities in the Trace
a_submitted_complete
a_partlysubmitted_complete
a_declined_complete
< Previous Step
Next Step >
STEP 3: Define LTL/DECLARE constraint
Constraint:
Activity#1:
Activity#2:
<<DEL
Corresponding LTL constraint :
< Previous Step
Next Step >


Plan-based Conformance Checker for Declarative Processes
File
New
Open XES file
Import Declare model
Exit
Repository of activities
ADD>>
<<DEL
a_submitted_complete
a_partlysubmitted_complete
a_declined_complete
a_preaccepted_complete
Next Step >


Import Declare Constraints
Look In:
Real Life Model
Declare_Model_AAAI17.xml
File Name:
Declare_Model_AAAI17.xml
Files of Type:
Declare file (*.xml)
Open
Cancel

In case of a declarative model including data conditions, the user should select the menu item “Import Data Atom List”, selecting the TXT file reporting the atoms associated to the activity of the model of interest.

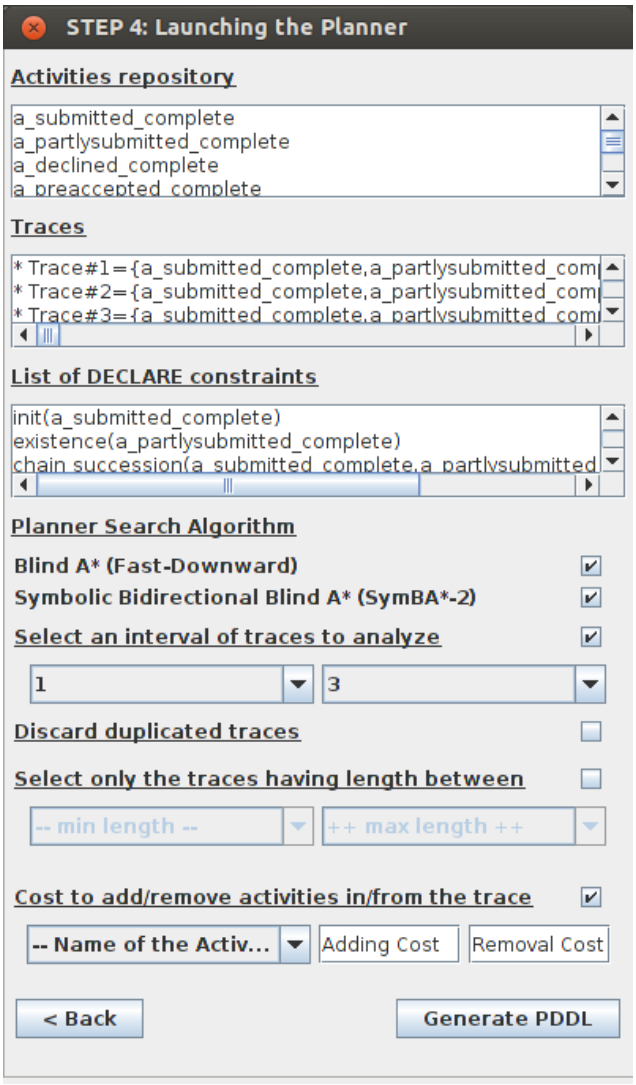
Now the user can proceed to the **Next Step** (see the picture below).



ATTENTION: For each declarative models there are many potential logs associated to it. To check that a declarative model is compliant with a log, it is required to import logs related to the specific declarative model used. The above pictures are related to the loading of the Declare model of a bank and a real-life log (without additive noise) associated to it.

At this point, the user:

a) must select the type of heuristics to search for a valid optimal plan alignment. The current version of tool provides a **Bidirectional A* search strategy** (through the SymBA*-2 planner).
b) can define (optional feature) a specific interval of log traces to be analysed.
c) can decide to discard the duplicated log traces. If selected, this option allows to discard the log traces already checked during the tool usage.
d) can decide (optional feature) to concentrate the analysis over log traces having a specific length.
e) can specify the cost of adding or removing an activity in the log trace under interest. By default, this option is already selected, and a cost of 1 is associated to the adding/deletion of any activity.



Once customized the kind of analysis to be performed, the user can generate the PDDL files required to run the tool in batch mode. If the '**Generate PDDL'** button is pushed, the planning domain/problem files related to the Declare model/event log imported are generated and stored in some internal folders. To check the performance of the planner with the above encoding, the user can run the following script from the Linux terminal. Specifically:

-- `run_SYMBA_ML_all` : runs the SymBA*-2 planner with the Bidirectional A* heuristics for all the PDDL problems generated.

Furthermore, the user can run another script to invoke the planner on a specific trace:

-- `run_SYMBA i` : runs the SymBA*-2 planner with the Bidirectional A* heuristics for the i-th problem generated.