Companion technical report to “A Purpose-Guided Log Generation Framework”

1 Process discovery via order relations

Listing 1.1 provides the pseudocode of the evaluator implemented in PURPLE for the process discovery via order relations purpose. The evaluator function takes as input (line 1) the coverage, i.e., a number from 0 to 1 indicating the percentage of order relation to be *covered* in the final log, and the *footprint* matrix of the input model. Then, in case the log does not contain traces yet (line 3), the function returns an empty set as delta, triggering a random simulation. Otherwise, the function calculates the footprint matrix of the current log (line 5), and compare it to the one of the model to retrieve only the relation that are still missing (line 6). Therefore, if the number of missing relations divided by the number of relations in the model exceeds the required coverage, the function returns and stops the log generation (line 8). While, if it is not the case, each missing relation is analyzed separately (lines 9-17). Depending on the type of relation (sequence, parallel, or choice) (lines 10,12,15) the delta is increased with traces composed of the activities involved in the missing relation. Specifically, if it is a sequence relation, the resulting trace contains the first member of the relation followed by the second (line 11). If it is a parallel relation, the function adds in the delta two traces with the activities in the relation in any order (lines 13 and 14). Finally, if it is a choice relation, the delta is increased with 2 traces each of which contains an event labeled with the activities involved in the relation (lines 16 and 17). The resulting delta is therefore returned to the simulator (line 18).

```
1  $\textsc{orderRelations}(coverage, footprint)$ :
2      $delta = \emptyset$
3      if $|log| = 0$
4          return $delta$
5      $currentFootprint := \textsc{calculateFootprint}(log)$
6      $missingRelations := footprint \setminus currentFootprint$
7      if $\frac{|missingRelations|}{|footprint|} \geq coverage$
8          return
9      for $relation$ in $missingRelations$
10         if $relation.type =$ “$\rightarrow$”
11             $delta := delta \cup \{\langle relation.first, relation.second\rangle\}$
12         else if $relation.type =$ “$||$”
13             $delta := delta \cup \{\langle relation.first, relation.second\rangle\}$
14             $delta := delta \cup \{\langle relation.second, relation.first\rangle\}$
15         else if $relation.type =$ “#”
16             $delta := delta \cup \{\langle relation.first\rangle\}$
17             $delta := delta \cup \{\langle relation.second\rangle\}$
18     return $delta$
```

Listing 1.1. Evaluator for process discovery via order relations (given a context $\langle lts, log\rangle$).

2 Process discovery via frequencies

Listing 1.2 provides the pseudocode of the evaluator implemented in PURPLE for the process discovery via frequencies. The evaluator function (line 1) takes as input a minimum number of traces to generate, the traces that the input model can generate, and the user selections for trace frequencies and loop repetitions. Firstly, the function iterates over the traces the model can generate (line 3) and checks if its frequency in the current log (line 4) is lower than requested (line 6). In this case, if it contains a loop, the trace is modified by repeating the loop a number of times lower than requested (line 8) and then added to the delta (line 9). Then, if the resulting delta contains traces it is returned to the simulator (line 12), while in case the delta is empty (since traces frequencies are sufficient) and the log size is greater than requested (line 10) the function returns and stop the log generation.

```
1  REALTIONSFREQUENCY($minT, traces, traceFrequencies, loopMaxRepetitions$) :
2      $delta := \emptyset$
3      for $t$ in $traces$
4          $currentFrequency :=$ GETFREQUENCY($t, log$)
5          $tFrequency := traceFrequencies[t]$
6          if $currentFrequency < tFrequency$
7              if HASLOOP($t$)
8                  $t := repeatLoop(t, loopMaxRepetitions)$
9              $delta := delta \cup \{t\}$
10     if $delta = \emptyset \wedge |log| > minT$
11         return
12     return $delta$
```

Listing 1.2. Evaluator for process discovery via frequencies (given a context $\langle lts, log \rangle$).

3 Conformance checking via noise frequencies

Listing 1.3 provides the pseudocode of the evaluator implemented in PURPLE for conformance checking via noise frequencies. The evaluator function (line 1) takes as input the number of traces to generate, the frequencies of each type of noise, and a precision value from 0 to 1. Immediately, the evaluator checks if the current log is empty, in this case, it returns an empty delta to get back a random trace (lines 2-4). Otherwise, the function calculates the current noise frequencies from the current log, and checks which of these frequencies is the lowest with respect to the user choice (lines 5 - 12). In doing so, the function saves temporally in the variable *minimum* the lowest frequency, and in *noiseType* the corresponding type of noise. Then, the function checks if the minimum frequency reaches the requested precision and if the log size reaches the one requested (line 13). In this case, the function returns and stops the log generation. Otherwise, the evaluator retrieves from the log the last trace generated and adds to it the type of noise corresponding to the minimum frequency (line 15).

```
1  NOISEFREQUENCIES($numT, noiseFrequencies, precision$) :
2      $delta := \emptyset$
3      if $|log| = 0$
4          return $delta$
```

```
    $currentNoiseFrequencies := \textsc{calculateNoise}(log)$
    minimum := 1
    noiseType := null
    for $noise$ in $noiseTypes$
        $currentPrecision := \frac{currentNoiseFrequencies[noise]}{noiseFrequencies[noise]}$
        if currentPrecision < minimum
            minimum := currentPrecision
            noiseType := noise
    if $minimum = 1 - precision \wedge |log| \geq numT$
        return
    $\textsc{addNoise}(\textsc{getLastTrace}(log), noiseType)$
    return $delta$
```

Listing 1.3. Evaluator for conformance checking via noise frequencies (given a context $\langle lts, log \rangle$).

4 Conformance checking via fixed align cost

Listing 1.4 provides the pseudocode of the evaluator implemented in PURPLE for conformance checking via fixed align cost. The evaluator function (line 1) takes as input the number of traces to produce, the alignment cost required, the set of traces the model can produce, and the precision in reaching the exact alignment cost. Therefore, the function calculates the cost for aligning the current log with the traces the model can perform (via function CALCULATECOST) and compare it with the cost to reach considering the precision value. If the reached cost is sufficiently high and the log size exceeds the required dimension (line 2), the function returns and stops the log generation (line 3). Otherwise, it adds a random type of noise to the last trace added in the log (line 5).

```
1 $\textsc{fixedAlign}(numT, cost, traces, precision)$ :
2     if $\frac{\textsc{calculateCost}(log, traces)}{cost} > precision \wedge |log| > minT$
3         return
4     $delta := \emptyset$
5     $\textsc{addRandomNoise}(\textsc{getLastTrace}(log))$
6     return $delta$
```

Listing 1.4. Evaluator for conformance checking via fixed align cost (given a context $\langle lts, log \rangle$).