Topics

• Recent Work
  • Online Conformance Checking
    – Big Mine 2017 @ KDD

• Current Work
  • Online Filtering
  • With Marcello La Rosa / Raffaele Conforti @ QUT
Topics

• Recent Work
  • Online Conformance Checking
    – Big Mine 2017 @ KDD
Process Mining

Discovery

Conformance Checking

Enhancement
Conformance Checking

\(<a,b,c,d,e,g>\)
Conformance Checking

\(<a,d,c,d,e,g>\)
Alignments

Log file:

<table>
<thead>
<tr>
<th>a</th>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>e</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

Model:

| a | b | d | c | e | g |

Diagram:

- Start (a) -> Register Call (b) -> Take Call (c)
- p2 Connects to c
- p4 Connects to e
- e Connects to g
- d Connects to Reset Modem (p3)
- p5 Connects to f
- p6 Connects to h
- Register Success (g)
- Dispatch (h)
Online Conformance Checking

Event stream
Online Conformance Checking

start → Register Call → p1 → Take Call

p2 → Check Line → p4 → Check Connectivity

c

d → Reset Modem → p3 → p5 → Re-Test

p6 → g → Register Success → e → f → h → Dispatch

Event stream

TU/e Technische Universität Eindhoven University of Technology

/ Mathematics & Computer Science
Online Conformance Checking

Diagram:

- Start
- Register Call (p1)
- Take Call
- Check Line (p2)
- Reset Modem (p3)
- Check Connectivity (p4)
- Re-Test (p5)
- Dispatch (p6)
- Register Success

Event stream:

- Register Call
- Check Line
Online Conformance Checking
Online Conformance Checking

Event stream

Register Call
Check Line
Reset Modem
Reset Modem
Online Conformance Checking
Prefix Alignments

Diagram showing a sequence of events and states.

Event stream:
- Start
- Register Call
- Take Call
- Reset Modem
- Re-Test
- Dispatch

States:
- a
- b
- c
- d
- e
- f
- g
- h

Transitions:
- p1
- p2
- p3
- p4
- p5
- p6
Prefix Alignments
Prefix Alignments

\[ a \]

\[ a \]
Prefix Alignments
Prefix Alignments
Prefix Alignments
Incremental Scheme

1. Foreach new activity from the stream
   1. Try to Execute Directly (synchronous move)
   2. If not unknown activity:
      1. Solve prefix-alignment search problem
      3. Else perform a label move (>>)

- Register Call
- Check Line
- Reset Modem
- Reset Modem
Incremental Scheme
Incremental Scheme

Register Call

Reset Modem

Event stream

start
Register Call
p1
Take Call

a

b

p2
Check Line

c

p4
Check Connectivity

d
Reset Modem

p3

e

f

Re-Test

p5

h

Dispatch

g

Register Success

/ Mathematics & Computer Science
Basic Incremental Scheme

Event stream

Register Call

Reset Modem

start

Register Call

p1

Take Call

b

Check Line

c

p2

Check Connectivity

e

p4

Dispatch

g

Register Success

end

p6

Re-Test

f

p5

Reset Modem

d

p3
Incremental Scheme

Event stream

Register Call

Reset Modem

\[ a \quad d \]

\[ a \quad >> \]

\[ a \quad >> \quad d \]

\[ a \quad b \quad d \]
Parametrization #1
Upper Bound

\[ a \gg d \]

\[ a \quad b \quad d \]

\[ \gamma = x \]

\[ \text{cost}(\gamma) = x \]
Parametrization #1
Upper Bound

\[
\begin{array}{ccc}
a & >> & d \\
a & b & d \\
\end{array}
\]

\[
\text{cost}(\gamma) = x
\]

Receive e
Parametrization #1
Upper Bound

$\gamma$ is upper bounded by: $\text{cost}(\gamma) + \text{cost}(e, \gg)$
- Any path results with larger costs can directly be ignored (pruning)
- We can solve the shortest path problem *from any state in the model*
- We revert the alignment to a pre-defined length (window size)
- We can solve the shortest path problem from any state in the model
- We revert the alignment to a pre-defined length (window size)
Experimental Evaluation - Settings

- For artificial data sets:
  1. Generate a process model
  2. Generate stream from the model (100% fitting the model)
     - 18 different models, 18 different event streams (324000 traces, 44,537,728 events)
  3. Add artificial noise (i.e. to “break” the data)

- For a real data set:
  1. An event log from Dutch hospital treatment of patients suspect of having sepsis
     - 15214 events divided over 1050 cases
     - Skewed trace lengths (majority below 30)
  2. Generate a process model out of 90% of the data set

- Compute prefix-alignments
Results

![Graph showing average enqueued states against percentage noise. The graph compares different noise conditions labeled as True and False, with bars indicating upper bounds.]
Results

![Bar chart showing average queue size vs. percentage of noise. The chart has bars for both true and false conditions. The y-axis represents the average queue size, ranging from 0 to 60. The x-axis represents the percentage of noise, ranging from 0.0 to 0.3. The upper bound is indicated for both true and false conditions.]
Results

![Graph showing average traversed arcs vs. noise percentage for True and False cases](image)

Upper bound

- True
- False

Average traversed arcs:
- 0.0%
- 0.1%
- 0.2%
- 0.3%
Results

![Bar chart showing average visited states vs. noise percentage. The chart compares true and false conditions with an upper bound.]
Results

![Graph showing the relationship between average enqueued states and prefix length, with upper bounds for True and False states.](image)

- Upper bound
- True
- False
Results

![Graph showing average queue size versus prefix length with upper bound for both True and False cases.](image-url)
Results

![Graph showing the relationship between noise percentage and average enqueued states for different window sizes.](image-url)
Results
Results – real data set

![Graph showing the relationship between average alignment cost and prefix length for different versions with varying window sizes. The graph compares conventional alignment with different prefix versions, highlighting the impact of window size on alignment cost.]
Results – real data set
Conclusion

- Prefix-alignments underestimate true alignment cost

- Upper bounds
  - Effective pruning

- Windows
  - Reduce memory usage further
  - Over-approximate \textit{optimal prefix alignments}
Future Work

• Investigate trade-off between over-approximation versus true alignment costs

• Develop techniques with theoretical memory bounds
Topics

• Recent Work
  • Online Conformance Checking
    – Big Mine 2017 @ KDD

• Current Work
  • Online Filtering
  • With Marcello La Rosa / Raffaele Conforti @ QUT
• July – Mid August
QUT - Goal

Event stream

Register Call  Take Call  Reset Modem  Reset Modem

TU/e
Technische Universiteit Eindhoven
University of Technology

/ Mathematics & Computer Science
QUT – Actual Work

Event stream

Register Call → Take Call → Reset Modem → Reset

Event stream

Register Call → Take Call → Reset Modem

 TU/e 
Technische Universiteit Eindhoven
University of Technology

/ Mathematics & Computer Science

13-10-2017 PAGE 50
QUT – Actual Work

Event stream

Register Call  Take Call  Reset Modem  Reset Modem

Event stream

Register Call  Take Call  Reset Modem

TU/e
Technische Universiteit
Eindhoven
University of Technology
QUT – Global Architecture

Input Event Stream $S_i$  

Sliding Window $w$

New Event $e$

Event $e'$

Apply Filter $f$

Output Event Stream $S_o$

Delay $d$

Drop $e''$
QUT – The Filter (That does not work)

1. Maintain a Prefix-Tree (over time behaviour is removed)
QUT – The Filter (That does not work)

1. **Maintain a Prefix-Tree** (over time behaviour is removed)
   - case-id

2. **Receive new event** $e = (c,a)$
   - activity
1. Maintain a Prefix-Tree (over time behaviour is removed)

2. Receive new event \( e = (c,a) \)

3. For \( c \), fetch previously received event in the tree.
QUT – The Filter (That does not work)

1. Maintain a Prefix-Tree (over time behaviour is removed)

2. Receive new event \( e = (c, a) \)

3. For \( c \), fetch previously received event in the tree.

4. Assess whether \( a \) makes sense at that position
QUT – The Filter (That does not work)

1. Maintain a Prefix-Tree (over time behaviour is removed)

2. Receive new event \( e = (c,a) \)

3. For \( c \), fetch previously received event

4. Assess whether \( a \) makes sense
QUT – The Filter (That does not work)

1. Maintain a prefix tree (over time behaviour is removed).

2. Receive new event $e = (c, a)$.

3. For $c$, fetch previously received event in the tree.

4. Assess whether $a$ makes sense.
Global Architecture

Input Event Stream $S_i$ $\Rightarrow$ Sliding Window $w$ $\Rightarrow$ Apply Filter $f$ $\Rightarrow$ Output Event Stream $S_o$

New Event $e$ $\Rightarrow$ Delay $d$ $\Rightarrow$ Event $e'$ $\Rightarrow$ Drop $e''$

Time $\Rightarrow$ $\infty$
1. Maintain Traces (e.g. a Prefix-Tree)
QUT – The Filter #2

1. Maintain Traces (e.g. a Prefix-Tree)

2. Receive new event $e = (c,a)$
1. Maintain Traces (e.g. a Prefix-Tree)

2. Receive new event \( e = (c, a) \)

3. For \( c \), fetch previously received \( k \) events.
1. Maintain Traces (e.g. a Prefix-Tree)

2. Receive new event $e = (c, a)$

3. For $c$, fetch previously received $k$ events.
   1. Example:
      1. Past: <a,b,c,d>, Receive: e, k=2;
1. Maintain Traces (e.g. a Prefix-Tree)

2. Receive new event \( e = (c,a) \)

3. For \( c \), fetch previously received \( k \) events.
   1. Example:
      1. Past: \( <a,b,c,d> \), Receive: \( e \), \( k=2 \);
      2. Last 2 events: \( <c,d> \)
QUT – The Filter #2

1. Maintain Traces (e.g. a Prefix-Tree)

2. Receive new event \( e = (c, a) \)

3. For \( c \), fetch previously received \( k \) events.
   1. Example:
      1. Past: \( <a, b, c, d> \), Receive: \( e \), \( k=2 \);
      2. Last 2 events: \( <c, d> \)
      3. In memory:
         1. Increase counters \( <d> -> e, <c, d> -> e \)
QUT – The Filter #2

1. Maintain Traces (e.g. a Prefix-Tree)

2. Receive new event $e = (c, a)$

3. For $c$, fetch previously received $k$ events.

4. Assess whether $a$ makes sense at that position.
1. Maintain Traces (e.g., a Prefix-Tree)

2. Receive new event $e = (c, a)$

3. For $c$, fetch previously received $k$ events.

4. Assess whether $a$ makes sense.
1. Maintain Traces (e.g. a Prefix-Tree)

2. Receive new event \( e = (c, a) \)

3. For \( c \), fetch previously received \( k \) events.

4. Assess whether \( a \) makes sense at that position.

Parallelism!
QUT – The Filter; Problem 1

TOKEN IS SOMEWHERE BELOW…
QUT – The Filter; Problem I

TOKEN IS SOMEWHERE BELOW…
All are relatively infrequent

Token is somewhere below...
QUT – The Filter; Problem I

???

TOKEN IS SOMEWHERE BELOW...

ALL ARE RELATIVELY INFREQUENT
QUT – The Filter; Problem II