Repairing soundness properties in data-aware processes

Paolo Felli, Marco Montali, Sarah Winkler
Starting point
A holistic view of information systems

process model

dynamic constraints

event log
Starting point
A holistic view of information systems

conventional process mining

process model
dynamic constraints

event log
Starting point
A holistic view of information systems

- Data
- Event log
- Process model
- Dynamics
Starting point
A holistic view of information systems

data-aware/object-centric
process mining

process model

data

event log
dynamics
Starting point
A holistic view of information systems

- data-aware/object-centric process mining
- need of combining mining and reasoning

data
event log
dynamics
Why reasoning?
Adapted from [Mannhardt et al., Comput. 2016], studied in [___,CAiSE2022]
Why reasoning?
Adapted from [Mannhardt et al., Comput. 2016], studied in [____,CAiSE2022]
Why reasoning?
Adapted from [Mannhardt et al., Comput. 2016], studied in [____,CAiSE2022]
Why reasoning?
Adapted from [Mannhardt et al., Comput. 2016], studied in [____,CAiSE2022]
(multi-perspective) mining  modelling
Is the model “correct”?
Is the model “correct”?
The control-flow way
Is the model “correct”?

The control-flow way

1. Strip-off the data

- Pay
- Send fine
- Insert notification
- Collect credit
- Add penalty
- Notify
- Appeal to prefecture
- Send to prefecture
- Appeal to judge
- Pay
- Notify
- Pay
- Pay
- Fine received

Result prefecture
Is the model “correct”?  
The control-flow way

1. Strip-off the data
2. Encode control-flow into bounded Petri net (finite state-space)
Is the model “correct”?  
The control-flow way

1. Strip-off the data
2. Encode control-flow into bounded Petri net (finite state-space)
3. Explore the state space

Verdict: all good!
Is the model “correct”?
The integrated way
Is the model “correct”?
The integrated way

infinitely many runs with infinitely many distinct variable assignments
Is the model “correct”?

Pay

Send fine

Insert notification

Add penalty

Notify

Appeal to prefecture

Result prefecture

Collect credit

Appeal to judge

0 ≤ dpw ≤ 60 days → dwr ≥ 2

0 ≤ djw ≤ 60 days ∧ dwr ≥ 0 → dwr = 2

0 ≤ dsw ≤ 90 days ∧ ewr ≥ 0 → dwr = 1

pr = 0 ∧ wr ≥ ar

dr ≠ 0 ∨ (pr = 0 ∧ wr ≥ ar)

tr ≥ ar + er

dr = 0

dw ≥ 0

tr < ar + er

The model is correct.

Send to prefecture

Result prefecture
Is the model “correct”?

Verdict: NO!
Process stuck if “send to prefecture” writes \( d > 1 \)
Is the model “correct”? 

Verdict: NO! 
Process stuck if “send to prefecture” writes \( d > 1 \)

Issue: blocked state

Reached state
(with some data)
Is the model “correct”?

**Verdict: NO!**
Process stuck if “send to prefecture” writes d > 1

**Issue:** blocked state

- reached state (with some data)
- final state

- deadlock or livelock

```
d^w \geq 0
\text{Send to prefecture}
\text{Result prefecture}
d^r = 0
```

```
d^r = 1
\text{Result prefecture}
```

```
0 \leq dp^w \leq 60\text{days}
0 \leq dj^w \leq 60\text{days} \land d^w \geq 0
```

**Process stuck if “send to prefecture” writes d > 1**

- initial state
- final state
Process mining is an iterative process
Process mining is an iterative process
Process mining is an iterative process.
Data-aware process mining with separate techniques
Data-aware process mining with separate techniques

control-flow mining
Data-aware process mining with separate techniques

control-flow mining

guards/decision mining
Data-aware process mining with separate techniques

control-flow mining

guards/decision mining
Data-aware process mining with separate techniques

control-flow mining

unsound!

guards/decision mining
Data-aware process mining

- control-flow mining
- guards/decision mining
Data-aware process mining with reasoning

- Control-flow mining
- Modelling
- Soundness repair
- Guards/decision mining
Data Petri Nets
[Mannhardt, PhD2018; _____, ER2018; _____, ACSD2019]

• Petri nets enriched with typed variables (ranging over infinite domains)
• Transitions access variables via read and write guards
• State: marking + variable assignment
• Transition firing: usual firing semantics + variable assignment update given a binding for the written variables

Infinite reachability graph even when the net is bounded
Possibility of reasoning depends on the guard language
Fragile setting: undecidability around the corner!
Goal

Data-aware soundness checks: using [____, CAiSE 2022]

Data-aware unsound DPN N has blocked states

soundness repair

data-aware sound DPN “minimally adapted” from N
Assumptions

- Data-aware unsound DPN N has blocked states.
- Soundness repair.
- Data-aware sound DPN "minimally adapted" from N.
1. Underlying Petri net (without data) is sound

Data-aware unsound DPN \( N \) has blocked states

**Assumptions**

- Soundness repair

Data-aware sound DPN “minimally adapted” from \( N \)
Assumptions

1. Underlying Petri net (without data) is sound

2. Guard language in a fragment where soundness can be checked [___, CAiSE 2022]
   E.g.: variable-to-constant guards

data-aware unsound DPN N has blocked states

soundness repair

data-aware sound DPN “minimally adapted” from N
Assumptions

1. Underlying Petri net (without data) is sound

2. Guard language in a fragment where soundness can be checked [___, CAiSE 2022] E.g.: variable-to-constant guards

3. Does not modify control structure, only guards

Data-aware unsound DPN N has blocked states

Soundness repair

Data-aware sound DPN “minimally adapted” from N
The two views of a process model…

… and what “minimality” means!

process representation

Minimal number of interventions on guards

[Zavatteri et al., FM-BPM 2023]

Our approach: minimal impact on behavior

We only impact traces leading to a blocked state
control-flow infrastructure: sound!
execution
or
write
read

if
if

execution

blocked state
Execution

First strategy: restrict behavior

write

read

If write

If read

or

Execution

Blocked state
First strategy: restrict behavior
First strategy: **restrict behavior**

**execution**
Second strategy: extend behavior

execution

blocked state
Second strategy: extend behavior

execution

blocked state
Second strategy: extend behavior
How to?

Step 1. Formula to characterise blocking runs

bounded Data Petri Net

Symbolic representation of reachable states

constraint graph

Data-aware soundness

• There are no dead tasks
• The final marking is only reached in a clean way for some variable assignment
• In every reachable marking, it must be possible to reach the final marking for some variable assignment

SMT
How to?

Step 1. Formula to characterise blocking runs

- **bounded Data Petri Net**

  Symbolic representation of reachable states

  Constraint graph

- **Extract formula**

  $\phi_{\text{blocked}}$ capturing exactly those runs that get stuck

**Data-aware soundness**

- There are no dead tasks
- The final marking is only reached in a clean way for some variable assignment
- In every reachable marking, it must be possible to reach the final marking for some variable assignment
Step 2. Carefully iterate over blocked states, using $\varphi_{\text{blocked}}$ to minimally avoid/unblock them.

**Restriction:** avoid blocked states by tightening guards

**How to modify guards?**
- Retrieve formula $\varphi_{\text{blocked}}$
- Let $a$ be a transition leading to that blocked state
- **Update** $\text{guard}(a) = \text{guard}(a) \land \neg \varphi_{\text{blocked}}$

**Extension:** let blocked states proceed by relaxing guards

**How to modify guards?**
- Retrieve formula $\varphi_{\text{blocked}}$
- Let $a$ be a transition leading to that blocked state
- **Update** $\text{guard}(a) = \text{guard}(a) \lor \varphi_{\text{blocked}}$
How to?

Step 2. Carefully iterate over blocked states, using $\varphi_{\text{blocked}}$ to minimally avoid/unblock them.

Restriction: avoid blocked states by tightening guards.

Expansion: let blocked states proceed by avoiding guards.

Can be applied to general DPNs, but may not terminate.

Termination guaranteed for DPNs using variable-to-constant guards.
Back to the road fine example

\[ d^w \geq 0 \]

Send to prefecture

Result prefecture

\[ d^r = 0 \]

\[ d^r = 1 \]
Back to the road fine example

\[ d^w \geq 0 \]

\[ d^r = 0 \]

\[ d^r = 1 \]

\[ \varphi_{blocked} = d > 0 \land d \neq 0 \land d \neq 1 \]

Send to prefecture

Result prefecture
Back to the road fine example
Restriction: modify the write guard on “send to prefecture”

\[
\phi_{\text{blocked}} = d > 0 \land d \neq 0 \land d \neq 1
\]
Back to the road fine example

Restriction: modify the write guard on “send to prefecture”

\[
d^w \geq 0 \land \neg (d^w > 0 \land d^w \neq 0 \land d^w \neq 1) \\
\equiv \\
d^w = 0 \lor d^w = 1
\]

\[\varphi_{\text{blocked}} = d > 0 \land d \neq 0 \land d \neq 1\]
Back to the road fine example
Extension: nondet. pick one of the two choice guards and fix it

\[ d^w \geq 0 \]

Send to prefecture

\[ d^r = 0 \]

Result prefecture

\[ d^r = 1 \]

\[ \varphi_{\text{blocked}} = d > 0 \land d \neq 0 \land d \neq 1 \]
Back to the road fine example
Extension: nondet. pick one of the two choice guards and fix it

\[ d^w \geq 0 \]

Send to prefecture

Result prefecture

\[ d^r = 0 \]

\[ d^r = 1 \]

\[ \varphi_{\text{blocked}} = d > 0 \land d \neq 0 \land d \neq 1 \]
Back to the road fine example

Extension: nondet. pick one of the two choice guards and fix it

\[ d^w \geq 0 \]

Send to prefecture

\[ d^r = 0 \]

Result prefecture

\[ d^r = 1 \lor (d^r > 0 \land d^r \neq 0 \land d^r \neq 1) \]
\[ \equiv \]
\[ d^r > 0 \]

\[ \varphi_{\text{blocked}} = d > 0 \land d \neq 0 \land d \neq 1 \]
Fully implemented: soundness.adatool.dev

Model

<pre><xml version="1.0" encoding="UTF-8"?>
<pnml>
  <net id="net1" type="http://www.pnml.org/version-2009/grammar/pnmlcoremodel">
    <name>
      <text>Road-Fine Management</text>
    </name>
    <page id="n0">
      <place id="n1">
        <!-- Diagram content -->
      </place>
      <!-- More diagram content -->
    </page>
  </net>
</pnml></pre>
## Experiments

### Repair of unsound DPNs

<table>
<thead>
<tr>
<th>DPN</th>
<th>repair/Restrict</th>
<th>repair/Extend</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td># deadlocks</td>
</tr>
<tr>
<td>(a) road fines normative [MannhardtLRA16]</td>
<td>50s</td>
<td>2</td>
</tr>
<tr>
<td>(b) road fines mined [Mannhardt18]</td>
<td>24s</td>
<td>1</td>
</tr>
<tr>
<td>(c) dig. whiteboard/transfer [Mannhardt18]</td>
<td>2.1s</td>
<td>1</td>
</tr>
<tr>
<td>(d) package handling [Fig. 5, deLeoniFM21]</td>
<td>6s</td>
<td>0</td>
</tr>
<tr>
<td>(e) auction [FMW22a]</td>
<td>8s</td>
<td>1</td>
</tr>
<tr>
<td>(f) auction example</td>
<td>2.5s</td>
<td>1</td>
</tr>
<tr>
<td>(g) livelock example</td>
<td>2.1s</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Extension** takes more time (larger constraint graphs)

### Conformance checking (road fines example)

<table>
<thead>
<tr>
<th></th>
<th>original DPN</th>
<th>after repair/Restrict</th>
<th>after repair/Extend</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>average distance</strong></td>
<td>1.2009</td>
<td>1.2009</td>
<td>1.1305</td>
</tr>
</tbody>
</table>

- **Repair does not affect conformance negatively**
Conclusions

Take home

• Need of **soundness repair** in **data-aware process mining**
• **SMT-based automatic repair** for DPNs that is **conservative on the control-flow** and on the original behavior
• Two repair directions: **restrict** or **extend**
• **Fully implemented**

Next steps

• **Blending** of restriction and extension, user-in-the-loop
• **Log-driven data-aware repairs**
Repairing soundness properties in data-aware processes

Paolo Felli, Marco Montali, Sarah Winkler